Theoretical Runtime Bounds for Information Spreading and a New Vehicle Routing Algorithm

Rami Daknama

Dissertation
an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig–Maximilians–Universität München

vorgelegt von
Rami Daknama
aus Starnberg

München, den 2. Mai 2018

Erstgutachter: Prof. Dr. Martin Schottenloher
Zweitgutachter: Prof. Dr. Holger Hoos
Tag der mündlichen Prüfung: 11. Oktober 2018

Contents

List of Figures

List of Tables

Abstract

In this thesis we investigate theoretical bounds on the runtimes of information spreading algorithms, study theoretical conditions for networks (i.e. graphs) that assure that information spreading cannot be corrupted and, as a third subject, formally define a vehicle routing problem and provide a local search algorithm to solve it.

The first part of this thesis explores the runtime of information spreading algorithms; in particular, we consider the three well-known algorithms *Push*, *Pull* and *Push&Pull*: Consider a graph G. Assume that in the beginning one node of G has a piece of information. In the *Push* setting, every round every informed node chooses a neighbour independently and uniformly at random, and, if the neighbour is uninformed, informs it. Similarly, in the *Pull* setting, every round every uninformed node chooses a neighbour independently and uniformly at random and, if the neighbour is informed, becomes informed itself. *Push&Pull* combines *Push* and *Pull*; each round each node chooses a neighbour independently and uniformly at random and if at least one of both nodes is informed, now both are informed.

A crucial task is to understand the random variables that count how many rounds are needed by these algorithms to inform all nodes; these random variables are called runtimes of the respective algorithms. Clearly they depend on the underlying graph. Interestingly, though the algorithms themselves can be formulated very easily, understanding their runtimes, even for simple graph classes (e.g. complete graphs), turns out to be a more complicated task. The first part of this thesis addresses questions concerning these runtimes. In particular, the following three aspects are considered.

First, we investigate the probability distribution of the runtime X_n of *Push* on the complete graph with n nodes. More specifically, we prove the following: Let $x(n) := \log_2(n) - \lfloor \log_2(n) \rfloor$ and let γ denote the Euler-Mascheroni constant. Furthermore, $c : \mathbb{R} \to \mathbb{R}$ denotes a function that is defined as a certain limit expression. Let $d(n) := \ln(n) + \gamma + c(x(n))$ and let $G \sim Gumbel_\gamma$, i.e. for all $x \in \mathbb{R}$ it is $P[G \leq x] = e^{(-e^{-(x+\gamma)})}$. Then there is a function $m : \mathbb{N} \to \mathbb{R}^+$ with $m(n) = o(1)$ (for $n \to \infty$) such that for all $k, n \in \mathbb{N}$ it holds $|P[X_n \geq k] - P[\lfloor \log_2(n) \rfloor + \lceil G + d(n) \rceil \geq k]| \leq m(n)$.

Second, on graphs with good expansion properties, we explore how robust the runtime of *Push* is. In particular, we investigate up to which fraction of edges can be deleted at each node without possibly slowing down *Push* by $\Omega(\ln(n))$ rounds. It turns out that *Push* is not robust at all in this sense, but if we instead only require that *almost* all nodes have to be informed, *Push* turns out to be very robust. We also prove respective results in the presence of independent message transmission failures.

Third, we determine the expected runtimes of *Pull* and *Push&Pull* if the underlying graph is changing over time (for *Push* a respective result already exists). Let $a \in \mathbb{R}^+$ and

set $p := a/n$. Each round we consider a newly (and independently of the previous graphs) sampled Erdős-Rényi random graph $G(n,p)$ as the underlying graph. We show that the expected runtime for *Pull* is $\log_{2-e^{-a}}(n) + \ln(n)/a + \mathcal{O}(1)$. Let $\kappa := 2(1-e^{-a}) - (1-e^{-a})^2/a$. For *Push&Pull* we prove that the expected runtime is $\log_{1+\kappa}(n) + \ln(n)/a + \mathcal{O}(1)$. We also obtain large deviation bounds.

In the second part of this thesis, we consider a model introduced by Alon et al. to investigate the question "How robust is the wisdom of the crowds?". In this model not only one piece of information but two competing pieces of information (one true, one false) spread in a graph. There are two kinds of adversaries (weak and strong) who are equipped with certain powers that they can use to support the falsehood. We prove various robustness results, where, loosely speaking, robustness means that the true piece of information prevails in spite of the adversary's efforts. In particular, we prove that if the minimum degree is suitably bounded from below, this guarantees robustness against the weak adversary. Moreover, for Erdős-Rényi random graphs, which are known to be robust against the strong adversary, we investigate up to which fraction of the edges can be adversarially deleted at each node without being able to destroy the robustness. We also prove a respective result if edges may not only be deleted but can be inserted as well. Finally we prove that from the strong adversary's perspective, the problem is NP-hard.

In the third part of this thesis, we consider a vehicle routing problem with two types of vehicles. In particular, trucks and drones are available to deliver packages as fast as possible to certain positions. All vehicles start at the depot and, at the end of their tours, they have to return there. Trucks can carry an arbitrary number of drones and packages. Drones can travel on a truck but they can also fly on their own. However, while flying, a drone can carry at most one package at a time and after having delivered that package it has to fly to a truck to recharge or it must return to the depot and stay there. Hence sometimes a truck has to wait for a drone or vice versa. This adds an interesting scheduling aspect to the problem. We provide a formal definition of the problem and prove an equivalent characterisation of the feasibility of a solution. Then we introduce a local search algorithm and evaluate it empirically.

Zusammenfassung

In dieser Arbeit beschäftigen wir uns mit theoretischen Schranken für die Laufzeiten von Informationsverbreitungsalgorithmen, untersuchen theoretische Bedingungen für Netzwerke (Graphen), die gewährleisten, dass die Informationsverbreitung nicht korrumpiert werden kann und definieren formal ein Vehicle-Routing-Problem für das wir einen Lokale-Suche-Algorithmus vorschlagen.

Der erste Teil dieser Arbeit, der die Laufzeit von Informationsverbreitungsalgorithmen untersucht, beschäftigt sich mit den drei bekannten Algorithmen *Push*, *Pull* und *Push&Pull*: Betrachten wir einen Graphen G. Nehmen wir an, dass ein Knoten von G zu Beginn eine Information besitzt. *Push* ist dadurch definiert, dass jede Runde jeder informierte Knoten unabhängig und gleichverteilt zufällig einen Nachbarn wählt und diesem die Information mitteilt, falls dieser uninformiert ist. *Pull* ist ähnlich definiert; jede Runde wählt jeder uninformierte Knoten unabhängig und gleichverteilt zufällig einen Nachbarn und erfährt von diesem die Information, falls der gewählte Nachbar informiert ist. *Push&Pull* kombiniert die Mechanismen von *Push* und *Pull*; jede Runde wählt jeder Knoten einen Nachbarn unabhängig und gleichverteilt zufällig. Wenn mindestens einer der beiden Knoten informiert ist, sind es danach beide.

Ein zentrales Problem ist es, die Zufallsvariablen, die die Anzahl benötigter Runden bis alle Knoten informiert sind angeben, für die beschriebenen Algorithmen zu analysieren. Diese Zufallsvariablen werden auch als die Laufzeiten der jeweiligen Algorithmen bezeichnet. Sie variieren offenbar stark für verschiedene zugrunde liegende Graphen. Obwohl die Algorithmen selbst so leicht zu formulieren sind, stellt es sich selbst für einfache Graphklassen (beispielsweise vollständige Graphen) als ungleich komplizierter heraus, die entsprechenden Laufzeiten zu bestimmen. Der erste Teil dieser Arbeit beschäftigt sich mit Problemen, die diese Laufzeiten betreffen. Konkret werden wir die folgenden drei Aspekte untersuchen.

Erstens untersuchen wir die Wahrscheinlichkeitsverteilung der Laufzeit X_n von *Push* auf dem vollständigen Graphen mit n Knoten. Dabei beweisen wir Folgendes: Sei $x(n) := \log_2(n) - \lfloor \log_2(n) \rfloor$ und bezeichne γ die Euler-Mascheroni-Konstante. Ferner ist $c : \mathbb{R} \to \mathbb{R}$ eine Funktion, die wir als einen bestimmten Grenzwert definieren werden. Sei $d(n) := \ln(n) + \gamma + c(x(n))$ und sei $G \sim \mathit{Gumbel}_\gamma$, d. h. für alle $x \in \mathbb{R}$ ist $P[G \leq x] = e^{(-e^{-(x+\gamma)})}$. Dann gibt es eine Funktion $m : \mathbb{N} \to \mathbb{R}^+$ wobei $m(n) = o(1)$ (für $n \to \infty$), so dass für alle $k, n \in \mathbb{N}$ gilt $|P[X_n \geq k] - P[\lfloor \log_2(n) \rfloor + \lceil G + d(n) \rceil \geq k]| \leq m(n)$.

Zweitens beschäftigen wir uns auf Graphen mit guten Expansionseigenschaften mit der Robustheit der Laufzeit von *Push*. Genauer gesagt untersuchen wir, welcher Anteil an Kanten maximal pro Knoten gelöscht werden darf, ohne dass es dadurch möglich ist, *Push* um $\Omega(\ln(n))$ Runden zu verlangsamen. Es stellt sich heraus, dass *Push* in diesem Sinne nicht ro-

bust ist; wenn wir jedoch nur verlangen, dass *fast* alle Knoten informiert werden, erweist sich *Push* als äußerst robust. Wir beweisen entsprechende Resultate auch für den Fall, dass unabhängig voneinander jede Nachrichtenübermittlung zwischen zwei Knoten mit bestimmter Wahrscheinlichkeit fehlschlägt.

Drittens bestimmen wir die erwartete Laufzeit von *Pull* und *Push&Pull* wenn sich der zugrunde liegende Graph verändert (für *Push* existiert ein entsprechendes Resultat bereits). Sei $a \in \mathbb{R}^+$ und sei $p := a/n$. Jede Runde betrachten wir eine neue (und unabhängige) Realisierung eines Erdős-Rényi-Zufallsgraphen $G(n,p)$ als zugrunde liegenden Graphen. Wir zeigen, dass die erwartete Laufzeit für *Pull* $\log_{2-e^{-a}}(n) + \ln(n)/a + \mathcal{O}(1)$ ist. Sei ferner $\kappa := 2(1-e^{-a}) - (1-e^{-a})^2/a$. Für *Push&Pull* beweisen wir, dass die erwartete Laufzeit $\log_{1+\kappa}(n) + \ln(n)/a + \mathcal{O}(1)$ ist. Wir zeigen außerdem Konzentration um den Erwartungswert.

Im zweiten Teil dieser Arbeit beschäftigen wir uns mit einem von Alon et al. eingeführten Modell, das die Problematik, wie robust die „Weisheit der Vielen“ ist, behandelt. In diesem Modell gibt es nicht nur eine Information, die sich verbreitet, sondern zwei konkurrierende Informationen (eine wahre und eine falsche), die sich in einem Graphen verbreiten. Es gibt zwei Arten (schwach und stark) von Widersachern (kurz: WS), die bestimmte Möglichkeiten haben, die Verbreitung der falschen Information zu unterstützen. Wir beweisen verschiedene Robustheitsresultate, wobei, grob gesagt, Robustheit bedeutet, dass sich die wahre Information trotz der Einflussnahme des WS durchsetzt. Insbesondere zeigen wir, dass ein geeignet von unten beschränkter Minimalgrad Robustheit gegen den schwachen WS garantiert. Außerdem untersuchen wir für Erdős-Rényi-Zufallsgraphen, für die bereits bekannt ist, dass sie robust gegen den starken WS sind, wie viele Kanten an jedem Knoten gelöscht werden dürfen, ohne dass die Robustheit zerstört werden kann. Ein entsprechendes Resultat beweisen wir auch für den Fall, dass Kanten nicht nur gelöscht, sondern auch hinzugefügt werden dürfen. Schließlich zeigen wir, dass das Problem aus Sicht des starken WS NP-schwer ist.

Im dritten Teil dieser Arbeit betrachten wir ein Vehicle-Routing-Problem mit zwei Arten von Fahrzeugen. Es stehen Lieferwagen und Drohnen zur Verfügung, um Pakete so schnell wie möglich zu bestimmten Positionen zu liefern. Alle Fahrzeuge starten an einem Depot und müssen am Ende ihrer Touren wieder dort ankommen. Lieferwagen können eine beliebige Anzahl an Drohnen und Paketen transportieren. Drohnen können auf Lieferwagen mitfahren, aber auch selbstständig fliegen. Während eine Drohne fliegt, kann sie höchstens ein Paket tragen und, nachdem sie das Paket abgeliefert hat, muss sie zu einem Lieferwagen zurückfliegen um dort zu laden oder direkt zum Depot zurückkehren und dort bleiben. Deswegen kommt es vor, dass ein Lieferwagen auf eine Drohne warten muss oder umgekehrt. Dadurch erhält dieses Vehicle-Routing-Problem einen interessanten Ablaufplanungsaspekt. Wir geben eine formale Definition des Problems an und beweisen eine äquivalente Charakterisierung der Zulässigkeit von Lösungen. Anschließend schlagen wir einen Lokale-Suche-Algorithmus vor und werten ihn empirisch aus.

Acknowledgement

I was supported by many people during my doctoral studies. I am truly grateful to them! I want to express my sincere gratitude to my supervisor Professor Martin Schottenloher, who always had an open door for me, for his steady support, his very valuable suggestions and countless interesting discussions. I am also indebted to Professor Konstantinos Panagiotou; he introduced me to various beautiful topics, first and foremost to the analysis of information spreading algorithms. While working together he shared his outstanding expertise and intuition with me and always provided me with sound advice. Furthermore I want to thank my friends and colleagues Lisa Kraus, Simon Reisser and Leon Ramzews for countless hours of mathematical and non-mathematical discussions and fruitful joint projects. I am grateful to my other friends and colleagues for their support, too; we shared a coffee machine, had innumerable chats and conversations on various topics and motivated each other. I also want to thank Professor Holger Hoos for agreeing to co-examine my doctoral thesis.

Finally, I want to express my deep gratitude towards my parents for their unconditional support throughout my life.

Chapter 1

Introduction

In this thesis we investigate how a piece of information spreads in graphs and prove theoretical bounds for the (random) time until all nodes are informed. Moreover, we consider a setting where not one but two competing pieces of information (one true and one false) spread in graphs. We prove robustness results that guarantee that in graphs with certain properties the true piece of information prevails despite adversarial mechanisms that support the falsehood. As our third subject, we formally define a vehicle routing problem and provide a local search algorithm to solve it. Now, before we give an outline of the theoretical results that we obtain, we take a brief look at relevant practical problems that motivate many of the questions covered in this thesis.

As our first subject, we investigate how fast a piece of information spreads in networks (i.e. graphs). This is crucial for, e.g., maintaining large distributed database systems. When at one part of the system new information is inserted, then the other parts of the distributed system should also obtain the information. However, if every part always forwards any new information to every other part then there will be a huge amount of data traffic. Hence efficient and robust information spreading algorithms are desirable. One possibility is to use simple randomised algorithms. While such algorithms can reduce the data traffic tremendously, there are no deterministic bounds on the time until each part of the system has received the piece of information. However, often it suffices if after a short runtime the probability that every part has obtained the piece of information is sufficiently large.

We consider the three well-known randomised round based information spreading algorithms *Push*, *Pull* and *Push&Pull*. They can be formulated very easily (we will provide definitions later). Nevertheless, even for simple networks, analysing the random variable that counts how many rounds the respective algorithm needs until it has informed every part turns out to be surprisingly challenging; we will call this random variable the runtime of the algorithm; it depends on the underlying graph. To a large extent, the difficulties in the analysis of the algorithms arise due to their round based character where the outcome of one round affects the next rounds significantly. We will study the runtime of information spreading algorithms on different graph classes. Moreover, we will explore how the runtime behaves in the presence of adversarial edge deletions (e.g. deletions of connections between databases).

The motivation for our second subject is based on the fact that nowadays, due to the internet, information and opinions often spread very fast. For example rating based systems

and social networks are omnipresent. In many cases someone tries to manipulate such ratings, for example to cover the low quality of a product; this could be realised by paying people to write positive reviews. Also in social networks, often a piece of information spreads even though it might be incorrect. It is crucial to find out under which conditions the majority of people will believe the truth. Therefore, as our second subject, we explore the behaviour of competing pieces of information; in particular, we assume that there is a true and a false version that both spread within a network. We investigate in which networks the truth prevails in spite of an adversary's efforts who is equipped with certain powers. We also switch perspectives and show that it is NP-hard for the adversary to act optimally.

As our third subject, we investigate a vehicle routing problem where trucks as well as drones can be used to deliver packages to certain positions. The practical motivation is apparent: Logistics companies aim to deliver goods to customers as fast as possible, but within cities there often is a lot of congestion slowing down trucks; also drones that deliver goods are no longer science fiction but technically possible, even though their flight range is still rather restricted. Therefore it seems promising to combine drones and trucks where the drones are used for the so called "last mile delivery". This means the drones are carried by trucks and if a drone is close enough to a customer then it can take a package and deliver it autonomously. This could increase the speed of package delivery. Besides the technical obstacles that have to be overcome (e.g. safety aspects and flight range restrictions) also a mathematical problem arises: As a drone has to return to a truck after having delivered a package, sometimes a truck has to wait for a drone or vice versa. Hence the question of how to deliver packages as fast as possible combines typical vehicle routing components with an interesting scheduling aspect. First we will provide a formalisation of the model and prove an equivalent characterisation of the feasibility of a solution. Then we will introduce a local search algorithm to solve the vehicle routing problem and evaluate it empirically.

After having provided some motivation, now we present theoretical outlines of the three main topics of this thesis corresponding to Chapters 2, 3 and 4 and summarise the results that we obtain. Chapter 2 is devoted to the analysis of the runtime of information spreading algorithms (also called rumour spreading algorithms or gossip algorithms). I worked on this subject together with Simon Reisser. He will publish different but thematically related results in his doctoral thesis which were also obtained during this joint work. Consider a graph G with n nodes and assume that in the beginning one node of G has a piece of information. Now consider the following three randomised algorithms according to which the information spreads. All three algorithms proceed in rounds. In *Push*, in each round each informed node chooses a neighbour independently and uniformly at random (iuar) and, if the chosen neighbour is not informed already, informs it. Similarly, in *Pull*, each round each uninformed node picks a neighbour iuar and if the neighbour is informed, the pulling node is also informed now. *Push&Pull* is a combination of *Push* and *Pull*: Each round each node chooses a neighbour iuar. If at least one of both nodes is informed, afterwards both nodes are informed. For various settings we will investigate the number of rounds that these information spreading algorithms need to inform all nodes, i.e. their runtimes. If the remaining information is clear from the context, then we will refer to the runtime by X_n.

In Section 2.3, we investigate *Push* on the complete graph with n nodes; this setting was subject of various articles since the 1980th (cf. the literature discussion in Section 2.2).

Recently, for the expected runtime, Doerr and Künnemann ([35]) have shown that

$$\lfloor \log_2(n) \rfloor + \ln(n) - 1.116 \leq E[X_n] \leq \lceil \log_2(n) \rceil + \ln(n) + 2.765 + o(1).$$

They describe the random variable X_n itself fairly precisely, too. To state the respective result we need some notation. In the Coupon Collector's Problem (CCP), there are n different types of coupons. Coupons are drawn sequentially and at each draw each type of coupon appears with the same probability. Let $C_n(m)$ denote the (random) number of draws needed to collect the last m coupons and set $C_n = C_n(n)$. We write $Z \sim Geom(p)$ if for all $k \in \mathbb{N} = \{1, 2, \dots\}$ it is $P[Z = k] = (1-p)^{k-1}p$. For two real valued random variables X and Y the expression $Y \preccurlyeq X$ means that X stochastically dominates Y, i.e. for all $x \in \mathbb{R}$ it holds $P[X \geq x] \geq P[Y \geq x]$. Let $\varepsilon > 0$ denote an arbitrarily small positive constant and let $Z \sim Geom(1 + \mathcal{O}(n^{-1+\varepsilon}))$. In [35] it is shown that

$$\lfloor \log_2(n) \rfloor - 1 + \left\lceil \frac{C_n\left(\lceil \frac{n}{2} \rceil\right)}{n} \right\rceil \preccurlyeq X_n \preccurlyeq \lceil \log_2(n) \rceil + 1.562 + \frac{1+\mathcal{O}(n^{-\frac{1}{2}+\varepsilon})}{n} C_n + Z + o(1). \tag{1.0.1}$$

Note that in the literature there are two non-equivalent definitions of geometrically distributed random variables: Either the number of trials until the first success is counted (in this case the random variable takes values in $\{1, 2, \dots\}$) or the number of failed trials is counted (in this case the random variable takes values in $\{0, 1, 2, \dots\}$). In this thesis we use the former variant; in [35] the latter variant is used. Hence, in [35], the constant 1.562 in (1.0.1) is replaced by 2.562. In this work we determine the probability distribution much more accurately, in particular, we prove the following. Let $x(n) := \log_2(n) - \lfloor \log_2(n) \rfloor$ and let $\gamma = 0.57\dots$ denote the Euler-Mascheroni constant. The function c is defined as a certain limit expression (see Notation 2.3.1). Let $d(n) := \ln(n) + \gamma + c(x(n))$ and let $G \sim Gumbel_\gamma$, i.e. for all $x \in \mathbb{R}$ it is $P[G \leq x] = e^{(-e^{-(x+\gamma)})}$. Then there is a function $m : \mathbb{N} \to \mathbb{R}^+$ with $m(n) = o(1)$ (for $n \to \infty$) such that for all $k, n \in \mathbb{N}$

$$|P[X_n \geq k] - P[\lfloor \log_2(n) \rfloor + \lceil G + d(n) \rceil \geq k]| \leq m(n).$$

In Section 2.4, we explore the robustness of *Push*. *Push* is often referred to as a very robust algorithm which usually is meant in the sense that *Push* still informs all nodes fast in spite of independent message transmission failures or node failures (cf. the article by Feige et al. ([43])). We will investigate the robustness of *Push* in a different sense. In particular, we will consider *Push* on graphs with good expansion properties. Panagiotou et al. ([83]) have shown that on such graphs *Push* is as fast as on the complete graph: With high probability (i.e. with probability $1 + o(1)$ for $n \to \infty$) it takes $\log_2(n) + \ln(n) + o(\ln(n))$ rounds until all nodes are informed. We investigate how this runtime changes in the presence of adversarial edge deletions, i.e. before the process starts, an adversary deletes edges of the graph (and may thereby destroy the good expansion properties). Let $\varepsilon > 0$. On the positive side we show that the time that *Push* needs to inform all but $n/\ln(n)$ nodes can be increased by at most $o(\ln(n))$ rounds, even if the adversary may delete up to a $(1/2 - \varepsilon)$ fraction of the edges at each node. On the negative side we show that even if the adversary is only allowed to delete up to an ε fraction of the edges at each node, the time until all nodes are informed may

increase by $\Omega(\ln(n))$ rounds. Let $q \in (0,1]$. We provide respective results in the presence of message transmission failures that occur independently with probability $1-q$ as well.

In Section 2.5, we consider a variant of information spreading where the edges of the underlying graph change over time; these "random evolving graph" settings were introduced by Clementi et al. ([21]). They intend to capture properties of dynamic real-world networks, like e.g. mobile networks, which also change over time. One such setting that is investigated in [21] assumes that the underlying graph is an independently sampled Erdős-Rényi random graph $G(n,p)$ each round. Let $a \in \mathbb{R}^+$. In [34], Doerr and Kostrygin introduced a general framework to analyse information spreading algorithms; using this framework, for $p := a/n$, they determined the expected runtime of *Push* in the described setting up to $\mathcal{O}(1)$ terms and also obtained large deviation bounds. We use this framework to provide respective results for *Pull* and *Push&Pull*. In particular, we show that the expected runtime for *Pull* is $\log_{2-e^{-a}}(n) + \ln(n)/a + \mathcal{O}(1)$. Let $\kappa := 2(1-e^{-a}) - (1-e^{-a})^2/a$. For *Push&Pull* we prove that the expected runtime is $\log_{1+\kappa}(n) + \ln(n)/a + \mathcal{O}(1)$. As a byproduct, we also obtain large deviation bounds. Particularly the result for *Push&Pull* is interesting, as two unusual things happen: The first is that in the beginning pushes and pulls do get in each other's way, i.e. a significant number of nodes is informed by a push as well as by a pull which makes one of the two operations useless. The second is that individually, *Push* and *Pull* both need logarithmic time to inform the last nodes. Hence one might expect that both will contribute substantially to the last phase of the information spreading process; however, it turns out that to inform the last nodes, *Push* is useless. We provide an explanation for these observations in Remark 2.5.15. Note that we have made the contributions of Section 2.5 also available in [23].

While in Chapter 2 we consider the dissemination of one piece of information, in Chapter 3 we investigate how two competing opinions spread. I worked on this subject together with Simon Reisser, too. He will publish different but thematically related results in his doctoral thesis which were also obtained during this joint work. We work in a framework introduced by Alon et al. in [5]: Consider a graph with n nodes. In the beginning, for $0 < \mu < 1/2$, there are μn "expert nodes" that already have an opinion, namely either red or blue. Each non-expert node takes the opinion that the majority of its expert neighbours has, ties (including zero-zero ties) are broken independently and uniformly at random. In [5], the *weak* and the *strong adversary* were introduced; they can affect how the experts are set and thereby they try to promote the blue opinion. The weak adversary is allowed to choose the μn expert nodes; however, while he may choose the nodes, for fixed $0 < \delta < 1/2$, each of the nodes becomes red with probability $1/2 + \delta$ independently of all other nodes and blue otherwise. Like the weak adversary, the strong adversary is allowed to decide which μn nodes have an opinion in the beginning; additionally he is allowed to assign the opinions (red or blue) to the individual nodes as long as he respects the ratio red/blue $= (1/2+\delta)/(1/2-\delta)$. A sequence of graphs $\mathcal{G} = (G_n)_{n\in\mathbb{N}}$ where G_n has n nodes is called robust against a certain kind of adversary if with high probability the majority of the nodes of G_n become red. In [5], it is shown that a suitably upper bounded maximum degree implies robustness against the weak adversary. Let $f : \mathbb{N} \to \mathbb{R}^+$ with $f = \omega(n^{-0.5})$. Complementary to the result from [5], we prove that if the minimum degree is bounded from below by $n(1+\mu-2\delta\mu+f(n))/2$, this assures robustness against the weak adversary. Afterwards we consider Erdős-Rényi random graphs. Let $p = \omega(\ln(n)/n)$. From Theorem 4 in [5] it is known that, with high probability, a sequence of Erdős-Rényi random graphs, $\mathcal{G} = (G_n)_{n\in\mathbb{N}} = (G(n,p(n)))_{n\in\mathbb{N}}$, is robust against

the strong adversary. We investigate the local resilience (cf. the article by Sudakov and Vu ([99])) of $\mathcal{G}$ with respect to robustness against the strong adversary; loosely speaking, this means that we investigate up to which fraction of the edges an adversary has to be allowed to delete at each node of the graphs of the sequence $\mathcal{G}$ to destroy the robustness of $\mathcal{G}$ against the strong adversary. The critical fraction turns out to be $2(1-\mu+2\delta\mu)\delta/(1+2\delta)$. Moreover, we prove a respective result for the case when the adversary is additionally allowed to insert edges (and an insertion counts as much as a deletion, for details see Section 3.3). Finally, we change the perspective and prove that for the strong adversary it is NP-hard to find an optimal strategy.

In Chapter 4, we study a vehicle routing problem that involves drones. I worked on this subject together with Elisabeth Kraus. She will publish different but thematically related results in her doctoral thesis which were also obtained during this joint work; portions of Chapter 4 are part of our article ([24]). The basic model that we consider was (up to rather small differences) introduced by Wang et al. in [107]. Consider two kinds of vehicles, namely trucks and drones, which have to deliver packages to certain destinations. All vehicles start at the depot and at the end of their tours they have to return there. Trucks have an unlimited driving range and can carry an arbitrary number of packages and drones. In contrast, drones can only carry one package at a time and after having delivered the package, they have to recharge on one of the trucks while travelling on it before they can deliver the next package. Now we want to use the trucks and drones to deliver the packages such that the average time a customer has to wait for his delivery is minimised. An interesting aspect of this model is that besides the typical challenges of vehicle routing problems, here also scheduling aspects are critical; in particular, sometimes a truck has to wait for a drone or vice versa. We provide a formal definition of the model (in [107] an intuitive description was given) and prove an equivalent characterisation of the feasibility of a solution. We then develop a local search algorithm and evaluate it empirically.

Chapter 2

Runtime and Robustness of Information Spreading Algorithms

2.1 Introduction

In large networks and distributed databases it is of great importance to spread information efficiently and robustly. We consider the well-known information spreading algorithms (also called rumour spreading algorithms) *Push*, *Pull* and *Push&Pull*. All three algorithms work on graphs and proceed in rounds. In the beginning, one node has a piece of information.[1] In the *Push* algorithm, each round each informed node chooses a neighbour independently and uniformly at random (iuar) and, if it is uninformed, informs it. In the *Pull* algorithm, each round each uninformed node chooses a neighbour iuar. If the neighbour is informed, then the pulling node becomes informed, too. *Push&Pull* combines *Push* and *Pull*; in each round each node chooses a neighbour iuar and if at least one of the two nodes is informed, after this round both are informed. In this chapter, we will prove various results concerning the runtime (i.e. the number of rounds needed to inform all nodes) and the robustness of the described information spreading algorithms. Before we continue, we introduce some notation.

Notation 2.1.1. *For a graph $G = (V, E)$ with $|V| = n$, a node $v \in V$ and $p \in \{Push, Pull, Push\&Pull\}$ let $X(G, v, p)$ denote the (random) number of rounds needed by the information spreading protocol p to inform all n nodes where at the beginning of the first round only v has the piece of information; the random variable $X(G, v, p)$ is also called the runtime of p (on G with start node v). If the choice of v does not matter (this will be always the case in this thesis), we will omit it in the notation; if G or p are clear from the context, we will omit them, too. In these cases, we simply write $X(G, p)$, $X_n(p)$, $X(G)$ or X_n respectively instead of $X(G, v, p)$. In some cases we additionally consider the setting that each message transmission fails independently with probability $1 - q \in [0, 1)$; if q is not clear from the context we will write $X_n^{(q)}$ instead of X_n etc.; we call q the success probability of a message transmission. For $t \in \mathbb{N}$ we define $I_t = I_t^{(q)}(G, v, p) \subseteq V$ as the set of informed nodes at the beginning of round t of the respective information spreading protocol where $I_1 = \{v\}$.*

[1] For general graphs it can make a significant difference which node is informed in the beginning. However, this is not the case for the graph classes that we consider in this thesis, our results hold independently of the choice of the first informed node.

Analogously we set $U_t = U_t^{(q)}(G, v, p) = V \setminus I_t^{(q)}(G, v, p)$ *as the set of uninformed nodes at the beginning of round* t*. For an event* A*, we sometimes write* $P_A[\dots]$ *instead of* $P[\dots \mid A]$ *to denote the conditional probability and we write* $E_A[\dots]$ *instead of* $E[\dots \mid A]$ *for the conditional expectation; if we condition on* I_t*, then we index with* t*, i.e. instead of* $P[\dots \mid I_t]$ *we write* $P_t[\dots]$ *and instead of* $E[\dots \mid I_t]$ *we write* $E_t[\dots]$*. Mostly we will consider sequences of graphs* $\mathcal{G} = (G_n)_{n\in\mathbb{N}}$ *where* G_n *has* n *nodes. This induces a (random) sequence of runtimes* $(X_n)_{n\in\mathbb{N}}$*; we will study the asymptotic behaviour for* $n \to \infty$*. Thus, if not stated differently, asymptotic notation is with respect to* $n \to \infty$*; in particular, "whp" (which is short for "with high probability") means with probability* $1 + o(1)$ *for* $n \to \infty$*. We denote the natural logarithm by* $\ln$*. For a random variable* X *that takes values in* $\mathbb{N} = \{1, 2, \dots\}$*, we say that* X *follows a geometric distribution with parameter* $p \in (0, 1)$ *and write* $X \sim Geom(p)$ *if for any* $k \in \mathbb{N}$ *it is* $P[X = k] = (1 - p)^{k-1}p$*. We denote the complete graph on* n *nodes by* K_n*. Again consider a graph* $G = (V, E)$*; for a node* $v \in V$ *we write* $N(v)$ *or* $N_G(v)$ *to denote its neighbourhood and* $d(v) = d_G(v) := |N_G(v)|$ *to denote its degree. For subsets* $U, W \subseteq V$ *with* $U \cap W = \emptyset$ *we write* $E(U, W) = E_G(U, W) \subseteq E$ *to denote the set of edges that have one endpoint in* U *and one endpoint in* W*; we set* $e(U, W) = e_G(U, W) := |E_G(U, W)|$*. Moreover, we write* $N(U) = N_G(U) := \bigcup_{u\in U} N_G(u)$*. Notation specific to the individual subsections will be introduced there. For simplicity of exposition we will ignore rounding issues that do not affect the results.*

In Section 2.2, we provide a literature overview of work related to various aspects of information spreading. In Section 2.3, we determine the probability distribution of the runtime of *Push* on the complete graph with n nodes very accurately. Afterwards, in Section 2.4, we investigate the robustness of the runtime of *Push* against adversarial edge deletions. Then, in Section 2.5, we consider *Pull* and *Push&Pull* in a setting where the underlying graph changes each round; for *Push* respective results already exist, cf. the paragraph "Evolving Graphs" in Subsection 2.2.2.

2.2 Background and Related Literature

In this section, we summarise several important lines of research in the field of information spreading.

2.2.1 Fundamental Articles Establishing the Field of Information Spreading

In [48], Frieze and Grimmett investigate *Push* on the complete graph with n nodes; they show that, whp,

$$X(K_n, Push) = \log_2(n) + \ln(n) + o(\ln(n)).$$

Moreover, it is shown that for any $\gamma, \varepsilon > 0$

$$P[X(K_n, Push) > (1 + \varepsilon)(1 + (1 + \gamma)\ln(2))\log_2(n)] = o(n^{-\gamma}).$$

In [87], Pittel has improved the results from [48], in particular, he has shown that for any $f : \mathbb{N} \to \mathbb{R}^+$ with $f = \omega(1)$, whp,

$$|X(K_n, Push) - \log_2(n) - \ln(n)| \leq f(n).$$

Demers et al. ([30]) have considered information spreading in the context of replicating databases. *Push*, *Pull* and *Push&Pull* approaches were defined and investigated. They observed that it "is possible to replace complex deterministic algorithms for replicated database consistency with simple randomised algorithms that require few guarantees from the underlying communication system". They provide empirical simulation results.

In [43], Feige et al. have investigated *Push*. For a connected graph G with n nodes, similarly to the (random) runtime $X(G, v, Push)$, they consider the (deterministic) guaranteed runtime $T(G, v, Push)$ which is defined as the minimum number of rounds after which all nodes are informed with probability at least $1 - 1/n$. As for the runtime, also for the guaranteed runtime we suppress the node that is informed in the beginning in the notation if its choice does not affect the result. Let G_n be a connected graph with n nodes. They show

$$\log_2(n) \leq T(G_n, Push) \leq 12n \log_2(n).$$

The lower bound is trivial because using *Push*, the number of informed nodes can at most double each round. Furthermore, they prove that these bounds are tight up to constant factors. They also provide bounds for the guaranteed runtime in terms of the maximum degree and the diameter. In particular, let $\mathcal{G} = (G_n)_{n\in\mathbb{N}}$ be a sequence of graphs where G_n has n nodes, maximum degree Δ_n and diameter $diam_n$, then

$$T(G_n, Push) = \mathcal{O}(\Delta_n(diam_n + \ln(n))).$$

Furthermore, the following resilience result is shown. Let $c < 1/3$ and let $(G_n)_{n\in\mathbb{N}}$ denote a sequence of graphs where G_n has n nodes and such that in G_n less than $\lfloor cn \rfloor$ edges are missing (compared to the complete graph). Then

$$T(G_n, Push) = \mathcal{O}(\ln(n)).$$

Moreover, they also consider *Push* on hypercubes and on random graphs; for details see [43].

In [68], Karp et al. consider the so called random phone call model; *Push&Pull* can be regarded as a special case of the random phone call model. They show that, whp,

$$X(K_n, Push\&Pull) = \log_3(n) + \mathcal{O}(\ln(\ln(n))).$$

Besides the runtime, they also consider the number of message transmissions that are needed. Therefore, they investigate certain stopping criteria because they do not assume that the protocol stops automatically if every node has received the information and hence such stopping criteria are necessary to avoid unnecessary message transmissions; this reflects the idea, that nodes have only local information. In order to avoid such dispensable message transmissions, they introduce robust termination schemes; for details see [68].

2.2.2 Information Spreading on Certain Graph Classes

The Complete Graph Information spreading on the complete graph is the most basic case. Nevertheless, its analysis contains several difficulties. We summarise the currently best known bounds for the runtimes of *Push*, *Pull* and *Push&Pull*. We start with *Push*. Doerr and Künnemann ([35]) have shown that

$$\lfloor \log_2(n) \rfloor + \ln(n) - 1.116 \leq E[X_n(Push)] \leq \lceil \log_2(n) \rceil + \ln(n) + 2.765 + o(1). \tag{2.2.1}$$

Also large deviation bounds were obtained. Besides this precise result, an interesting proof idea was developed in [35]: Assume that $n_1 \in \mathbb{N}$ nodes are informed. A target number $n_2 \in \mathbb{N}$ with $n_2 > n_1$ is fixed such that with probability $1-q^*$ close to 1 at least n_2 nodes are informed after the next round; if this fails, simply retry. Hence the number of rounds needed until at least n_2 nodes are informed (starting with n_1 informed nodes) can be bounded from above by a geometrically distributed random variable with parameter $1-q^*$. Thereby it is avoided to consider the distribution of the number of informed nodes at certain times; instead one essentially has to deal with sums of geometrically distributed random variables with high success probabilities. This idea and its applicability is substantially generalised by Doerr and Kostrygin ([34, 70]). They introduce a general framework to analyse information spreading algorithms. We will apply their framework in this thesis in Section 2.5. In a large class of situations, it allows to obtain bounds for the runtime of information spreading algorithms up to constant additive terms. In particular, this is the case for *Pull* and *Push&Pull* (and *Push*, but there also the bound (2.2.1) from [35] is known) on the complete graph; it yields (see [34])

$$\begin{aligned} E[X_n(Pull)] &= \log_2(n) + \log_2(\ln(n)) + \mathcal{O}(1), \\ E[X_n(Push\&Pull)] &= \log_3(n) + \log_2(\ln(n)) + \mathcal{O}(1). \end{aligned}$$

Now additionally consider the situation that each message transmission fails independently with probability $1-q \in (0,1)$. Then (see [34])

$$\begin{aligned} E[X_n^{(q)}(Push)] &= \log_{1+q}(n) + \frac{1}{q}\ln(n) + \mathcal{O}(1), \\ E[X_n^{(q)}(Pull)] &= \log_{1+q}(n) - \frac{1}{\ln(1-q)}\ln(n) + \mathcal{O}(1) \text{ and} \\ E[X_n^{(q)}(Push\&Pull)] &= \log_{1+2q}(n) + \frac{1}{q-\ln(1-q)}\ln(n) + \mathcal{O}(1). \end{aligned}$$

It is noteworthy that besides the expected values, the framework from [34] also yields large deviation bounds of the form

$$P[|X_n - E[X_n]| \geq r] \leq A\exp(-\alpha r) \tag{2.2.2}$$

where $A, \alpha > 0$ are suitable constants.

Graphs with Evenly Distributed Edges There are several results that, loosely speaking, state the following. If the edges of a graph are distributed rather uniformly, then information spreading is as fast as on the complete graph, i.e. density is not a crucial factor. We will present some of these results.

In [45], Fountoulakis et al. investigate the runtime of *Push* on Erdős-Rényi random graphs: Let $q \in (0,1]$, let $p = p(n) = \omega(\ln(n)/n)$ and let $\varepsilon > 0$. Let $\mathcal{G} = (G_n)_{n\in\mathbb{N}}$ be a sequence of Erdős-Rényi random graphs, $G_n = G(n,p)$. Then, whp,

$$X^{(q)}(G_n, \mathit{Push}) = (1 \pm \varepsilon)\left(\log_{1+q}(n) + \frac{1}{q}\ln(n)\right).$$

In fact, a stronger result is shown, for details see [45].

In [46], Fountoulakis and Panagiotou study information spreading on random regular graphs and on expander graphs. A random d-regular graph $G(n,d)$ with n nodes is obtained by sampling uniformly at random from the set of all d-regular graphs with n nodes; for $d \geq 3$, whp, (for those n for which a d-regular graph with n nodes exists) this yields a connected graph (see [12]). Let $d \geq 3$, recall that a d-regular graph on $n \in \mathbb{N}$ nodes exists if and only if $n \geq d+1$ and dn is even. Therefore define $M = M(d) = \{n \in \mathbb{N} \mid n \geq d+1 \wedge dn \text{ is even}\}$. Let us write $M = \{m_1, m_2, \dots\}$ such that for $i, j \in \mathbb{N}$ with $i < j$ it is $m_i < m_j$. Let $\mathcal{G} = (G_n)_{n\in\mathbb{N}}$ be a sequence of d-regular graphs where G_n has m_n nodes. Let

$$C_d = \frac{1}{\ln(2(1-1/d))} - \frac{1}{d\ln(1-1/d)}.$$

Then, whp,

$$|X(G_n, \mathit{Push}) - C_d \cdot \ln(m_n)| = \mathcal{O}\left((\ln(\ln(m_n)))^2\right).$$

It is easy to see that for $d \to \infty$ we have $C_d \to 1/\ln(2) + 1$; recall that the runtime of *Push* on the complete graph on n nodes, whp, is $(1/\ln(2) + 1)\ln(n) + \mathcal{O}(1)$. This confirms the insensitivity of *Push* with respect to density.

In [83], Panagiotou et al. prove results concerning *Push* on graphs with good expansion properties. Intuitively, such graphs are characterised by the fact that their edges are distributed rather uniformly. Loosely speaking, it is shown that on graphs with good expansion properties *Push* is as fast as on the complete graph. More specifically, let $\mathcal{G} = (G_n)_{n\in\mathbb{N}}$ denote a sequence of graphs that have good expansion properties where G_n has n nodes (for a formal statement see [83, Theorem 1.1] or Theorem 2.4.6). Then, whp,

$$|X(G_n, \mathit{Push}) - (\log_2(n) + \ln(n))| = o(\ln(n)).$$

This underlines that density is not a crucial factor.

Power Law Graphs Real world communication networks and social networks often essentially follow a power-law distribution (cf., e.g., the article by Adamic et al. ([3])).

In [47], Fountoulakis et al. investigate *Push&Pull* on random graphs with a power law degree distribution with exponent $\beta > 2$; besides the usual round-based variant of *Push&Pull* also an asynchronous variant is studied; for a description of this asynchronous variant see the paragraph "Asynchronous *Push&Pull*" in Subsection 2.2.4. In [47], the Chung-Lu model

([20]) is used to construct power law graphs. Roughly speaking, it is shown that if $2 < \beta < 3$, then *Push&Pull* informs almost all nodes in $\Theta(\ln(\ln(n)))$ rounds whp, whereas for $\beta > 3$ it takes $\Omega(\ln(n))$ rounds; the asynchronous version of *Push&Pull* (where $\beta \in (2,3)$) informs almost all nodes in constant time. This is particularly remarkable since, whp, the distance of two nodes selected uniformly at random is at least $2\ln(\ln(n))/|\ln(\beta-2)|$ (see the article by Dereich et al. ([31])).

In [33], Doerr et al. consider *Push&Pull* and a modified version of *Push&Pull* on preferential attachment graphs (which are power law graphs, cf. the article of Barabási and Albert ([7])). In the modified version of *Push&Pull* they assume that nodes do not choose the same communication partner in two consecutive rounds. They show that then the runtime is sublogarithmic, in particular they show that, whp, the runtime is $\mathcal{O}(\ln(n)/\ln(\ln(n)))$. They also prove that the usual *Push&Pull* protocol, whp, needs $\Theta(\ln(n))$ rounds to inform all nodes in a preferential attachment graph, thereby improving the previously best known bound of $\mathcal{O}(\ln^2(n))$ rounds by Chierichetti et al. ([19]); in [19], also *Push* and *Pull* were investigated on preferential attachment graphs. It is shown that, independently of the start node, with constant positive probability *Push* needs a polynomial number of rounds and there are start nodes such that with constant positive probability *Pull* needs a polynomial number of rounds. This shows that the combination of *Push* and *Pull* in *Push&Pull* is crucial for the fast runtime.

Evolving Graphs Clementi et al. ([21]) have considered information spreading algorithms where the underlying graph changes over time. They investigate two models. In the first model, in each round, the underlying graph is a newly (and independently of all previous events) sampled Erdős-Rényi random graph $G(n,p)$. In the second model, they start with an initial graph G_1 and then, for all $t \in \mathbb{N}$ with $t \geq 2$ the (random) graph G_{t+1} for round $t+1$ is obtained from G_t as follows: For $\bar{p}, \bar{q} \in (0,1)$, each edge that is present in G_t is not present in G_{t+1} with probability $\bar{q}$; similarly, each edge that is not present in G_t is present in G_{t+1} with probability $\bar{p}$. The latter setting is called the edge-Markovian setting. In the former setting, they show that for every p the runtime of *Push* is, whp, $\mathcal{O}(\ln(n)/(\hat{p}n))$ where $\hat{p} = \min\{p, 1/n\}$. This has been improved in [34] for $p = a/n$ where $a > 0$ is a constant; in particular, using the framework that they introduced, they show that the expected runtime is $\log_{2-e^{-a}}(n) + \ln(n)/(1-e^{-a}) + \mathcal{O}(1)$ and a large deviation bound in the form of (2.2.2) is obtained as well. In Section 2.5, we will prove respective results for *Pull* and *Push&Pull*. In the edge-Markovian setting, in [21], they prove that for $1/n \leq \bar{p} < 1$ and $\bar{q} = \Omega(1)$ *Push*, whp, needs $\Theta(\ln(n))$ rounds.

2.2.3 Results Depending on Graph Parameters

In Subsection 2.2.2 we have seen that one important line of research aims to obtain precise runtime bounds for specific graph classes. In this subsection, we consider a different line of research: We consider results that quantify the runtime of information spreading algorithms in terms of graph parameters of the underlying graph. Before we state some respective results, we need two definitions of such graph parameters, namely the conductance and the vertex expansion.

Definition 2.2.1 (Conductance, cf., e.g., [50]). *Let $G = (V, E)$ be a connected graph. For $S \subseteq V$ set $vol(S) := \sum_{v \in S} d(v)$. The conductance ϕ of G is defined as*

$$\phi := \min_{S \subseteq V, vol(S) \leq |E|} \leq \frac{e(S, S^c)}{vol(S)}.$$

Definition 2.2.2 (Vertex expansion, cf., e.g., [51]). *Let $G = (V, E)$ be a connected graph and set $n := |V|$. For $S \subseteq V$ let $\partial S := N(S) \backslash S$ denote the boundary of S. The vertex expansion of S is defined as $\alpha(S) := |\partial S|/|S|$ and the vertex expansion of the graph G is defined as $\alpha(G) := \min\{\alpha(S) \mid S \subseteq V, 0 < |S| \leq n/2\}$.*

The relation between the speed of information spreading and graph conductance was studied by Mosk-Aoyama and Shah ([81]) and by Chierichetti et al. ([17, 18]). The results from those articles have been improved by Giakkoupis ([50]). There the following bound in terms of the conductance ϕ is proven. For any connected graph with n nodes and for any start node, whp, *Push&Pull* needs $\mathcal{O}(\phi^{-1} \ln(n))$ rounds to inform all nodes which is tight for $\phi = \Omega(1/n)$ (for $\varepsilon > 0$ and $\phi \geq 1/n^{1-\varepsilon}$, a matching lower bound was already derived in [17]).

The relation between information spreading and vertex expansion has been recognised as an interesting problem by Chierichetti et al. ([18]). It has been studied by Sauerwald and Stauffer ([95]) and Giakkoupis and Sauerwald ([53]). In [51], Giakkoupis has improved the results; in particular, he has shown that for a graph $G = (V, E)$ with $|V| = n$, maximum degree at most Δ and vertex expansion at least α, for any constant $\beta > 0$, with probability $1 + \mathcal{O}(n^{-\beta})$, *Push&Pull* informs all nodes within $\mathcal{O}(\ln(n) \ln(\Delta)/\alpha)$ rounds. This result matches a lower bound from [53].

These results about the conductance and the vertex expansion underline that good expansion properties constitute a crucial factor for the speed of information spreading whereas density is not important.

2.2.4 Further Variants of Information Spreading

Besides the classical *Push*, *Pull* and *Push&Pull* protocols, also several modified versions have been studied.

Only One Call Can Be Answered In [25], Daum et al. consider the following two restricted variants of *Pull*. In both, each node can only answer at most one pull request per round. In *random* RPULL in each round for each node that obtained more than one pull request, the answered pull request is chosen independently and uniformly at random. In *adversarial* RPULL an adaptive adversary chooses the pull request that is answered. On the one hand, it is proven that on trees both variants perform essentially equally fast. On the other hand, it is shown that there are graphs where the random version only needs a polylogarithmic number of rounds while the adversarial version needs $\Omega(\sqrt{n})$ rounds. Moreover, a relation between *Pull* and random RPULL is derived: Let Δ and δ denote the maximum and the minimum degree of the underlying graph respectively; it is shown that if *Pull* can inform all nodes within T rounds with probability p, then, with probability $(1 + o(1))p$, random RPULL informs all nodes in $\mathcal{O}(T\Delta/\delta \ln(n))$ rounds. Furthermore, an analogously restricted

variant of *Push&Pull* (i.e. *Push* canonically combined with random RPULL) is introduced and it is noted that $\mathcal{O}(\Delta/\delta \ln(n))$ rounds in the restricted setting stochastically dominate a single round of the classical *Push&Pull* algorithm; for details see [25].

Multiple Calls per Round In [84], Panagiotou et al. investigate a variant of information spreading where nodes can make multiple calls (call refers to both, push and pull attempts). The number of calls that a node makes per round is determined in the beginning of the information spreading process for each node independently as a sample of a random variable R. Besides other results, roughly speaking, the following is shown for *Push&Pull*: If R follows a power law distribution with exponent $2 < \beta < 3$ then, whp, the runtime of *Push&Pull* is $\Theta(\ln \ln(n))$; if $\beta = 3$, then, whp, it is $\Theta(\ln(n)/\ln(\ln(n)))$. Furthermore, the respective variant of *Push* with multiple calls is considered: It is shown that if $E[R] = \mathcal{O}(1)$ and $Var[R] = \mathcal{O}(1)$, then, whp, the runtime is $\log_{1+E[R]}(n) + \ln(n)/E[R] + o(\ln(n))$. Also a lower bound is shown; in particular, assume that $E[R] = \mathcal{O}(1)$, then, whp, the runtime of *Push* is $\Omega(\ln(n))$.

In [34], Doerr and Kostrygin investigate a different variant with multiple calls. Consider a random variable R on $\mathbb{N}_0$. In each round each node has a sample (which is independent of the other samples) of R assigned that specifies how many different neighbours the node calls in this round. So the main difference is that here the number of calls a node can make is newly sampled in each round. They prove that if $E[R] = \Theta(1)$ and $Var[R] = \mathcal{O}(1)$, then the expected runtime of *Push* is $\log_{1+E[R]}(n) + \ln(n)/E[R] + \mathcal{O}(1)$. Now let l be the smallest value that R takes with constant positive probability. They prove that if $l = 0$ then the expected runtime of *Push&Pull* is $\log_{1+2E[R]}(n) + \ln(n)/(E[R] - \ln(P[R = 0])) + \mathcal{O}(1)$. If $l \geq 1$, then the expected runtime of *Push&Pull* is $\log_{1+2E[R]}(n) + \log_{1+l}(\ln(n)) + \mathcal{O}(1)$.

Asynchronous *Push&Pull* Besides the classical round based variants of information spreading protocols, also asynchronous variants are of interest. Such asynchronicity was first considered by Boyd et al. in [14] (in the slightly different context of the so called averaging problem, for details see [14]). The asynchronous variant of *Push&Pull* is defined as follows. Each node has a rate 1 Poisson clock associated, all clocks work independently of each other. Whenever the clock of a node rings, then it chooses a neighbour independently and uniformly at random and if at least one of the two nodes is informed, afterwards both are informed. To denote the runtime in the asynchronous setting (i.e. the time until the asynchronous variant of *Push&Pull* has informed all nodes), we write $\tilde{X}(G, v, p)$.

In [85], Panagiotou and Speidel study asynchronous information spreading on Erdős-Rényi random graphs. We summarise some of the results that are obtained for the expected runtime; respective bounds that hold whp are also proven, for details see [85]. Let $\mathcal{G} = (G_n)_{n\in\mathbb{N}}$ denote a sequence of graphs where G_n is an Erdős-Rényi random graph, $G_n = G(n, p)$. Let $\alpha(n) = \omega(1)$. They show that for $p = p(n) = \alpha(n) \ln(n)/n$ whp (where the remaining randomness stems from the sampling of the random graph)

$$E[\tilde{X}(G_n, \textit{Push\&Pull})] = \left(1 \pm \sqrt{34/\alpha(n)}\right) H_{n-1} + \mathcal{O}\left(\frac{\ln(n)}{n}\right)$$

where H_n is the n-th harmonic number, i.e. $H_n := \sum_{1\leq j\leq n} j^{-1}$. Note that $H_n = \ln(n) + \gamma + \mathcal{O}(1/n)$ where $\gamma = 0.57\ldots$ denotes the Euler-Mascheroni constant. Also the case where p is

closer to the connectivity threshold $\ln(n)/n$ is considered; it is shown that for any constant $c > 1$ for $p = c\ln(n)/n$ whp (as above, the remaining randomness stems from the sampling of the random graph)

$$E[\tilde{X}(G_n, \mathit{Push\&Pull})] \leq 1.5H_{n-1} + \mathcal{O}(\ln^{3/4}(n)).$$

This is particularly remarkable as for this range of p the runtime of the synchronous variant of *Push&Pull* cannot be bounded by $C\ln(n)$ for a constant C that is independent of p (see [83]). Additionally, several robustness results are proven.

In [2], Acan et al. compare the asynchronous variant and the synchronous variant of *Push&Pull*. Recall that the guaranteed runtime of an information spreading algorithm on a graph with n nodes is defined as the minimum number $T \in \mathbb{N}$ such that with probability at least $1 - 1/n$ after T rounds all nodes are informed. In [2], the following is shown. For all connected graphs with n nodes, the guaranteed runtime in the asynchronous version is at most by a factor of $\mathcal{O}(\ln(n))$ larger than in the synchronous variant. However, the asynchronous variant may be much faster than the synchronous variant: There are graphs where the asynchronous variant has logarithmic guaranteed runtime while the guaranteed runtime of the synchronous version is polynomial. On the positive side they show that the ratio of the guaranteed runtime in the synchronous setting to the guaranteed runtime in the asynchronous setting is always $\mathcal{O}(n^{2/3})$. In [52], Giakkoupis et al. have improved this to $\mathcal{O}(n^{1/2})$ and Angel et al. ([6]) have further improved this to $\tilde{\mathcal{O}}(n^{1/3})$ ($\tilde{\mathcal{O}}$ in contrast to $\mathcal{O}$ ignores logarithmic factors). According to an example provided in [2], this is tight.

2.3 The Distribution of the Runtime of *Push* on the Complete Graph

In Section 2.2, we have seen that since the 1980th there have been various articles investigating the runtime of *Push* on the complete graph with n nodes. In spite of these efforts, it was still not understood very precisely. This section is devoted to determine this runtime, i.e. to analyse the probability distribution of the random variable that counts how many rounds *Push* needs to inform all nodes on the complete graph with n nodes. While *Push* can be formulated so easily, its analysis includes several difficulties. These obstacles mainly arise by the division into separated rounds where the previous rounds have an essential impact on the next rounds. We will overcome these obstacles by a careful analysis in three phases.

Notation 2.3.1. *Define*

$$f : [0,1] \to [0,1], \quad x \mapsto 1 - e^{-x}(1-x) \qquad \text{and} \qquad g : [0,1] \to [0,1], \quad x \mapsto xe^{x-1}.$$

Note that for $x \in [0,1]$ it holds $f(x) = 1 - g(1-x)$ and hence also $g(x) = 1 - f(1-x)$. Set $f_1 := f$ and for any $i \in \mathbb{N}$ recursively define $f_{i+1} := f \circ f_i$ and analogously set $g_1 := g$ and $g_{i+1} := g \circ g_i$. A simple inductive argument yields that for $x \in [0,1]$ and $i \in \mathbb{N}$ it holds $f_i(x) = 1 - g_i(1-x)$ as well as $g_i(x) = 1 - f_i(1-x)$. We consider Push on the complete graph with n nodes. We are interested in large values of n, thus recall that, unless otherwise

specified, asymptotic notation is with respect to $n \to \infty$ and that in particular "whp" (which is short for "with high probability") means with probability $1+o(1)$ for $n \to \infty$. As before, let $X_n := \min\{t-1 \mid t \in \mathbb{N}, |I_t| = n\}$ denote the random variable that counts how many rounds Push needs to inform all nodes of the complete graph with n nodes, i.e. the runtime of Push on the complete graph. Define

$$\begin{aligned} x : \mathbb{N} &\to [0,1) \\ n &\mapsto \log_2(n) - \lfloor \log_2(n) \rfloor \end{aligned}$$

and define $c : \mathbb{R} \to \mathbb{R}$ by

$$c(x) = \lim_{a \to \infty, a \in \mathbb{N}} \lim_{b \to \infty, b \in \mathbb{N}} -a + b + \ln(g_b(1 - 2^{-a-x})).$$

Lemma 2.3.18 assures that c is well-defined. Let $\gamma = 0.57\ldots$ denote the Euler-Mascheroni constant and define

$$\begin{aligned} d : \mathbb{N} &\to \mathbb{R} \\ n &\mapsto \ln(n) + \gamma + c(x(n)). \end{aligned}$$

We say a real valued random variable G follows a $Gumbel_\gamma$ distribution, $G \sim Gumbel_\gamma$, if for all $x \in \mathbb{R}$

$$P[G \leq x] = e^{(-e^{-(x+\gamma)})}.$$

For all $n \in \mathbb{N}$, let Y_n and Z_n denote real valued random variables; we write

$$Y_n \stackrel{\delta}{=} Z_n, \quad Y_n \stackrel{\delta}{\geq} Z_n \quad \text{or} \quad Y_n \stackrel{\delta}{\leq} Z_n$$

respectively if there is a function $h : \mathbb{N} \to \mathbb{R}^+$ with $h = o(1)$ such that for all $n \in \mathbb{N}$ and all $x \in \mathbb{R}$

$$|P[Y_n \geq x] - P[Z_n \geq x]| \leq h(n)$$

or

$$P[Y_n \geq x] - P[Z_n \geq x] \geq -h(n) \quad \text{or} \quad P[Y_n \geq x] - P[Z_n \geq x] \leq h(n)$$

respectively.

The quantity $f(x)n$ is approximately (for large values of n) the number of informed nodes after a round of *Push* that started with xn informed nodes. We will see why this is the case in the proofs of Lemmas 2.3.15 and 2.3.27. Similarly, $g(x)n$ is approximately the number of uninformed nodes after a round of *Push* that started with xn uninformed nodes. Assume that after round t, xn nodes are uninformed; the fact that the number of uninformed nodes after round $t+1$ is concentrated around $g(x)n$ has already been observed in the early stage of research on information spreading, cf. [87, Lemma 2]. As a tool to prove our main result, we will prove Lemma 2.3.15 which assures that during the entire information spreading process with very high probability the numbers of informed and uninformed nodes follow very closely deterministic recursions as described above.

2.3.1 Main Result

Our main result is to determine the probability distribution of X_n very precisely.

Theorem 2.3.2. *Let $G \sim Gumbel_\gamma$. There is a function $m : \mathbb{N} \to \mathbb{R}^+$ with $m = o(1)$, such that for all $k, n \in \mathbb{N}$*

$$|P[X_n \geq k] - P[\lfloor \log_2(n) \rfloor + \lceil G + d(n) \rceil \geq k]| \leq m(n).$$

2.3.2 Preliminaries

In this subsection we collect some preliminaries that we will use on our way to prove Theorem 2.3.2. Fact 2.3.3 describes the asymptotic behaviour of a certain expression and provides a (non-asymptotic) bound for it.

Fact 2.3.3. *Let $(a_n)_{n\in\mathbb{N}}$ be a real-valued sequence with $a_n = \mathcal{O}(1)$. Then*

$$(1 + a_n/n)^n = e^{a_n} + \mathcal{O}(a_n^2/n).$$

Moreover, for any $n \in \mathbb{N}$ and any $a \in \mathbb{R}$ with $|a| \leq n$

$$\left(1 + \frac{a}{n}\right)^n \leq e^a.$$

We will use the Chernoff bounds to bound $|I_{t+1}|$ given $|I_t|$. In order to see that they are applicable, we consider self-bounding functions (see Definition 2.3.4 and Lemmas 2.3.5 and 2.3.6); this approach to bound $|I_{t+1}|$ was already used in [93]. Note that Definition 2.3.4 and Lemma 2.3.5 can be stated for much more general settings; however, the provided versions suffice for our purposes.

Definition 2.3.4 (Self-bounding function, [13, 76])**.** *A non-negative function $h : \mathbb{N}^n \to \mathbb{R}$ is self-bounding if there exist functions $h_i : \mathbb{N}^{n-1} \to \mathbb{R}$ such that for all $x_1, ..., x_n \in \mathbb{N}$ and all $i = 1, ..., n$,*

$$0 \leq h(x_1, ..., x_n) - h_i(x_1, ..., x_{i-1}, x_{i+1}, ..., x_n) \leq 1$$

and also

$$\sum_{i=1}^{n} h(x_1, ..., x_n) - h_i(x_1, ..., x_{i-1}, x_{i+1}, ..., x_n) \leq h(x_1, ..., x_n).$$

Lemma 2.3.5 (Exponential inequalities for self-bounding functions, [13])**.** *Let $n \in \mathbb{N}$ and $h : \mathbb{N}^n \to \mathbb{R}$ be a self-bounding function. Let $Y_1, \ldots, Y_n$ denote independent random variables that take values in $\mathbb{N}$. Let $Y = h(Y_1, \ldots, Y_n)$. Then for $s \geq 0$ and $0 < t < E[Y]$*

$$P[Y \geq E[Y] + s] \leq \exp\left(-\frac{s^2}{2E[Y] + 2s/3}\right) \quad \text{and}$$
$$P[Y \leq E[Y] - t] \leq \exp\left(-\frac{t^2}{2E[Y]}\right).$$

Lemma 2.3.6 ([93]). *There is an $m \in \mathbb{N}$ such that, conditioned on I_t, there are independent random variables $Y_1, ..., Y_m$ that take values in $\mathbb{N}$ and a self-bounding function $h : \mathbb{N}^m \to \mathbb{R}$ such that*

$$|I_{t+1}| = h(Y_1, ..., Y_m).$$

In particular, Lemma 2.3.5 is applicable.

Lemma 2.3.8 will be helpful to assure that the number of informed nodes essentially doubles each round in the beginning of the information spreading process. To formulate it, we need Definition 2.3.7. Note that in [35], in contrast to the notation we use, geometrically distributed random variables take values in $\mathbb{N}_0$ and count the number of failures until the first success; also there they start with I_0 as the set of informed nodes in the beginning of the process, where $|I_0| = 1$. If we state results of [35], then we adapt them to our notation.

Definition 2.3.7 (Stochastic dominance). *Let X and Y be real valued random variables. We say X stochastically dominates Y and write $Y \preceq X$ if for any $x \in \mathbb{R}$ it holds $P[X \geq x] \geq P[Y \geq x]$.*

Lemma 2.3.8 ([35, Lemma 3.1]). *Let $n_1 < \sqrt{n}$ be a power of two, $t_1 := \min\{t \mid t \in \mathbb{N}, |I_t| \geq n_1\}$ and $X \sim Geom(1 - n_1^2/n)$. Then t_1 is stochastically dominated by $\log_2(n_1) + X$.*

Theorem 2.3.9 provides a sufficient condition for uniform convergence of a sequence of functions.

Theorem 2.3.9 ([90, Nr. 127 on p. 81, proof on p. 270]). *Let $a, b \in \mathbb{R}$ with $a < b$ and let $(h_n)_{n\in\mathbb{N}}$ denote a sequence of functions $h_n : [a, b] \to \mathbb{R}$ with the following properties. For all $n \in \mathbb{N}$, the function h_n is monotonously increasing and the sequence converges pointwise to a continuous function $h : [a, b] \to \mathbb{R}$. Then the convergence is uniform, i.e.*

$$\lim_{n\to\infty} \sup_{x\in[a,b]} |h_n(x) - h(x)| = 0.$$

We will need the following well-known notions of convergence for sequences of random variables given in Definitions 2.3.10 and 2.3.12.

Definition 2.3.10 (Convergence in distribution). *For each $n \in \mathbb{N}$ let X_n denote a real-valued random variable with distribution function F_n and let X be a real-valued random variable with distribution function F. We say X_n converges in distribution to X and write $X_n \xrightarrow{d} X$ if for every $x \in \mathbb{R}$ where F is continuous it holds $\lim_{n\to\infty} F_n(x) = F(x)$.*

Theorem 2.3.11 assures that if $X_n \xrightarrow{d} X$ and the limit distribution F is continuous everywhere, then the convergence of $F_n(x)$ to $F(x)$ is uniform in x.

Theorem 2.3.11 (Polya's Theorem, [89, Theorem 1]). *For each $n \in \mathbb{N}$ let X_n be a real-valued random variable with distribution function F_n and assume that X_n converges in distribution to the random variable X; let F denote the distribution function of X and assume that F is continuous on $\mathbb{R}$. Then*

$$\lim_{n\to\infty} \sup_{x\in\mathbb{R}} |F_n(x) - F(x)| = 0.$$

Definition 2.3.12 (Convergence in probability)**.** *For each $n \in \mathbb{N}$ let X_n be a real valued random variable and suppose that there is a real valued random variable X such that for all $\varepsilon > 0$*

$$\lim_{n\to\infty} P[|X_n - X| > \varepsilon] = 0.$$

Then we say that X_n converges in probability to X and write $X_n \xrightarrow{p} X$.

Later, in Lemma 2.3.22, we will state a certain convergence result. In order to prove it, we will use Theorems 2.3.13 and 2.3.14.

Theorem 2.3.13 (Slutsky's Theorem, see, e.g., [97, p. 19])**.** *Let $(X_n)_{n\in\mathbb{N}}$, $(Y_n)_{n\in\mathbb{N}}$ and $(Z_n)_{n\in\mathbb{N}}$ be sequences of real-valued random variables. Suppose that X_n converges in distribution to a real valued random variable X and that there are constants $a, b \in \mathbb{R}$ such that $Y_n \xrightarrow{p} a$ and $Z_n \xrightarrow{p} b$. Then $Y_n X_n + Z_n \xrightarrow{d} aX + b$.*

Theorem 2.3.14 ([42])**.** *For $n \in \mathbb{N}$, let $\tilde{X}_0, \tilde{X}_1, \ldots, \tilde{X}_{n-1}$ be independent random variables where, for $i \in \{0, 1, \ldots, n-1\}$, it is $\tilde{X}_i = \tilde{X}_i^{(n)} \sim Geom((n-i)/n)$. Set $\tilde{D}_i = \tilde{D}_i^{(n)} := \tilde{X}_i - E[\tilde{X}_i] = \tilde{X}_i - n/(n-i)$ and let $G \sim Gumbel_\gamma$. Then*

$$\sum_{i=0}^{n-1} \frac{\tilde{D}_i}{n} \xrightarrow{d} G.$$

2.3.3 Preparation Results

In this subsection, we state several results that we will use in the proof of the main result; we prove these results in Subsection 2.3.4. We start with Lemma 2.3.15 that states that the (random) sequence $(|I_t|)_{t\in\mathbb{N}}$ can be described very precisely by a deterministic recursion.

Lemma 2.3.15. *Let $\delta \in (0, 1/2)$ and $t_0 = t_0(\delta, n) = (1/2 - \delta)\lfloor \log_2(n) \rfloor$. For $t \in \mathbb{N}_0$ set*

$$\alpha_{t+1} = \alpha_{t+1}(t_0, n) = \begin{cases} 2^t/n & \text{if } t < t_0 \\ f(\alpha_t) & \text{if } t \geq t_0 \end{cases}.$$

Let $\sqrt{2} < D < \sqrt{2.5}$, let $\varepsilon > 0$ and let $t_1 \in \mathbb{N}$ with $t_1 > t_0$. For all $t \in \mathbb{N}$ define $A_t = A_t(t_1) = A_t(t_1, t_0)$ as the event that

$$||I_t| - \alpha_t n| \leq \begin{cases} 0 & \text{if } t \leq t_0 \\ (\alpha_t n)^{1/2+\varepsilon} D^{t-t_0} & \text{if } t_0 \leq t \leq t_1 \\ (\alpha_t n)^{1/2+\varepsilon} \max\{1, D^{t_1 - t_0 - 2(t-t_1)}\} & \text{if } t_1 \leq t \end{cases}.$$

Then there is a constant $K^* \in \mathbb{N}$ such that for $t_1 := \lfloor \log_2(n) \rfloor + K^*$

$$P\left[\bigcap_{t\in\mathbb{N}} A_t(t_1)\right] = 1 + \mathcal{O}(n^{-2\delta}).$$

Moreover, there are $n_0 \in \mathbb{N}$, $\varepsilon, \delta > 0$ (e.g. $\varepsilon = \delta = 0.01$) and $\sqrt{2} < D < \sqrt{2.5}$ such that for all $n \geq n_0$ and $t \in \mathbb{N}$, A_t implies $||I_t| - \alpha_t n| \leq n^{0.8}$.

While Lemma 2.3.15 assures that the number of informed nodes follows closely a deterministic recursion, Lemma 2.3.16 provides a bound for the number of informed nodes if we go *backwards* along that recursion.

Lemma 2.3.16. *Consider the (random) sequence of informed nodes $(I_t)_{t\in\mathbb{N}}$ given by the Push process on the complete graph. Let $c = 1 - 1/e$. For $t \in \mathbb{N}$ let B_t denote the event that $|I_{t-1}| \geq e(|I_t| - cn) - 9n^{0.8}$. Then*

$$P\left[\bigcap_{t\in\mathbb{N}} B_t\right] = 1 + \mathcal{O}(n^{-0.02}).$$

Lemma 2.3.17 assures that in the beginning of the information spreading process the number of informed nodes almost doubles each round.

Lemma 2.3.17. *Let $0 < \varepsilon < 0.1$ with $\log_2(\varepsilon) \in \mathbb{Z}$. Let $t^* = \lfloor \log_2(n) \rfloor + \log_2(\varepsilon)$. Then there is an $\tilde{\varepsilon} > 0$ such that with probability $1 + \mathcal{O}(n^{-\tilde{\varepsilon}})$*

$$|2^{t^*-1} - |I_{t^*}|| = 2^{t^*-1} - |I_{t^*}| \leq \varepsilon 2^{t^*-1}.$$

Moreover, let $n \in \mathbb{N}$ and $\delta \in (0, 1/2)$ and let $t_0 = t_0(\delta, n)$ and $(\alpha_t)_{t\in\mathbb{N}} = (\alpha_t(t_0, n))_{t\in\mathbb{N}}$ be defined as in Lemma 2.3.15. Then

$$|2^{t^*-1} - \alpha_{t^*} n| = 2^{t^*-1} - \alpha_{t^*} n \leq \varepsilon 2^{t^*-2}.$$

Lemma 2.3.18 verifies that the function c defined in Notation 2.3.1 and used in Theorem 2.3.2 indeed is well-defined; Lemma 2.3.19 states that c is continuous on the interval $[0, 1]$ and Corollary 2.3.21 assures that, again on the interval $[0, 1]$, the convergence in Lemma 2.3.18 *b*) is uniform with respect to x.

Lemma 2.3.18. *a) For any $a \in \mathbb{N}$ and any $x \in \mathbb{R}$ with $x > -a$, the limit*

$$\eta(a, x) := \lim_{b\to\infty, b\in\mathbb{N}} b + \ln(g_b(1 - 2^{-a-x}))$$

exists. For $x \in [0, 1]$ the convergence is uniform with respect to x, i.e. for any $a \in \mathbb{N}$

$$\lim_{b\to\infty, b\in\mathbb{N}} \sup_{x\in[0,1]} \left|\eta(a, x) - (b + \ln(g_b(1 - 2^{-a-x})))\right| = 0.$$

b) For any $x \in \mathbb{R}$ the limit

$$\lim_{a\to\infty, a\in\mathbb{N}} -a + \eta(a, x) \tag{2.3.1}$$

exists. In particular, the function $c : \mathbb{R} \to \mathbb{R}$ given by

$$c(x) = \lim_{a\to\infty, a\in\mathbb{N}} \lim_{b\to\infty, b\in\mathbb{N}} -a + b + \ln(g_b(1 - 2^{-a-x}))$$

is well-defined.

Lemma 2.3.19. *The function c from Lemma 2.3.18 b) is continuous on the interval $[0,1]$.*

Remark 2.3.20. *Since every continuous function on a compact interval is uniformly continuous, we also could infer uniform continuity in Lemma 2.3.19; in fact, in our proof of Lemma 2.3.19 we show uniform continuity directly, as it does not complicate the proof.*

Corollary 2.3.21. *For $x \in [0,1]$ the convergence in (2.3.1) is uniform with respect to x, i.e.*

$$\lim_{a\to\infty, a\in\mathbb{N}} \sup_{x\in[0,1]} |c(x) - (-a + \eta(a,x))| = 0.$$

Lemma 2.3.22 provides a convergence result for a sequence of random variables. It is not stated in the most general form that is possible but rather formulated to fit our purposes.

Lemma 2.3.22. *Let $\varepsilon : \mathbb{N} \to (0, 1/3)$ with $\varepsilon = \omega(1/n)$ and such that for all $n \in \mathbb{N}$ it is $\varepsilon(n)n \in \mathbb{N}$. For all $n \in \mathbb{N}$, let $s_n : \mathbb{N} \to \mathbb{R}^+$ denote a monotonously increasing function that for all $m \in \mathbb{N}$ fulfils $s_n(m) \leq m$ and that also has the following property. For any function $\delta : \mathbb{N} \to (0, 1/3)$ such that for all $n \in \mathbb{N}$ it is $\delta(n)n \in \mathbb{N}$ and $\delta(n) \leq \varepsilon(n)$, it holds $s_n((1-\delta(n))n) \geq (1 - e\delta(n) + \mathcal{O}(n^{-0.2}))n$. For $n \in \mathbb{N}$, let $X_1, X_2 \ldots, X_{n-1}$ be independent random variables where, for $i \in \{1, 2, \ldots, n-1\}$, $X_i = X_i^{(n)} \sim Geom((n-i)/(n-1))$. Set $D_i = D_i^{(n)} := X_i - n/(n-i)$ and let $G \sim Gumbel_\gamma$. Then*

$$\sum_{i=(1-\varepsilon(n))n}^{n-1} \frac{D_i}{s_n(i)} \xrightarrow{d} G.$$

Corollary 2.3.23 is a consequence of Lemma 2.3.22 and Theorem 2.3.11.

Corollary 2.3.23. *Under the assumptions of Lemma 2.3.22 the following holds. There is a function $m : \mathbb{N} \to \mathbb{R}^+$ with $m = o(1)$ such that for all $l : \mathbb{N} \to \mathbb{R}$ and $k : \mathbb{N} \to \mathbb{N}$, for all $n \in \mathbb{N}$*

$$\left| P\left[\left\lceil l(n) + \sum_{i=(1-\varepsilon(n))n}^{n-1} \frac{D_i}{s_n(i)} \right\rceil \geq k(n)\right] - P[\lceil l(n) + G \rceil \geq k(n)] \right| \leq m(n).$$

We use Lemma 2.3.24 to obtain Corollary 2.3.25 which will allow us to deduce an upper bound from a lower bound for the runtime of *Push*.

Lemma 2.3.24. *There is an $a_0 \in \mathbb{R}^+$ such that for all $a \in \mathbb{R}^+$ with $a \geq a_0$ there is a $b^* = b^*(a) \in \mathbb{N}$ such that for all $b \in \mathbb{N}$ with $b \geq b^*$, for all $\eta \in [0,1]$ the following holds. Let $L_1^{(\eta)} := (1 - 2^{-a})2^{-a-\eta}$ and $U_1^{(\eta)} := 2^{-a-\eta}$. Then*

$$f_b(U_1^{(\eta)}) - f_b(L_1^{(\eta)}) \leq 2^{-\frac{3}{2}a} \quad \text{and} \quad 1 - f_b(L_1^{(\eta)}) \leq 2^{-a} \quad \text{and} \quad \frac{1 - f_b(L_1^{(\eta)})}{1 - f_b(U_1^{(\eta)})} \leq 1 + 2^{-a/2+4}.$$

Corollary 2.3.25. *There is an $a_0 \in \mathbb{R}^+$ such that for all $a \in \mathbb{R}^+$ with $a \geq a_0$ the following holds. Let $L_1^{(\eta)} = L_1^{(\eta)}(a)$, $U_1^{(\eta)} = U_1^{(\eta)}(a)$ and $b^* = b^*(a)$ be defined as in Lemma 2.3.24. Then for all $b \in \mathbb{N}$ with $b \geq b^*$, for all $\eta \in [0,1]$*

$$\ln\left(\frac{1}{f_b(L_1^{(\eta)})} - 1\right) - \ln\left(\frac{1}{f_b(U_1^{(\eta)})} - 1\right) \leq 2^{-a/2+5} \overset{a\to\infty}{\to} 0.$$

2.3.4 Proofs of the Preparation Results

To prove the preparation results we will make use of Lemmas 2.3.26, 2.3.27 and 2.3.28. Lemma 2.3.26 is a concentration result for the number of informed nodes.

Lemma 2.3.26. *Let $0 < c \leq 1$, let $t_0 \in \mathbb{N}$ and assume that $|I_{t_0}| = \Omega(n^c)$. For $t \in \mathbb{N}$ and $\varepsilon > 0$ let $C_t = C_t(\varepsilon)$ denote the event that*

$$||I_{t+1}| - E_t[|I_{t+1}|]| \leq (E_t[|I_{t+1}|])^{1/2+\varepsilon}.$$

Then there is a $\delta = \delta(\varepsilon, c) > 0$ such that for any $t \geq t_0$

$$P_t[C_t] = 1 + \mathcal{O}(\exp(-n^\delta)).$$

Moreover, also the following stronger result holds: There is a $\tilde{\delta} = \tilde{\delta}(\varepsilon, c) > 0$ such that

$$P_{t_0}\left[\bigcap_{t \geq t_0} C_t\right] = 1 + \mathcal{O}(\exp(-n^{\tilde{\delta}})). \tag{2.3.2}$$

Proof. To keep the notation short we write $d := E_t[|I_{t+1}|]^{1/2+\varepsilon}$. We use Lemma 2.3.5 which is applicable according to Lemma 2.3.6 and obtain

$$\begin{aligned} P_t[\neg C_t] &= P_t[||I_{t+1}| - E_t[|I_{t+1}|]| > d] \leq 2\exp\left(-\frac{d^2}{2E_t[|I_{t+1}|] + \frac{2}{3}d}\right) = \exp\left(-\frac{d^2}{\Theta(|I_t|)}\right) \\ &= \exp(-\Theta(|I_t|^{2\varepsilon})) = \exp(-\Omega(n^{2c\varepsilon})) = \mathcal{O}(\exp(-n^{2c\varepsilon})). \end{aligned}$$

Hence for $\delta := 2c\varepsilon$ the first claim follows. From [35, Corollary 3.2] it is known that for any $r > 0$

$$P[X_n \geq \lceil \log_2(n) \rceil + \ln(n) + 2.188 + r] \leq 2e^{-r}.$$

Thus it suffices to consider

$$P_{t_0}\left[\bigcup_{t_0 \leq t \leq n} \neg C_t\right]$$

and therefore the second claim follows by the union bound. □

Lemma 2.3.27 provides an asymptotic expression for the expected number of informed nodes after one additional round.

Lemma 2.3.27. *It is*

$$E_t[|I_{t+1}|] = |I_t| + (n - |I_t|)(1 - e^{-|I_t|/n}) + \mathcal{O}(1) = n - (n - |I_t|)e^{-|I_t|/n} + \mathcal{O}(1) = f(|I_t|/n)n + \mathcal{O}(1).$$

Proof. It is straightforward to see that for $|I_t| = \mathcal{O}(1)$ the claim is true. Hence we assume $|I_t| = \omega(1)$. At the beginning of round t, there are $n - |I_t|$ uninformed nodes. Each uninformed node u remains uninformed in round t if all $|I_t|$ informed nodes do not push to u. Thus, by

considering the complementary event, we obtain that the probability that u gets informed in round t is $1-(1-1/(n-1))^{|I_t|}$. Hence, by linearity of expectation, we have

$$E_t[|I_{t+1}|] = |I_t|+(n-|I_t|)\left(1-\left(1-\frac{1}{n-1}\right)^{|I_t|}\right) = |I_t|+(n-|I_t|)\left(1-\left(1-\frac{\frac{|I_t|}{n-1}}{|I_t|}\right)^{|I_t|}\right)$$

and therefore, using Fact 2.3.3,

$$E_t[|I_{t+1}|] = |I_t|+(n-|I_t|)(1-e^{-|I_t|/n})+\mathcal{O}(1) = n-(n-|I_t|)e^{-|I_t|/n}+\mathcal{O}(1) = f(|I_t|/n)n+\mathcal{O}(1).$$

□

Lemma 2.3.28 is an auxiliary result that we use in the proofs of Lemmas 2.3.15 and 2.3.16.

Lemma 2.3.28. *Let $0 < x_1 \leq x_2 < 1$. Then $|f(x_1) - f(x_2)| \leq (2-x_1)e^{-x_1}(x_2-x_1)$.*

Proof. It is $f'(x) = (2-x)e^{-x}$; in particular, f' is monotonically decreasing and takes only positive values on $[x_1, x_2]$. It is

$$\max_{x\in[x_1,x_2]} f'(x) = (2-x_1)e^{-x_1}$$

and therefore, as a direct consequence of the mean value theorem, we have

$$|f(x_1) - f(x_2)| \leq (x_2-x_1)\max_{x\in[x_1,x_2]} f'(x) = (2-x_1)e^{-x_1}(x_2-x_1).$$

□

Proof of Lemma 2.3.15. We start with the case that $t < t_0$. We prove that with probability at least $1-n^{-2\delta}$ it is $|I_{\lfloor t_0\rfloor}| = 2^{\lfloor t_0\rfloor-1}$. As the number of informed nodes at most doubles per round, this implies that, with probability at least $1-n^{-2\delta}$, A_t holds for all $t < t_0$. From Lemma 2.3.8, we obtain

$$P[|I_{\lfloor t_0\rfloor}| < 2^{\lfloor t_0\rfloor-1}] \leq 1-\left(1-\frac{\left(2^{\lfloor t_0\rfloor}\right)^2}{n}\right) = \frac{2^{2\lfloor t_0\rfloor}}{n} \leq \frac{n^{1-2\delta}}{n} = n^{-2\delta}.$$

We continue with the case that $t \geq t_0$. According to Lemma 2.3.26 there is a $\tilde{\delta}$ such that with probability $1+\mathcal{O}(\exp(-n^{\tilde{\delta}}))$ for all $t \geq t_0$

$$||I_{t+1}|-E_t[|I_{t+1}|]| \leq (E_t[|I_{t+1}|])^{1/2+\varepsilon/2}. \tag{2.3.3}$$

As $\mathcal{O}(n^{-2\delta})+\mathcal{O}(\exp(-n^{\tilde{\delta}})) = \mathcal{O}(n^{-2\delta})$ from now on we can assume that (2.3.3) holds. We write

$$d := (E_t[|I_{t+1}|])^{1/2+\varepsilon/2}.$$

Let $t_0 \leq t < t_1$. We prove the claim by induction with respect to t. Our induction hypothesis is

$$||I_t|-\alpha_t n| \leq (\alpha_t n)^{1/2+\varepsilon}D^{t-t_0} =: d_1 \tag{2.3.4}$$

or equivalently

$$||I_t|/n - \alpha_t| \leq \alpha_t^{1/2+\varepsilon} D^{t-t_0} n^{-1/2+\varepsilon}.$$

Note that $d = o(d_1)$. The base case is clear. Hence, assuming (2.3.4) and (2.3.3), we have to show

$$||I_{t+1}| - \alpha_{t+1} n| \leq (\alpha_{t+1} n)^{1/2+\varepsilon} D^{t+1-t_0}.$$

Using (2.3.3) and Lemma 2.3.27 we obtain

$$|I_{t+1}| = f(|I_t|/n)n + \mathcal{O}(d).$$

Using (2.3.4) and Lemma 2.3.28 we can approximate $f(|I_t|/n)$ by $f(\alpha_t)$ thereby obtaining

$$\begin{aligned} |I_{t+1}| &= f(|I_t|/n)n + \mathcal{O}(d) = f(\alpha_t)n \pm d_1(2 - (\alpha_t - d_1/n))e^{-\alpha_t + d_1/n} + \mathcal{O}(d) \\ &= \alpha_{t+1} n \pm d_1(2-\alpha_t)e^{-\alpha_t} + \mathcal{O}(d_1^2/n) + \mathcal{O}(d) = \alpha_{t+1} n \pm d_1(2-\alpha_t)e^{-\alpha_t} + o(d_1). \end{aligned}$$

Thus

$$||I_{t+1}| - \alpha_{t+1} n| \leq d_1(2-\alpha_t)e^{-\alpha_t} + o(d_1). \tag{2.3.5}$$

We verify that

$$\alpha_t^{1/2+\varepsilon}(2-\alpha_t)e^{-\alpha_t} \leq \sqrt{2}\alpha_{t+1}^{1/2+\varepsilon}.$$

This is equivalent to prove

$$\left(\frac{\alpha_{t+1}}{\alpha_t}\right)^{1/2+\varepsilon} \geq \frac{(2-\alpha_t)e^{-\alpha_t}}{\sqrt{2}}.$$

As $\alpha_{t+1}/\alpha_t \geq 1$ it suffices to show the stronger claim

$$\left(\frac{\alpha_{t+1}}{\alpha_t}\right)^{1/2} \geq \frac{(2-\alpha_t)e^{-\alpha_t}}{\sqrt{2}}.$$

We have

$$\begin{aligned} \left(\frac{\alpha_{t+1}}{\alpha_t}\right)^{1/2} \geq \frac{(2-\alpha_t)e^{-\alpha_t}}{\sqrt{2}} &\iff \frac{\alpha_{t+1}}{\alpha_t} \geq \left(\frac{(2-\alpha_t)e^{-\alpha_t}}{\sqrt{2}}\right)^2 \iff 2\alpha_{t+1} \geq \alpha_t(2-\alpha_t)^2 e^{-2\alpha_t} \\ &\iff 2(1 - e^{-\alpha_t}(1-\alpha_t)) \geq \alpha_t(2-\alpha_t)^2 e^{-2\alpha_t} \\ &\iff -\alpha_t e^{-2\alpha_t}(\alpha_t - 2)^2 - 2(1-\alpha_t)e^{-\alpha_t} + 2 \geq 0. \end{aligned}$$

Using that for all $x \in \mathbb{R}$ it holds $e^x \geq x + 1$ we continue with

$$\begin{aligned} -\alpha_t e^{-2\alpha_t}(\alpha_t - 2)^2 - 2(1-\alpha_t)e^{-\alpha_t} + 2 &\geq -\alpha_t e^{-\alpha_t}(\alpha_t - 2)^2 - 2(1-\alpha_t)e^{-\alpha_t} + 2 \\ &= e^{-\alpha_t}\left(-\alpha_t(\alpha_t-2)^2 - 2(1-\alpha_t) + 2e^{\alpha_t}\right) \\ &\geq e^{-\alpha_t}\left(-\alpha_t(\alpha_t-2)^2 - 2(1-\alpha_t) + 2(\alpha_t+1)\right) \\ &= e^{-\alpha_t}(4-\alpha_t)\alpha_t^2 \geq 0 \end{aligned}$$

and thus indeed

$$\alpha_t^{1/2+\varepsilon}(2-\alpha_t)e^{-\alpha_t} \leq \sqrt{2}\alpha_{t+1}^{1/2+\varepsilon}.$$

Hence we obtain

$$d_1(2-\alpha_t)e^{-\alpha_t} \leq D^{t-t_0}\sqrt{2}(\alpha_{t+1}n)^{1/2+\varepsilon}.$$

Therefore, as $D > \sqrt{2}$, recalling (2.3.5), for n sufficiently large we arrive at

$$||I_{t+1}|-\alpha_{t+1}n| \leq (\alpha_{t+1}n)^{1/2+\varepsilon}D^{t+1-t_0}.$$

Now we continue with the case that $t \geq t_1$. Again we use induction with respect to t. This time our induction hypothesis is

$$||I_t|-\alpha_t n| \leq (\alpha_t n)^{1/2+\varepsilon}\max\{1, D^{t_1-t_0-2(t-t_1)}\} =: d_2. \tag{2.3.6}$$

Note that, as $\max\{1, D^{t_1-t_0-2(t-t_1)}\} \geq 1$, it holds $d = o(d_2)$. The base case is clear. Recall that we can assume that (2.3.3) holds. Assuming (2.3.6), we have to show

$$||I_{t+1}|-\alpha_{t+1}n| \leq (\alpha_{t+1}n)^{1/2+\varepsilon}\max\{1, D^{t_1-t_0-2(t+1-t_1)}\}.$$

Analogously to the calculation that yielded (2.3.5), here we arrive at

$$|I_{t+1}| = f(|I_t|/n)n + \mathcal{O}(d) = \alpha_{t+1}n \pm d_2(2-\alpha_t)e^{-\alpha_t} + o(d_2) \tag{2.3.7}$$

and thus

$$||I_{t+1}|-\alpha_{t+1}n| \leq d_2(2-\alpha_t)e^{-\alpha_t} + o(d_2). \tag{2.3.8}$$

In order to complete the proof we need to bound $d_2(2-\alpha_t)e^{-\alpha_t}$; to do this, we will verify that there is a $K^* \in \mathbb{N}$ such that for all $t \in \mathbb{N}$ with $t \geq t_1 = \lfloor \log_2(n) \rfloor + K^*$

$$(2-\alpha_t)e^{-\alpha_t} \leq 0.4. \tag{2.3.9}$$

Note that for any constant $x \in (0,1)$ it holds $f_t(x) \overset{t\to\infty}{\to} 1$; thus, as $(2-1)e^{-1} = 1/e < 0.4$, to prove the existence of such a K^*, it suffices to prove that $\alpha_{\lfloor \log_2(n) \rfloor}$ can be bounded from below by a positive constant. In order to do this, by considering the series representation of the exponential function, we obtain that for $x \in [0,1]$

$$\begin{aligned} f(x) &= 1 - e^{-x}(1-x) \geq 1 - \left(1 - x + \frac{x^2}{2}\right)(1-x) = x\left(2 - \frac{3}{2}x + \frac{1}{2}x^2\right) \\ &\geq x\left(2 - \frac{3}{2}x\right) \geq 2x - 2x^2 =: \tilde{f}(x). \end{aligned} \tag{2.3.10}$$

Set $\tilde{f}_1 := \tilde{f}$ and for $i \in \mathbb{N}$ set $\tilde{f}_{i+1} := \tilde{f} \circ \tilde{f}_i$. We will derive an explicit form for $\tilde{f}_i$. To do this, consider the following recursively defined sequence $(x_i)_{i\in\mathbb{N}}$. Let $x_1 \in (0,1)$ and, for $i \in \mathbb{N}$, set $x_{i+1} := \tilde{f}(x_i)$. We claim that

$$x_i = \frac{1}{2}\left(1 - (1-2x_1)^{(2^{i-1})}\right).$$

Let us verify this claim: For $i \in \mathbb{N}$ we have $x_{i+1} = 2x_i - 2x_i^2$ which is equivalent to

$$1 - 2x_{i+1} = 1 - 4x_i + 4x_i^2 = (1-2x_i)^2.$$

For $i \in \mathbb{N}$ set $y_i := 1 - 2x_i$. We can reformulate the recurrence relation in terms of y_i and obtain $y_{i+1} = y_i^2$. Hence $y_i = y_1^{(2^{i-1})}$ and thus

$$x_i = \frac{1}{2}(1 - y_i) = \frac{1}{2}\left(1 - y_1^{(2^{i-1})}\right) = \frac{1}{2}\left(1 - (1-2x_1)^{(2^{i-1})}\right).$$

Therefore, for $i \in \mathbb{N}$ and $x \in (0,1)$, we have

$$\tilde{f}_i(x) = \frac{1}{2}(1 - (1-2x)^{(2^i)}) \tag{2.3.11}$$

and hence

$$\alpha_i \geq f_{i-1}(1/n) \geq \tilde{f}_{i-1}(1/n) = \frac{1}{2}\left(1 - \left(1 - \frac{2}{n}\right)^{(2^{i-1})}\right). \tag{2.3.12}$$

In particular, for $i = \lfloor \log_2(n) \rfloor$,

$$\begin{aligned}\alpha_{\lfloor \log_2(n) \rfloor} &\geq \tilde{f}_{\lfloor \log_2(n) \rfloor - 1}(1/n) = \frac{1}{2}\left(1 - \left(1 - \frac{2}{n}\right)^{(2^{\lfloor \log_2(n) \rfloor - 1})}\right) \\ &\geq \frac{1}{2}\left(1 - \left(1 - \frac{2}{n}\right)^{(2^{\log_2(n)-2})}\right) = \frac{1}{2}\left(1 - \left(1 - \frac{2}{n}\right)^{n/4}\right)\end{aligned}$$

and thus, using Fact 2.3.3, we get

$$\alpha_{\lfloor \log_2(n) \rfloor} \geq \frac{1}{2}(1 - e^{-1/2}).$$

Therefore we can infer that there is a $K^* \in \mathbb{N}$ such that (2.3.9) holds for all $t \in \mathbb{N}$ with $t \geq t_1 = \lfloor \log_2(n) \rfloor + K^*$. Thus

$$d_2(2 - \alpha_t)e^{-\alpha_t} \leq (\alpha_t n)^{1/2+\varepsilon} \max\{1, D^{t_1 - t_0 - 2(t-t_0)}\} \cdot 0.4.$$

Hence, as $D^2 \cdot 0.4 < 2.5 \cdot 0.4 = 1$, considering (2.3.8), for n sufficiently large we can infer

$$||I_{t+1}| - \alpha_{t+1} n| \leq (\alpha_{t+1} n)^{1/2+\varepsilon} \max\{1, D^{t_1 - t_0 - 2(t+1-t_1)}\}.$$

To prove the second claim, we have to show that there are $\varepsilon, \delta > 0$ and $\sqrt{2} < D < \sqrt{2.5}$ such that for all $t \in \mathbb{N}$, A_t implies $||I_t| - \alpha_t n| \leq n^{0.8}$. Let $\delta = \varepsilon = 0.01$ and let $D = 2^{1/2+\varepsilon}$ and let $K^* \in \mathbb{N}$ such that for all $t \in \mathbb{N}$ with $t \geq t_1 = \lfloor \log_2(n) \rfloor + K^*$ inequality (2.3.9) holds. Note that

$$D^{t_1 - t_0} \leq (2^{(1/2+\delta)\log_2(n) + K^*})^{1/2+\varepsilon} = (n^{1/2+\delta} 2^{K^*})^{1/2+\varepsilon} = n^{1/4 + 1/2(\varepsilon+\delta) + \varepsilon\delta} 2^{K^*(1/2+\varepsilon)}.$$

Hence, for sufficiently large n, we have

$$D^{t_1 - t_0} \leq n^{1/4 + 1/2(\varepsilon+\delta) + \varepsilon\delta} 2^{K^*(1/2+\varepsilon)} < n^{0.27}$$

and thus, assuming that A_t holds,

$$||I_t| - \alpha_t n| \leq (\alpha_t n)^{1/2+\varepsilon} D^{t_1 - t_0} \leq (\alpha_t n)^{1/2+\varepsilon} n^{0.27} \leq n^{0.8}.$$

□

Proof of Lemma 2.3.16. We use the notation introduced in Lemma 2.3.15. Let $\delta = 0.01$. From Lemma 2.3.15, we obtain that with probability $1 + \mathcal{O}(n^{-2\delta})$ for all $t \in \mathbb{N}$

$$|I_t| \leq \alpha_t n + n^{0.8} = f(\alpha_{t-1})n + n^{0.8}.$$

Thus, using Lemma 2.3.28, we continue with

$$\begin{aligned}|I_t| &\leq f(\alpha_{t-1})n + n^{0.8} \leq (f(|I_{t-1}|/n) + 2n^{-0.2})n + n^{0.8} = f(|I_{t-1}|/n)n + 3n^{0.8} \\ &= n - e^{-|I_{t-1}|/n}(n - |I_{t-1}|) + 3n^{0.8} \leq n - \frac{1}{e}(n - |I_{t-1}|) + 3n^{0.8} = cn + |I_{t-1}|/e + 3n^{0.8}.\end{aligned}$$

Hence we arrive at $|I_t| \leq cn + |I_{t-1}|/e + 3n^{0.8}$. Solving this for $|I_{t-1}|$ yields

$$|I_{t-1}| \geq e(|I_t| - cn - 3n^{0.8}) \geq e(|I_t| - cn) - 9n^{0.8}.$$

□

Proof of Lemma 2.3.17. We start with the proof of the second claim. From (2.3.12) we know that

$$\alpha_{t^*} \geq \frac{1}{2}\left(1 - \left(1 - \frac{2}{n}\right)^{(2^{t^*-1})}\right) = \frac{1}{2}\left(1 - \left(1 - \frac{2^{t^*}/n}{2^{t^*-1}}\right)^{(2^{t^*-1})}\right).$$

Thus, using Fact 2.3.3, we obtain

$$\alpha_{t^*} \geq \frac{1}{2}(1 - e^{-2^{t^*}/n}).$$

Hence, considering the series representation of the exponential function, we can infer

$$\alpha_{t^*} \geq \frac{1}{2}\left(1 - \left(1 - \frac{2^{t^*}}{n} + \frac{(2^{t^*}/n)^2}{2}\right)\right) = \frac{2^{t^*-1}}{n} - \left(\frac{2^{t^*-1}}{n}\right)^2$$

and therefore

$$2^{t^*-1} - \alpha_{t^*} n \leq \frac{2^{t^*-1}}{n} 2^{t^*-1} \leq \varepsilon 2^{t^*-2}.$$

We continue with the proof of the first claim. In order to do this we use Lemma 2.3.15 which gives that with probability $1 + \mathcal{O}(n^{-0.02})$ it holds $||I_{t^*}| - \alpha_{t^*} n| \leq n^{0.8}$ and therefore

$$2^{t^*-1} - I_{t^*} \leq 2^{t^*-1} - \alpha_{t^*} n + n^{0.8} \leq \varepsilon 2^{t^*-2} + n^{0.8} = (\varepsilon + \mathcal{O}(n^{-0.2}))2^{t^*-2}.$$

□

In Definition 2.3.29, we quantify "exponentially fast convergence" and in Lemma 2.3.30 we state related properties.

Definition 2.3.29 (Exponentially fast convergence)**.** *Let $(a_n)_{n\in\mathbb{N}}$ be a real-valued sequence and let $c \in (0,1)$. If there is an $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$ we have $|a_{n+1}| < c|a_n|$ then we say that a_n converges exponentially fast to zero at rate c with start number n_0.*

Lemma 2.3.30. *a) Let $c \in (0,1)$ and let $(a_n)_{n\in\mathbb{N}}$ be a real-valued sequence that converges exponentially fast to zero at rate c. Then $\sum_{n\geq 1} a_n$ converges absolutely.*
b) Let $c \in (0,1)$, $n_0 \in \mathbb{N}$ and let $(h_n)_{n\in\mathbb{N}}$ denote a sequence of functions with $h_n : [0,1] \to \mathbb{R}$ such that for any $x \in [0,1]$ the sequence $(h_n(x))_{n\in\mathbb{N}}$ converges exponentially fast to zero at rate c with start number at most n_0 (note that c and n_0 do not depend on x). Define $h : [0,1] \to \mathbb{R}$ by $h(x) := \sum_{n\geq 1} h_n(x)$ (according to a) this is well-defined). Then the sequence of functions $(h_n)_{n\in\mathbb{N}}$ converges uniformly to h, i.e.

$$\lim_{n\to\infty} \sup_{x\in[0,1]} |h_n(x) - h(x)| = 0.$$

Proof. We start with a). Let $n_0 \in \mathbb{N}$ be such that for all $n \geq n_0$ it is $|a_{n+1}| < c|a_n|$. Then

$$\sum_{n=1}^{\infty} |a_n| = \sum_{n=1}^{n_0} |a_n| + \sum_{n=n_0+1}^{\infty} |a_n| \leq \sum_{n=1}^{n_0} |a_n| + \sum_{n=1}^{\infty} c^n |a_{n_0}| = \sum_{n=1}^{n_0} |a_n| + |a_{n_0}| \frac{c}{1-c}.$$

Now we prove b). Let $\varepsilon > 0$. We show that there is an $n_1 \in \mathbb{N}$ such that for all $n \geq n_1$ and for all $x \in [0,1]$ it holds $|h_n(x) - h(x)| < \varepsilon$. For $n \geq n_0$ it is

$$|h_n(x) - h(x)| = \left| \sum_{j=n+1}^{\infty} h_n(x) \right| \leq \sum_{j=n+1}^{\infty} |h_n(x)| \leq |a_{n_0}| \sum_{j=n+1}^{\infty} c^j = |a_{n_0}| \frac{c^{n+1}}{1-c}$$

which implies that an n_1 as required exists. □

Lemma 2.3.31 is a slight generalisation of the monotone convergence theorem; we use it in the proof of Lemma 2.3.18.

Lemma 2.3.31. *Let $(a_n)_{n\in\mathbb{N}}$ be a bounded real-valued sequence. Assume that there is a real-valued sequence $(b_n)_{n\in\mathbb{N}}$ that converges exponentially fast to zero in the sense of Definition 2.3.29 such that for all $n \in \mathbb{N}$ it holds $a_{n+1} - a_n \geq b_{n+1}$. Then $(a_n)_{n\in\mathbb{N}}$ converges.*

Proof. Let $\varepsilon > 0$. We show that there is an $n_0 \in \mathbb{N}$ such that for all $n \in \mathbb{N}$ with $n \geq n_0$ we have $|a_n - a_{n_0}| < \varepsilon$ which implies that $(a_n)_{n\in\mathbb{N}}$ is a Cauchy sequence in $\mathbb{R}$ and hence convergent. According to Lemma 2.3.30 a) there is an $n_1 \in \mathbb{N}$ such that $\sum_{n\geq n_1} |b_n| < \varepsilon/4$. Define the auxiliary sequence $(\tilde{a}_n)_{n\in\mathbb{N}}$ as follows. For $n < n_1$ set $\tilde{a}_n := a_n$ and for $n \geq n_1$ set $\tilde{a}_n := a_n + \sum_{n_1\leq k\leq n} |b_k|$. Note that $\sup_{n\in\mathbb{N}} |a_n - \tilde{a}_n| < \varepsilon/4$. The sequence $(\tilde{a}_n)_{n\in\mathbb{N}}$ is bounded and, for $n \geq n_1$, monotonically increasing and therefore convergent. Set $\tilde{a} := \lim_{n\to\infty} \tilde{a}_n$ and let $n_2 \in \mathbb{N}$ be such that for all $n \geq n_2$ we have $|\tilde{a} - \tilde{a}_n| < \varepsilon/4$. In particular, for all $n \geq n_0 := \max\{n_1, n_2\}$ we have $|a_n - a_{n_0}| < |\tilde{a}_n - \tilde{a}_{n_0}| + \varepsilon/2 < \varepsilon/2 + \varepsilon/2 = \varepsilon$. □

Proof of Lemma 2.3.18. First we prove claim a). Let a be fixed. We show that for any $x \in \mathbb{R}$ with $x > -a$ the limit

$$\lim_{b\to\infty, b\in\mathbb{N}} b + \ln(g_b(1 - 2^{-a-x}))$$

exists. Inductively we get

$$\begin{aligned} b + \ln(g_b(1-2^{-a-x})) &= b + \ln(g_{b-1}(1-2^{-a-x})) + g_{b-1}(1-2^{-a-x}) - 1 \\ &= b + \ln(g(1-2^{-a-x})) + \left(\sum_{j=1}^{b-1} g_j(1-2^{-a-x})\right) - b + 1 \\ &= 1 + \ln(1-2^{-a-x}) - 2^{-a-x} + \sum_{j=1}^{b-1} g_j(1-2^{-a-x}) \end{aligned} \tag{2.3.13}$$

which, according to Lemma 2.3.30 *a*), converges for $b \to \infty$ because $g_j(1-2^{-a-x})$ converges exponentially fast to zero at rate at most $e^{(-2^{-a-1})} < 1$ and start number 1 for $j \to \infty$ in the sense of Definition 2.3.29. For $x \in [0,1]$, according to Lemma 2.3.30 *b*), the convergence is uniform with respect to x.
Next we show claim *b*), i.e. we show that for any $x \in \mathbb{R}$

$$\lim_{a\to\infty, a\in\mathbb{N}} \lim_{b\to\infty, b\in\mathbb{N}} -a + b + \ln(g_b(1-2^{-a-x}))$$

converges. According to the previous calculations this is equivalent to prove that the sequence

$$(\gamma_a)_{a\in\mathbb{N}} := \left(-a + \sum_{j=1}^{\infty} g_j(1-2^{-a-x})\right)_{a\in\mathbb{N}} \tag{2.3.14}$$

converges.[2] In order to show this, we first prove that $(\gamma_a)_{a\in\mathbb{N}}$ is a bounded sequence. For $a \in \mathbb{N}$ we have

$$\gamma_a = \sum_{j=1}^{a} (g_j(1-2^{-a-x}) - 1) + \sum_{j=a+1}^{\infty} g_j(1-2^{-a-x}).$$

Set

$$h_a := h_a(x) := \sum_{j=a+1}^{\infty} g_j(1-2^{-a-x}) \quad \text{and} \quad \tilde{h}_a := \tilde{h}_a(x) := \sum_{j=1}^{a} (g_j(1-2^{-a-x}) - 1).$$

We prove that $(\gamma_a)_{a\in\mathbb{N}}$ is bounded by showing that the sequences $(h_a)_{a\in\mathbb{N}}$ and $(\tilde{h}_a)_{a\in\mathbb{N}}$ both are bounded. We start with h_a. Recall (2.3.10) and (2.3.11), we get

$$\tilde{f}_{a+\lfloor x\rfloor}(2^{-a-x}) \geq \frac{1}{2}\left(1 - \left(1 - \frac{1}{2^{a+x-1}}\right)^{(2^{a+x-1})}\right) = \frac{1}{2}(1-e^{-1}) + \mathcal{O}(2^{-a}).$$

In particular, as $(1-e^{-1})/2 > 0.3$, there is an $a_0 = a_0(x) \in \mathbb{N}$ such that for all $a \in \mathbb{N}$ with $a \geq a_0$ it holds $\tilde{f}_{a+\lfloor x\rfloor}(2^{-a-x}) \geq 0.3$ and therefore

$$\tilde{f}_a(2^{-a-x}) \geq \min\{0.3 \cdot 2^{-\lfloor x\rfloor}, 0.3\} =: c_1 \in (0, 0.3].$$

[2] As we have defined g_j only on the domain $[0,1]$, strictly speaking $g_j(1-2^{-a-x})$ is only well-defined for $a \geq -x$; but since we investigate the asymptotic behaviour for $a \to \infty$, for simplicity of exposition we can ignore this detail without affecting the result.

Thus we can infer

$$g_a(1-2^{-a-x}) = 1 - f_a(2^{-a-x}) \leq 1 - \tilde{f}_a(2^{-a-x}) \leq 1 - c_1.$$

Note that for $y \in [0,1]$ and $i \in \mathbb{N}$ we have $g_i(y) \leq y(e^{y-1})^i$. Set $c_2 := e^{(1-c_1)-1} = e^{-c_1}$. It is $c_2 < 1$. For $j \in \mathbb{N}$ we obtain $g_{a+j}(1-2^{-a-x}) \leq c_2^j(1-c_1)$. Hence

$$h_a = \sum_{j=a+1}^{\infty} g_j(1-2^{-a-x}) = \sum_{j=1}^{\infty} g_{j+a}(1-2^{-a-x}) \leq (1-c_1)\sum_{j=1}^{\infty} c_2^j = \frac{(1-c_1)c_2}{1-c_2}$$

which shows that $(h_a)_{a\in\mathbb{N}}$ is a bounded sequence. Next we show that $(\tilde{h}_a)_{a\in\mathbb{N}}$ is a bounded sequence. It is

$$|\tilde{h}_a| = \left|\sum_{j=1}^{a} -f_j(2^{-a-x})\right| = \sum_{j=1}^{a} f_j(2^{-a-x}) \leq \sum_{j=1}^{a} 2^{j-a-x} = 2^{1-x} - 2^{-a-x+1} \leq 2^{1-x}$$

which shows that $(\tilde{h}_a)_{a\in\mathbb{N}}$ is a bounded sequence. We claim that $\gamma_{a+1} - \gamma_a \geq -f(2^{-a-x-1})$; note that $f(2^{-a-x-1})$ converges exponentially fast to zero for $a \to \infty$ in the sense of Definition 2.3.29; hence, if we prove the claim, Lemma 2.3.31 yields the convergence of the bounded sequence $(\gamma_a)_{a\in\mathbb{N}}$. In order to prove the claim we continue with

$$\sum_{j=1}^{\infty} g_j(1-2^{-a-x}) = \sum_{j=1}^{\infty} 1 - f_j(2^{-a-x}) \leq \sum_{j=1}^{\infty} 1 - f_{j+1}(2^{-a-x-1}) = \sum_{j=2}^{\infty} g_j(1-2^{-a-x-1}).$$

Hence we can infer

$$\begin{aligned}\sum_{j=1}^{\infty} g_j(1-2^{-a-x-1}) - \sum_{j=1}^{\infty} g_j(1-2^{-a-x}) &= g_1(1-2^{-a-x-1}) + \sum_{j=2}^{\infty} g_j(1-2^{-a-x-1}) - \sum_{j=1}^{\infty} g_j(1-2^{-a-x}) \\ &\geq g(1-2^{-a-x-1}) = 1 - f(2^{-a-x-1})\end{aligned}$$

which implies $\gamma_{a+1} - \gamma_a \geq -f(2^{-a-x-1})$. □

Proof of Lemma 2.3.19. In order to prove the continuity of c on $[0,1]$, as it does not complicate the proof, we show uniform continuity, i.e. we show that for any $\tilde{\varepsilon} > 0$ there is a $\delta > 0$ such that for all $x, y \in [0,1]$ with $|x-y| < \delta$ it holds $|c(y) - c(x)| \leq \tilde{\varepsilon}$. However, note that continuity would directly imply uniform continuity anyway as the considered domain is a compact interval. Let $\tilde{\varepsilon} > 0$ be given and consider an arbitrary pair $x, y \in [0,1]$. W.l.o.g. $y > x$ and we write $y = x + \varepsilon$ for $\varepsilon := y - x > 0$. Considering (2.3.13) we obtain

$$\begin{aligned}|c(y) - c(x)| &= c(x+\varepsilon) - c(x) \\ &= \left(\lim_{a\to\infty, a\in\mathbb{N}} 1 - a + \sum_{j=1}^{\infty} g_j(1-2^{-a-x-\varepsilon})\right) - \left(\lim_{a\to\infty, a\in\mathbb{N}} 1 - a + \sum_{j=1}^{\infty} g_j(1-2^{-a-x})\right) \\ &= \lim_{a\to\infty, a\in\mathbb{N}} \sum_{j=1}^{\infty} g_j(1-2^{-a-x-\varepsilon}) - g_j(1-2^{-a-x}).\end{aligned}$$

Using that for $j \in \mathbb{N}$ and $q \in [0,1]$ it is $g_j(1-q) = 1 - f_j(q)$ we can infer

$$|c(y) - c(x)| = \lim_{a\to\infty, a\in\mathbb{N}} \sum_{j=1}^{\infty} f_j(2^{-a-x}) - f_j(2^{-a-x-\varepsilon}).$$

In particular, it suffices to show that there is a $\delta = \delta(\tilde{\varepsilon})$ (where δ does only depend on $\tilde{\varepsilon}$ and is independent of a) and an $a_0 \in \mathbb{N}$ (where a_0 is independent of all other parameters) such that for all $0 < \varepsilon < \delta$ and all $a \in \mathbb{N}$ with $a \geq a_0$

$$\sum_{j=1}^{\infty} f_j(2^{-a-x}) - f_j(2^{-a-x-\varepsilon}) \leq \tilde{\varepsilon}. \tag{2.3.15}$$

In order to prove (2.3.15) we introduce some notation. Let f_0 denote the identity function on $[0,1]$; for $j \in \mathbb{N}$ we define

$$c_j := \frac{f_j(2^{-a-x})}{f_{j-1}(2^{-a-x})} \quad \text{and} \quad \tilde{c}_j = \frac{f_j(2^{-a-x-\varepsilon})}{f_{j-1}(2^{-a-x-\varepsilon})}.$$

As the function $x \mapsto f(x)/x$ is monotonically decreasing for $x \in (0,1]$, for all $j \in \mathbb{N}$ we have $c_j \leq \tilde{c}_j$. We define $\tau_a \in \mathbb{N}$ as the minimum natural number that satisfies $f_{\tau_a}(2^{-a-x-\varepsilon}) \geq \tilde{\varepsilon}/10$. Note that (as for all $q \in [0,1]$ it is $f(q) \leq 2q$) this implies

$$\frac{\tilde{\varepsilon}}{10} \leq f_{\tau_a}(2^{-a-x-\varepsilon}) \leq \frac{\tilde{\varepsilon}}{5}. \tag{2.3.16}$$

Furthermore, we define

$$c := c_{\tau_a} \cdot \ldots \cdot c_1 = \prod_{j=1}^{\tau_a} c_j \quad \text{and} \quad \tilde{c} := \tilde{c}_{\tau_a} \cdot \ldots \cdot \tilde{c}_1 = \prod_{j=1}^{\tau_a} \tilde{c}_j.$$

It is $c < \tilde{c}$. We have

$$c2^{-a-x} = f_{\tau_a}(2^{-a-x}) \quad \text{and} \quad \tilde{c}2^{-a-x-\varepsilon} = f_{\tau_a}(2^{-a-x-\varepsilon}).$$

Thus, together with (2.3.16), we obtain

$$\frac{\tilde{\varepsilon}}{10} \leq \tilde{c}2^{-a-x-\varepsilon} \leq \frac{\tilde{\varepsilon}}{5}. \tag{2.3.17}$$

Let $\kappa = \kappa(\tilde{\varepsilon})$ denote the minimum natural number such that

$$f_{\kappa(\tilde{\varepsilon})}\left(\frac{\tilde{\varepsilon}}{10}\right) \geq 1 - \frac{\tilde{\varepsilon}}{10}. \tag{2.3.18}$$

Note that κ only depends on $\tilde{\varepsilon}$. Now, after having introduced some notation, we will bound the series from (2.3.15); let us abbreviate $\nu(j,a,x) := f_j(2^{-a-x}) - f_j(2^{-a-x-\varepsilon})$. We have

$$\sum_{j=1}^{\infty} f_j(2^{-a-x}) - f_j(2^{-a-x-\varepsilon}) = \sum_{j=1}^{\tau_a-1} \nu(j,a,x) + \sum_{j=\tau_a}^{\tau_a+\kappa(\tilde{\varepsilon})} \nu(j,a,x) + \sum_{j=\tau_a+\kappa(\tilde{\varepsilon})+1}^{\infty} \nu(j,a,x). \tag{2.3.19}$$

We bound the three summands individually. In order to bound the first sum, similarly to (2.3.13), inductively we obtain

$$\begin{aligned}
\ln\left(\frac{1-f_{\tau_a}(2^{-a-x-\varepsilon})}{1-f_{\tau_a}(2^{-a-x})}\right) &= \ln(g_{\tau_a}(1-2^{-a-x-\varepsilon})) - \ln(g_{\tau_a}(1-2^{-a-x})) \\
&= \ln(g_{\tau_a-1}(1-2^{-a-x-\varepsilon})) + g_{\tau_a-1}(1-2^{-a-x-\varepsilon}) - \ln(g_{\tau_a-1}(1-2^{-a-x})) - g_{\tau_a-1}(1-2^{-a-x}) \\
&= \ln(g(1-2^{-a-x-\varepsilon})) - \ln(g(1-2^{-a-x})) + \sum_{j=1}^{\tau_a-1} g_j(1-2^{-a-x-\varepsilon}) - g_j(1-2^{-a-x}) \\
&= \ln(1-2^{-a-x-\varepsilon}) - 2^{-a-x-\varepsilon} - \ln(1-2^{-a-x}) + 2^{-a-x} + \sum_{j=1}^{\tau_a-1} f_j(2^{-a-x}) - f_j(2^{-a-x-\varepsilon}).
\end{aligned}$$

There is a constant $a_0 \in \mathbb{N}$ (independent of all other parameters) such that for all $a \in \mathbb{N}$ with $a \geq a_0$

$$\left|\ln(1-2^{-a-x-\varepsilon}) - 2^{-a-x-\varepsilon} - \ln(1-2^{-a-x}) + 2^{-a-x}\right| < \frac{\tilde{\varepsilon}}{10}.$$

Therefore, for $a \geq a_0$,

$$\begin{aligned}
\sum_{j=1}^{\tau_a-1} \nu(j,a,x) &\leq \frac{\tilde{\varepsilon}}{10} + \ln\left(\frac{1-f_{\tau_a}(2^{-a-x-\varepsilon})}{1-f_{\tau_a}(2^{-a-x})}\right) = \frac{\tilde{\varepsilon}}{10} + \ln\left(\frac{1-\tilde{c}2^{-a-x-\varepsilon}}{1-c2^{-a-x}}\right) \\
&\leq \frac{\tilde{\varepsilon}}{10} + \ln\left(\frac{1-\tilde{c}2^{-a-x-\varepsilon}}{1-\tilde{c}2^{-a-x}}\right).
\end{aligned}$$

Thus, using (2.3.17), we can infer

$$\sum_{j=1}^{\tau_a-1} \nu(j,a,x) \leq \frac{\tilde{\varepsilon}}{10} + \ln\left(1-\frac{\tilde{\varepsilon}}{10}\right) - \ln\left(1-\frac{\tilde{\varepsilon}}{5}2^{\varepsilon}\right) \leq \frac{\tilde{\varepsilon}}{10} - \ln\left(1-\frac{\tilde{\varepsilon}}{5}2^{\varepsilon}\right).$$

Hence, as for any $q > -1$ it is $q/(1+q) \leq \ln(1+q)$ and because for any $0 \leq \tilde{q} \leq 1/2$ it is $\tilde{q}/(1-\tilde{q}) \leq 2\tilde{q}$, for sufficiently small ε we obtain

$$\sum_{j=1}^{\tau_a-1} \nu(j,a,x) \leq \frac{\tilde{\varepsilon}}{10} + \frac{2^{\varepsilon}\tilde{\varepsilon}/5}{1-2^{\varepsilon}\tilde{\varepsilon}/5} \leq \frac{\tilde{\varepsilon}}{10} + \frac{\tilde{\varepsilon}}{5}2^{\varepsilon+1} \leq \frac{3}{5}\tilde{\varepsilon}. \tag{2.3.20}$$

To bound the second sum in (2.3.19) we observe that

$$1 \geq \frac{f_{\tau_a}(2^{-a-x-\varepsilon})}{f_{\tau_a}(2^{-a-x})} = \frac{\tilde{c}2^{-a-x-\varepsilon}}{c2^{-a-x}} = 2^{-\varepsilon}\frac{\tilde{c}}{c} \geq 2^{-\varepsilon}.$$

This implies that there is a function $e : \mathbb{R}^+ \to \mathbb{R}^+$ with $e(\varepsilon) \overset{\varepsilon\to 0}{\to} 0$ (for concreteness, set $e(\varepsilon) = 1-2^{-\varepsilon}$) such that

$$f_{\tau_a}(2^{-a-x}) \geq f_{\tau_a}(2^{-a-x-\varepsilon}) \geq (1-e(\varepsilon))f_{\tau_a}(2^{-a-x}) \geq f_{\tau_a}(2^{-a-x}) - e(\varepsilon).$$

In particular

$$\left|f_{\tau_a}(2^{-a-x}) - f_{\tau_a}(2^{-a-x-\varepsilon})\right| \leq e(\varepsilon).$$

Thus, as $x \mapsto \sum_{0\leq j\leq \kappa(\tilde{\varepsilon})} f_j(x)$ is a continuous function, if ε is sufficiently small (only depending on $\tilde{\varepsilon}$), then

$$\sum_{j=\tau_a}^{\tau_a+\kappa(\tilde{\varepsilon})} \nu(j,a,x) = \sum_{j=0}^{\kappa(\tilde{\varepsilon})} f_j(f_{\tau_a}(2^{-a-x})) - \sum_{j=0}^{\kappa(\tilde{\varepsilon})} f_j(f_{\tau_a}(2^{-a-x-\varepsilon})) < \frac{\tilde{\varepsilon}}{4}. \tag{2.3.21}$$

Next we bound the third summand in (2.3.19). From (2.3.16) and (2.3.18) we obtain

$$f_{\tau_a+\kappa(\tilde{\varepsilon})}(2^{-a-x-\varepsilon}) \geq 1 - \frac{\tilde{\varepsilon}}{10}.$$

Therefore, as for $q \in [0, 0.3]$ it is $g(q) \leq q/2$,

$$\begin{aligned}
\sum_{j=\tau_a+\kappa(\tilde{\varepsilon})+1}^{\infty} \nu(j,a,x) &= \sum_{j=1}^{\infty} f_j(f_{\tau_a+\kappa(\tilde{\varepsilon})}(2^{-a-x})) - f_j(f_{\tau_a+\kappa(\tilde{\varepsilon})}(2^{-a-x-\varepsilon})) \\
&= \sum_{j=1}^{\infty} g_j(1 - f_{\tau_a+\kappa(\tilde{\varepsilon})}(2^{-a-x-\varepsilon})) - g_j(1 - f_{\tau_a+\kappa(\tilde{\varepsilon})}(2^{-a-x})) \\
&\leq \sum_{j=1}^{\infty} g_j(1 - f_{\tau_a+\kappa(\tilde{\varepsilon})}(2^{-a-x-\varepsilon})) \leq \sum_{j=1}^{\infty} g_j\left(\frac{\tilde{\varepsilon}}{10}\right) \leq \sum_{j=1}^{\infty} \left(\frac{1}{2}\right)^j \frac{\tilde{\varepsilon}}{10} = \frac{\tilde{\varepsilon}}{10}.
\end{aligned} \tag{2.3.22}$$

In summary, combining the individual bounds (2.3.20), (2.3.21) and (2.3.22) for the three summands of (2.3.19), we obtain that there is a $\delta > 0$ (only depending on $\tilde{\varepsilon}$) and an $a_0 \in \mathbb{N}$ (independent of all other parameters) such that for all $0 < \varepsilon < \delta$ and all $a \in \mathbb{N}$ with $a \geq a_0$

$$\sum_{j=1}^{\infty} f_j(2^{-a-x}) - f_j(2^{-a-x-\varepsilon}) \leq \frac{3}{5}\tilde{\varepsilon} + \frac{\tilde{\varepsilon}}{4} + \frac{\tilde{\varepsilon}}{10} < \tilde{\varepsilon}$$

which shows (2.3.15) and hence completes the proof. □

Proof of Corollary 2.3.21. As, according to Lemma 2.3.19, the function c is continuous on the interval $[0, 1]$, the claim follows from Theorem 2.3.9 since for any $a \in \mathbb{N}$ the function $x \mapsto -a + \eta(a, x)$ (where η is defined as in Lemma 2.3.18 *b*)) is monotonically increasing on the interval $[0, 1]$. □

Proof of Lemma 2.3.22. Define $h : \mathbb{N} \to \mathbb{R}^+$ by $h(n) := \min\{\lfloor n^{0.1} \rfloor, \varepsilon(n)n - 1\} = \omega(1)$ and let $\tilde{X}_i$ and $\tilde{D}_i$ be defined as in Theorem 2.3.14. To lighten the notation we write ε instead of $\varepsilon(n)$ and s instead of s_n. In order to prove the claim, we first bound several variances and

expectations. Note that for all $i \in \{(1-\varepsilon)n, \dots, n-1\}$ it is $s(i) = \Theta(n)$. We have

$$\begin{aligned} Var\left[\sum_{i=(1-\varepsilon)n}^{n-h(n)-1} \frac{D_i}{s(i)}\right] &= \sum_{i=(1-\varepsilon)n}^{n-h(n)-1} Var\left[\frac{D_i}{s(i)}\right] = \sum_{i=(1-\varepsilon)n}^{n-h(n)-1} \frac{1}{s(i)^2} Var[D_i] = \sum_{i=(1-\varepsilon)n}^{n-h(n)-1} \frac{1}{s(i)^2} Var\left[X_i - \frac{n}{n-i}\right] \\ &= \sum_{i=(1-\varepsilon)n}^{n-h(n)-1} \frac{1}{s(i)^2} Var[X_i] = \sum_{i=(1-\varepsilon)n}^{n-h(n)-1} \frac{1}{s(i)^2} \frac{(i-1)(n-1)}{(n-i)^2} \\ &= \mathcal{O}\left(\sum_{i=(1-\varepsilon)n}^{n-h(n)-1} \frac{1}{(n-i)^2}\right) = \mathcal{O}\left(\sum_{i=h(n)+1}^{\varepsilon n} \frac{1}{i^2}\right) = o(1). \end{aligned}$$

Analogously we obtain

$$Var\left[\sum_{i=0}^{n-h(n)-1} \frac{\tilde{D}_i}{n}\right] = \sum_{i=0}^{n-h(n)-1} \frac{1}{n^2} Var[\tilde{X}_i] = \sum_{i=0}^{n-h(n)-1} \frac{1}{n^2} \frac{ni}{(n-i)^2} \leq \sum_{i=0}^{n-h(n)-1} \frac{1}{(n-i)^2} = o(1). \tag{2.3.23}$$

Hence for the standard deviations we have

$$\sigma\left[\sum_{i=(1-\varepsilon)n}^{n-h(n)-1} \frac{D_i}{s(i)}\right] = o(1) \quad \text{and} \quad \sigma\left[\sum_{i=0}^{n-h(n)-1} \frac{\tilde{D}_i}{n}\right] = o(1). \tag{2.3.24}$$

We continue with

$$\begin{aligned} E\left[\sum_{i=(1-\varepsilon)n}^{n-h(n)-1} \frac{D_i}{s(i)}\right] &= \sum_{i=(1-\varepsilon)n}^{n-h(n)-1} \frac{E[D_i]}{s(i)} = \sum_{i=(1-\varepsilon)n}^{n-h(n)-1} \frac{-1}{s(i)(n-i)} = \mathcal{O}\left(\frac{1}{n} \sum_{i=(1-\varepsilon)n}^{n-h(n)-1} \frac{1}{n-i}\right) \\ &= \mathcal{O}\left(\frac{1}{n} \sum_{i=h(n)+1}^{\varepsilon n} \frac{1}{i}\right) = \mathcal{O}\left(\frac{\ln(n)}{n}\right). \end{aligned} \tag{2.3.25}$$

As for all $i \in \{0, 1, \dots, n-h(n)-1\}$ it is $E[\tilde{D}_i] = 0$, by linearity of expectation we have

$$E\left[\sum_{i=0}^{n-h(n)-1} \frac{\tilde{D}_i}{n}\right] = 0. \tag{2.3.26}$$

From (2.3.24) and (2.3.25) we obtain that there are $f_1, f_2 : \mathbb{N} \to \mathbb{R}^+$ with $f_1, f_2 = o(1)$ such that

$$\sigma\left[\sum_{i=(1-\varepsilon)n}^{n-h(n)-1} \frac{D_i}{s(i)}\right] \leq f_1(n) \quad \text{and} \quad \left|E\left[\sum_{i=(1-\varepsilon)n}^{n-h(n)-1} \frac{D_i}{s(i)}\right]\right| \leq f_2(n).$$

Set $k := f_2 + \sqrt{f_1} = o(1)$. Using the Chebyshev inequality we obtain

$$\begin{aligned} P\left[\left|\sum_{i=(1-\varepsilon)n}^{n-h(n)-1} \frac{D_i}{s(i)}\right| \geq k(n)\right] &\leq P\left[\left|\sum_{i=(1-\varepsilon)n}^{n-h(n)-1} \frac{D_i}{s(i)} - E\left[\sum_{i=(1-\varepsilon)n}^{n-h(n)-1} \frac{D_i}{s(i)}\right]\right| \geq k(n) - f_2(n)\right] \\ &\leq \frac{f_1(n)^2}{f_1(n)} = f_1(n) = o(1). \end{aligned}$$

This implies

$$\sum_{i=(1-\varepsilon)n}^{n-h(n)-1} \frac{D_i}{s(i)} \xrightarrow{p} 0. \tag{2.3.27}$$

Analogously, using (2.3.24), (2.3.26) and the Chebyshev inequality, we get

$$\sum_{i=0}^{n-h(n)-1} \frac{\tilde{D}_i}{n} \xrightarrow{p} 0. \tag{2.3.28}$$

Next we show

$$\left(\sum_{i=n-h(n)}^{n-1} \frac{D_i}{s(i)} - \sum_{i=n-h(n)}^{n-1} \frac{D_i}{n} \right) \xrightarrow{p} 0. \tag{2.3.29}$$

We have

$$\begin{aligned} \left| \sum_{i=n-h(n)}^{n-1} \frac{D_i}{s(i)} - \sum_{i=n-h(n)}^{n-1} \frac{D_i}{n} \right| &\leq \sum_{i=n-h(n)}^{n-1} |D_i| \left(\frac{1}{s(i)} - \frac{1}{n} \right) \leq \left(\frac{1}{s(n-h(n))} - \frac{1}{n} \right) \sum_{i=n-h(n)}^{n-1} |D_i| \\ &= \left(\frac{1}{n + \mathcal{O}(n^{0.8})} - \frac{1}{n} \right) \sum_{i=n-h(n)}^{n-1} |D_i| = \mathcal{O}(n^{-1.2}) \sum_{i=n-h(n)}^{n-1} |D_i| = \mathcal{O}(n^{-1.2}) \sum_{i=n-h(n)}^{n-1} X_i + \frac{n}{n-i}. \end{aligned} \tag{2.3.30}$$

It is

$$E\left[\sum_{i=n-h(n)}^{n-1} X_i \right] = \sum_{i=n-h(n)}^{n-1} E[X_i] = \sum_{i=n-h(n)}^{n-1} \frac{n-1}{n-i} \leq n \sum_{i=n-h(n)}^{n-1} \frac{1}{n-i} = n \sum_{i=1}^{h(n)} \frac{1}{i} = \mathcal{O}(\ln(n)n)$$

and

$$\sum_{i=n-h(n)}^{n-1} \frac{n}{n-i} = n \sum_{i=1}^{h(n)} \frac{1}{i} = \mathcal{O}(\ln(n)n).$$

Hence, using Markov's inequality, we obtain

$$P\left[\sum_{i=n-h(n)}^{n-1} X_i + \frac{n}{n-i} \geq n^{1.1} \right] \leq \frac{E\left[\sum_{i=n-h(n)}^{n-1} X_i + \frac{n}{n-i} \right]}{n^{1.1}} = \frac{\mathcal{O}(\ln(n)n)}{n^{1.1}} = o(1).$$

This, together with (2.3.30) yields that, whp,

$$\left| \sum_{i=n-h(n)}^{n-1} \frac{D_i}{s(i)} - \sum_{i=n-h(n)}^{n-1} \frac{D_i}{n} \right| = \mathcal{O}(n^{-0.1}) = o(1)$$

which implies (2.3.29). As our next step, we prove

$$\sum_{i=n-h(n)}^{n-1} \frac{D_i}{n} \xrightarrow{d} G. \tag{2.3.31}$$

We have

$$\begin{aligned}\sum_{i=n-h(n)}^{n-1} \frac{D_i}{n} &= \sum_{i=n-h(n)}^{n-1} \frac{X_i - \frac{n}{n-i}}{n} = \frac{1}{n} \sum_{i=n-h(n)}^{n-1} \left(X_i - \frac{n-1}{n-i} \right) + \frac{1}{n} \sum_{i=n-h(n)}^{n-1} \left(\frac{n-1}{n-i} - \frac{n}{n-i} \right) \\ &= \frac{1}{n} \sum_{i=n-h(n)-1}^{n-2} \left(X_{i+1} - \frac{n-1}{(n-1)-i} \right) + \frac{1}{n} \sum_{i=n-h(n)}^{n-1} \frac{-1}{n-i} \\ &= \frac{n-1}{n} \left(\sum_{i=n-h(n)-1}^{n-2} \frac{X_{i+1} - \frac{n-1}{(n-1)-i}}{n-1} \right) + \mathcal{O}\left(\frac{\ln(n)}{n} \right). \end{aligned} \tag{2.3.32}$$

The random variables $X_{i+1} = X_{i+1}^{(n)}$ and $\tilde{X}_i^{(n-1)}$ both have the same distribution, in particular

$$X_{i+1}^{(n)}, \tilde{X}_i^{(n-1)} \sim Geom\left(\frac{(n-1)-i}{n-1} \right). \tag{2.3.33}$$

Hence, as

$$E\left[\tilde{X}_i^{(n-1)} \right] = \frac{n-1}{(n-1)-i},$$

by applying Theorem 2.3.14 we can infer

$$\sum_{i=0}^{n-2} \frac{X_{i+1} - \frac{n-1}{(n-1)-i}}{n-1} \xrightarrow{d} G.$$

Therefore, using (2.3.28), (2.3.33) and Theorem 2.3.13 we obtain

$$\sum_{i=n-h(n)-1}^{n-2} \frac{X_{i+1} - \frac{n-1}{(n-1)-i}}{n-1} \xrightarrow{d} G.$$

Thus, as $(n-1)/n \overset{n\to\infty}{\to} 1$ and $\mathcal{O}(\ln(n)/n) = o(1)$, together with (2.3.32), we arrive at (2.3.31). We continue with

$$\begin{aligned}\sum_{i=(1-\varepsilon)n}^{n-1} \frac{D_i}{s(i)} &= \sum_{i=(1-\varepsilon)n}^{n-h(n)-1} \frac{D_i}{s(i)} + \sum_{i=n-h(n)}^{n-1} \frac{D_i}{s(i)} \\ &= \sum_{i=(1-\varepsilon)n}^{n-h(n)-1} \frac{D_i}{s(i)} + \sum_{i=n-h(n)}^{n-1} \frac{D_i}{n} + \sum_{i=n-h(n)}^{n-1} \frac{D_i}{s(i)} - \sum_{i=n-h(n)}^{n-1} \frac{D_i}{n}. \end{aligned}$$

From (2.3.27), (2.3.31) and (2.3.29) we know that

$$\sum_{i=(1-\varepsilon)n}^{n-h(n)-1} \frac{D_i}{s(i)} \xrightarrow{p} 0 \quad \text{and} \quad \sum_{i=n-h(n)}^{n-1} \frac{D_i}{n} \xrightarrow{d} G \quad \text{and} \quad \left(\sum_{i=n-h(n)}^{n-1} \frac{D_i}{s(i)} - \sum_{i=n-h(n)}^{n-1} \frac{D_i}{n} \right) \xrightarrow{p} 0.$$

Hence, using Theorem 2.3.13, we can infer

$$\sum_{i=(1-\varepsilon)n}^{n-1} \frac{D_i}{s(i)} \xrightarrow{d} G.$$

□

Proof of Corollary 2.3.23. As in the proof of Lemma 2.3.22, to lighten the notation we write ε instead of $\varepsilon(n)$ and s instead of s_n. Note that according to Theorem 2.3.11 and Lemma 2.3.22

$$\sup_{x\in\mathbb{R}} \left| P\left[\sum_{i=(1-\varepsilon)n}^{n-1} \frac{D_i}{s(i)} \geq x \right] - P[G \geq x] \right| = o(1). \tag{2.3.34}$$

Note that the $o(1)$ terms in the remainder of the proof are independent of l and k and, as usual, are with respect to $n \to \infty$. We have

$$\begin{aligned} P\left[\left\lceil l(n) + \sum_{i=(1-\varepsilon)n}^{n-1} \frac{D_i}{s(i)} \right\rceil \geq k(n) \right] &= P\left[\sum_{i=(1-\varepsilon)n}^{n-1} \frac{D_i}{s(i)} > k(n) - 1 - l(n) \right] \\ &= P\left[\sum_{i=(1-\varepsilon)n}^{n-1} \frac{D_i}{s(i)} \geq k(n) - 1 - l(n) \right] + o(1). \end{aligned}$$

Thus, using (2.3.34), we can infer

$$\begin{aligned} P\left[\left\lceil l(n) + \sum_{i=(1-\varepsilon)n}^{n-1} \frac{D_i}{s(i)} \right\rceil \geq k(n) \right] &= P[G \geq k(n) - 1 - l(n)] + o(1) \\ &= P[l(n) + G > k(n) - 1] + o(1) \\ &= P[\lceil l(n) + G \rceil \geq k(n)] + o(1). \end{aligned}$$

□

Proof of Lemma 2.3.24. For sufficiently large a, for all $\eta \in [0,1]$ it holds $f(L_1^{(\eta)}) > U_1^{(\eta)}$ and therefore, for all $k \in \mathbb{N}$,

$$f_k(U_1^{(\eta)}) < f_{k+1}(L_1^{(\eta)}) < f_{k+1}(U_1^{(\eta)}). \tag{2.3.35}$$

Recall that for any $x \in (0,1)$

$$f_j(x) \overset{j\to\infty}{\to} 1 \tag{2.3.36}$$

and let $k^* \in \mathbb{N}$ denote the minimum natural number such that $f_{k^*-2}(L_1^{(0)}) \geq 0.85$; in particular, k^* is independent of η. Note that $f_4(0.85) < 0.998$ and that for all $\eta \in [0,1]$, for all $j \in \mathbb{N}$ it holds $f_j(U_1^{(\eta)}) \leq f_j(U_1^{(0)})$ and $f_{j+2}(L_1^{(\eta)}) \geq f_j(L_1^{(0)})$. This, together with (2.3.35), yields, for sufficiently large a, for all $\eta \in [0,1]$

$$\begin{aligned} 0.85 \leq f_{k^*-2}(L_1^{(0)}) \leq f_{k^*}(L_1^{(\eta)}) \leq f_{k^*}(U_1^{(\eta)}) \leq f_{k^*}(U_1^{(0)}) \leq f_{k^*+1}(L_1^{(0)}) = f_4(f_{k^*-3}(L_1^{(0)})) \\ \leq f_4(0.85) \leq 0.998. \end{aligned}$$

In particular

$$0.85 \leq f_{k^*}(L_1^{(\eta)}) \leq f_{k^*}(U_1^{(\eta)}) \leq 0.998. \tag{2.3.37}$$

We show

$$f_{k^*}(U_1^{(\eta)}) - f_{k^*}(L_1^{(\eta)}) \leq 2^{-a+2}. \tag{2.3.38}$$

To do this, for $i \in \{1, 2, \ldots, k^*\}$ and where f_0 denotes the identity function on $[0,1]$, essentially like in the proof of Lemma 2.3.19, we define

$$c_i = c_i(\eta) := \frac{f_i(U_1^{(\eta)})}{f_{i-1}(U_1^{(\eta)})} \quad \text{and} \quad \tilde{c}_i = \tilde{c}_i(\eta) := \frac{f_i(L_1^{(\eta)})}{f_{i-1}(L_1^{(\eta)})}.$$

The function $x \mapsto f(x)/x$ is monotonically decreasing for $x \in (0,1]$. Hence, since for all $i \in \mathbb{N}$, for all $\eta \in [0,1]$ it is $f_{i-1}(U_1^{(\eta)}) \geq f_{i-1}(L_1^{(\eta)})$, it holds $c_i \leq \tilde{c}_i$. Set $c = c(\eta) := c_{k^*} \cdot \ldots \cdot c_1$ and $\tilde{c} = \tilde{c}(\eta) := \tilde{c}_{k^*} \cdot \ldots \cdot \tilde{c}_1$. We have

$$f_{k^*}(U_1^{(\eta)}) = c \cdot U_1^{(\eta)} \quad \text{and} \quad f_{k^*}(L_1^{(\eta)}) = \tilde{c} \cdot L_1^{(\eta)}.$$

Note that $c \leq \tilde{c} \leq 2^{a+2}$ as otherwise we had $f_{k^*}(L_1^{(\eta)}) > 1$ which is a contradiction. Thus

$$\begin{aligned} f_{k^*}(U_1^{(\eta)}) - f_{k^*}(L_1^{(\eta)}) &= c2^{-a-\eta} - \tilde{c}2^{-a-\eta} + \tilde{c} \cdot 2^{-2a-\eta} \leq \tilde{c}2^{-a-\eta} - \tilde{c}2^{-a-\eta} + \tilde{c} \cdot 2^{-2a-\eta} \\ &= \tilde{c} \cdot 2^{-2a-\eta} \leq 2^{-a+2}, \end{aligned}$$

hence we arrive at (2.3.38). Note that for $0 < x < 0.15$ and $\varepsilon > 0$ such that $x + \varepsilon \leq 0.15$, as a direct consequence of the mean value theorem, we have

$$g(x+\varepsilon) - g(x) \leq \varepsilon \max_{y \in [x, x+\varepsilon]} g'(y) \leq \varepsilon \max_{y \in [0, 0.15]} (1+y)e^{y-1} < \varepsilon/2.$$

Consequently, as for $x \in [0,1]$ it holds $f(x) = 1 - g(1-x)$, for $0.85 < x' < 1$ and $\varepsilon > 0$ such that $x' - \varepsilon \geq 0.85$ we have

$$f(x') - f(x' - \varepsilon) < \varepsilon/2. \tag{2.3.39}$$

Recall from (2.3.37) that $0.85 \leq f_{k^*}(L_1^{(\eta)}) \leq f_{k^*}(U_1^{(\eta)})$. Thus, from (2.3.38) and (2.3.39), we can infer that for all $j \in \mathbb{N}_0$

$$f_{k^*+j}(U_1^{(\eta)}) - f_{k^*+j}(L_1^{(\eta)}) \leq 2^{-a+2}2^{-j}.$$

In particular, for $\tilde{b} := k^* + \lfloor a/2 \rfloor + 3$ and for all $j \in \mathbb{N}_0$ we have

$$f_{\tilde{b}+j}(U_1^{(\eta)}) - f_{\tilde{b}+j}(L_1^{(\eta)}) \leq 2^{-\frac{3}{2}a}.$$

Next we show that there is a $b^* \in \mathbb{N}$ with $b^* \geq \tilde{b}$ such that for all $j \in \mathbb{N}_0$

$$1 - f_{b^*+j}(L_1^{(\eta)}) \leq 2^{-a} \quad \text{and} \quad 1 - f_{b^*}(U_1^{(\eta)}) \geq 2^{-a}/e^2. \tag{2.3.40}$$

Note that for all $x \in (0,1)$

$$g(x) > x/e \tag{2.3.41}$$

and recall from (2.3.37) that $f_{k^*}(U_1^{(\eta)}) \leq 0.998$ and hence $1 - f_{k^*}(U_1^{(\eta)}) \geq 0.002$. Therefore, for sufficiently large a, we can infer

$$1 - f_{\tilde{b}}(U_1^{(\eta)}) \geq 0.002 \cdot e^{-a/2-3} \geq 2^{-a}.$$

Hence the existence of a $b^* \in \mathbb{N}$ that satisfies (2.3.40) follows from (2.3.35), (2.3.36) and (2.3.41). Thus, in summary, until now we have shown that there is a b^* such that for sufficiently large a for all $j \in \mathbb{N}_0$

$$f_{b^*+j}(U_1^{(\eta)}) - f_{b^*+j}(L_1^{(\eta)}) \leq 2^{-\frac{3}{2}a} \text{ and } 1 - f_{b^*+j}(L_1^{(\eta)}) \leq 2^{-a} \text{ and } 1 - f_{b^*}(U_1^{(\eta)}) \geq 2^{-a}/e^2. \tag{2.3.42}$$

Therefore, to complete the proof, we are left to show that, for sufficiently large a, for all $j \in \mathbb{N}_0$

$$\frac{1 - f_{b^*+j}(L_1^{(\eta)})}{1 - f_{b^*+j}(U_1^{(\eta)})} \leq 1 + 2^{-a/2+4}. \tag{2.3.43}$$

Set $L_2^{(\eta)} := f_{b^*}(L_1^{(\eta)})$ and $U_2^{(\eta)} := f_{b^*}(U_1^{(\eta)})$. For $j \in \mathbb{N}$ (and where g_0 denotes the identity function on $[0,1]$) we define

$$d_j(\eta) := \frac{g_j(1 - U_2^{(\eta)})}{g_{j-1}(1 - U_2^{(\eta)})} = \frac{1 - f_j(U_2^{(\eta)})}{1 - f_{j-1}(U_2^{(\eta)})} \quad \text{and} \quad \tilde{d}_j(\eta) := \frac{g_j(1 - L_2^{(\eta)})}{g_{j-1}(1 - L_2^{(\eta)})} = \frac{1 - f_j(L_2^{(\eta)})}{1 - f_{j-1}(L_2^{(\eta)})}.$$

To lighten the notation we write d_j instead of $d_j(\eta)$ and $\tilde{d}_j$ instead of $\tilde{d}_j(\eta)$. For $j \in \mathbb{N}$, the function $x \mapsto g_j(x)/x$ is monotonically increasing and it is $g_{j-1}(1 - L_2^{(\eta)}) \geq g_{j-1}(1 - U_2^{(\eta)})$. Thus, together with (2.3.35) and (2.3.41), we obtain that for all $j \in \mathbb{N}$

$$\frac{1}{e} \leq \tilde{d}_{j+1} \leq d_j \leq \tilde{d}_j.$$

Thus, for all $j \in \mathbb{N}$, we can infer

$$\frac{1 - f_{b^*+j}(L_1^{(\eta)})}{1 - f_{b^*+j}(U_1^{(\eta)})} = \frac{g_j(1 - L_2^{(\eta)})}{g_j(1 - U_2^{(\eta)})} = \frac{\tilde{d}_1 \cdot \ldots \cdot \tilde{d}_j(1 - L_2^{(\eta)})}{d_1 \cdot \ldots \cdot d_j(1 - U_2^{(\eta)})} \leq \frac{\tilde{d}_1(1 - L_2^{(\eta)})}{d_j(1 - U_2^{(\eta)})} \leq \frac{e\tilde{d}_1(1 - L_2^{(\eta)})}{1 - U_2^{(\eta)}}.$$

Hence, using that $\tilde{d}_1 = e^{-L_2^{(\eta)}}$ and (2.3.42), we obtain

$$\frac{1-f_{b^*+j}(L_1^{(\eta)})}{1-f_{b^*+j}(U_1^{(\eta)})} \leq \frac{e^{1-L_2^{(\eta)}}(1-L_1^{(\eta)})}{1-U_2^{(\eta)}} \leq e^{(2^{-a})}\frac{1-U_2^{(\eta)}+2^{-\frac{3}{2}a}}{1-U_2^{(\eta)}}$$
$$\leq e^{(2^{-a})}\left(1+\frac{2^{-\frac{3}{2}a}}{2^{-a}/e^2}\right) \leq e^{(2^{-a})}(1+2^{-a/2+3}).$$

Hence, as for $x \in [0,1]$ it holds $e^x \leq 1+2x$, for sufficiently large a, for all $j \in \mathbb{N}_0$ we have

$$\frac{1-f_{b^*+j}(L_1^{(\eta)})}{1-f_{b^*+j}(U_1^{(\eta)})} \leq (1+2^{-a+1})(1+2^{-a/2+3}) \leq 1+2^{-a/2+4}$$

which shows (2.3.43) and thus completes the proof. □

Proof of Corollary 2.3.25. Let $b \in \mathbb{N}$ with $b \geq b^*$. We have

$$\ln\left(\frac{1}{f_b(L_1^{(\eta)})}-1\right)-\ln\left(\frac{1}{f_b(U_1^{(\eta)})}-1\right) = \ln\left(\frac{1-f_b(L_1^{(\eta)})}{f_b(L_1^{(\eta)})}\cdot\frac{f_b(U_1^{(\eta)})}{1-f_b(U_1^{(\eta)})}\right).$$

Using Lemma 2.3.24, for sufficiently large a, we get

$$\frac{1-f_b(L_1^{(\eta)})}{f_b(L_1^{(\eta)})}\cdot\frac{f_b(U_1^{(\eta)})}{1-f_b(U_1^{(\eta)})} = \frac{1-f_b(L_1^{(\eta)})}{1-f_b(U_1^{(\eta)})}\cdot\frac{f_b(U_1^{(\eta)})}{f_b(L_1^{(\eta)})} \leq (1+2^{-a/2+4})\frac{f_b(L_1^{(\eta)})+2^{-\frac{3}{2}a}}{f_b(L_1^{(\eta)})}$$
$$= (1+2^{-a/2+4})\left(1+\frac{2^{-\frac{3}{2}a}}{f_b(L_1^{(\eta)})}\right) \leq (1+2^{-a/2+4})(1+2^{-\frac{3}{2}a+1})$$
$$\leq 1+2^{-a/2+5}.$$

Therefore, as for all $x > -1$ it holds $\ln(1+x) \leq x$, we obtain

$$\ln\left(\frac{1}{f_b(L_1^{(\eta)})}-1\right)-\ln\left(\frac{1}{f_b(U_1^{(\eta)})}-1\right) \leq \ln(1+2^{-a/2+5}) \leq 2^{-a/2+5} \overset{a\to\infty}{\to} 0.$$

□

2.3.5 Proof of the Main Result

After the preparatory work, we now are able to prove the main result of Section 2.3, i.e. Theorem 2.3.2.

Proof of Theorem 2.3.2. To prove the claim we will split the information spreading process into three phases. In the first phase, the number of informed nodes almost doubles in each round. The second phase consists of constantly many steps (in the end we will consider the limit when this number grows) in which we follow the deterministic sequence from Lemma 2.3.15. In the third phase, in each step we investigate how many pushes are needed to

inform the next node; this, essentially, is one step of the Coupon Collector's Problem (CCP); a relation between *Push* and the CCP was also recognised in [35]. Afterwards we will carefully convert this number of individual pushes into a (possibly fractional) number of rounds needed to inform the next node; this converting is critical, in particular, an approximate converting by dividing by n would not be accurate enough. Let $a \in \mathbb{N}$ denote a (large) natural number. Let $\delta = 0.01$, set $t_0 = t_0(\delta, n) = (1/2 - \delta)\lfloor \log_2(n) \rfloor$ and let $(\alpha_t)_{t\in\mathbb{N}} = (\alpha_t(t_0, n))_{t\in\mathbb{N}}$ be defined as in Lemma 2.3.15.

Phase 1. Phase 1 consists of the first $\lfloor \log_2(n) \rfloor - a$ rounds. According to Lemma 2.3.17

$$l_1 := 2^{-a}2^{\lfloor \log_2(n) \rfloor}(1 - 2^{-a}) \leq \alpha_{\lfloor \log_2(n) \rfloor - a + 1} n \leq 2^{-a}2^{\lfloor \log_2(n) \rfloor} =: u_1. \tag{2.3.44}$$

At this point, using Lemma 2.3.17, we could also bound $|I_{\lfloor \log_2(n) \rfloor - a + 1}|$ respectively; however this is not yet necessary, we will bound the number of informed nodes at the end of Phase 2.

Phase 2. Let $b^* = b^*(a) \in \mathbb{N}$ be defined as in Lemma 2.3.24 and let $b = b(a) \in \mathbb{N}$ with $b \geq b^*$ (we will specify the choice of b later). Phase 2 consists of the next b rounds. Recall the function $x : \mathbb{N} \to [0, 1)$ defined by $x(n) = \log_2(n) - \lfloor \log_2(n) \rfloor$. Set

$$l_2 := f_b(l_1/n)n = f_b((1 - 2^{-a})2^{-a-x(n)})n \quad \text{and} \quad u_2 := f_b(u_1/n)n = f_b(2^{-a-x(n)})n.$$

Lemma 2.3.24 yields

$$u_2 - l_2 \leq 2^{-\frac{3}{2}a} n \quad \text{and} \quad n - l_2 \leq 2^{-a} n. \tag{2.3.45}$$

As f is monotonically increasing, from (2.3.44) we obtain

$$l_2 \leq \alpha_{\lfloor \log_2(n) \rfloor - a + 1 + b} n \leq u_2.$$

Thus, using Lemma 2.3.15 and that, according to (2.3.45), $u_2 \geq l_2 \geq n - 2^{-a}n \geq n/2$ we get that with probability $1 + \mathcal{O}(n^{-2\delta})$

$$(1 - 2n^{-0.2})l_2 \leq |I_{\lfloor \log_2(n) \rfloor - a + 1 + b}| \leq (1 + 2n^{-0.2})u_2. \tag{2.3.46}$$

For simplicity of exposition we drop the $(1 \pm 2n^{-0.2})$ factors; considering the subsequent calculations, it is easy to verify that this does not affect the result. We call the event that (2.3.46) holds L_1 and from now on we condition on L_1. Note that conditioned on L_1, according to the definition of u_2, a linear fraction (depending on a and b) of the nodes is uninformed after Phase 2.

Phase 3. We number all nodes from 1 to n and assume that the pushes within a round are performed sequentially according to that numbering. This does not change the protocol. We assume that node 1 is the node that is already informed in the beginning of the process. For $j \in \mathbb{N}$ define the random variable P_j that takes values in $\{1, 2, \ldots, n\}$ as follows. The jth push operation that is performed during the information spreading process pushes the piece of information to node P_j. Note that the entire process is captured by the random sequence $(P_j)_{j\in\mathbb{N}}$. For $i \in \{2, \ldots, n\}$ let

$$N_i := \min \left\{ t \mid t \in \mathbb{N}, \left| \{1\} \cup \bigcup_{j=1}^{t} \{P_j\} \right| = i \right\},$$

i.e. after the N_ith push, for the first time i nodes are informed. Furthermore, for $i \in \{2, \dots, n-1\}$ let

$$Y_i := N_{i+1} - N_i$$

denote the number of pushes needed to inform the $(i+1)$st node if one starts with i informed nodes. Hence $Y_2, Y_3, \dots, Y_{n-1}$ are independent random variables and for $i \in \{2, 3, \dots, n-1\}$

$$Y_i \sim Geom\left(\frac{n-i}{n-1}\right). \tag{2.3.47}$$

For $i \in \{2, 3, \dots, n-1\}$ and $j \in \{1, \dots, Y_i\}$ let $r_{i,j} \in \{1, 2, \dots, X_n\}$ denote the round in which the jth push after the ith node has been informed is made. For $r \in R(i) := \{r_{i,1}, \dots, r_{i,Y_i}\}$ let $Y_{i,r}$ denote the number of pushes that Y_i represents which take place in round r. Hence

$$Y_i = \sum_{r \in R(i)} Y_{i,r}.$$

Now, for any $i \in \{2, \dots, n-1\}$, we define the random variable Z_i by

$$\frac{Y_i}{Z_i} = \sum_{r \in R(i)} \frac{Y_{i,r}}{|I_r|}.$$

If a round starts with j informed nodes, then there are j pushes in that round; thus, intuitively, the term Y_i/Z_i represents the fractional number of rounds that is needed to inform the $(i+1)$st node after the ith node has been informed. We have

$$|I_{r_{i,1}}| \leq Z_i \leq i.$$

Hence according to Lemma 2.3.16 for $c = 1 - 1/e$, as $i \leq |I_{r_{i,1}+1}|$, with probability $1 + \mathcal{O}(n^{-2\delta})$

$$(1 + \mathcal{O}(n^{-0.2}))e(i - cn) \leq |I_{r_{i,1}}| \quad \text{for all } i \in \{\lfloor l_2 \rfloor, \dots, n-1\}. \tag{2.3.48}$$

We call the event that (2.3.48) holds L_2 and from now on we condition on L_2, in particular

$$(1 + \mathcal{O}(n^{-0.2}))e(i - cn) \leq Z_i \leq i. \tag{2.3.49}$$

We write $X^{(3)} = X_n^{(3)}$ for the number of rounds needed after Phase 2 to finish the information spreading process, i.e.

$$X^{(3)} = X_n^{(3)} = X_n - \lfloor \log_2(n) \rfloor + a - b. \tag{2.3.50}$$

Using (2.3.49) we obtain

$$\left\lceil \sum_{i=\lfloor u_2 \rfloor}^{n-1} \frac{Y_i}{i} \right\rceil \leq \left\lceil \sum_{i=\lfloor u_2 \rfloor}^{n-1} \frac{Y_i}{Z_i} \right\rceil \leq X^{(3)} \leq \left\lceil \sum_{i=\lceil l_2 \rceil}^{n-1} \frac{Y_i}{Z_i} \right\rceil \leq \left\lceil \sum_{i=\lceil l_2 \rceil}^{n-1} \frac{Y_i}{(1 + \mathcal{O}(n^{-0.2}))e(i - cn)} \right\rceil. \tag{2.3.51}$$

Now we continue to derive upper and lower bounds for $X^{(3)}$. We start with the lower bound. Recall (2.3.47), let

$$D_i := Y_i - \frac{n}{n-i} \tag{2.3.52}$$

and set $k_l = n/l_2$ and $k_u = n/u_2$. By applying Corollary 2.3.23, for $G \sim Gumbel_\gamma$ we obtain

$$X^{(3)} \geq \left\lceil \sum_{i=\lfloor n/k_u \rfloor}^{n-1} \frac{Y_i}{i} \right\rceil = \left\lceil \sum_{i=\lfloor n/k_u \rfloor}^{n-1} \frac{\frac{n}{n-i}}{i} + \sum_{i=\lfloor n/k_u \rfloor}^{n-1} \frac{D_i}{i} \right\rceil \overset{\delta}{=} \left\lceil \sum_{i=\lfloor n/k_u \rfloor}^{n-1} \frac{1}{i(1-i/n)} + G \right\rceil.$$

Next we use

$$\frac{1}{i(1-i/n)} = \frac{1}{n-i} + \frac{1}{i}.$$

This yields

$$X^{(3)} \overset{\delta}{\geq} \left\lceil \sum_{i=\lfloor n/k_u \rfloor}^{n-1} \frac{1}{n-i} + \sum_{i=\lfloor n/k_u \rfloor}^{n-1} \frac{1}{i} + G \right\rceil = \left\lceil \sum_{i=1}^{n-\lfloor n/k_u \rfloor} \frac{1}{i} + \sum_{i=\lfloor n/k_u \rfloor}^{n-1} \frac{1}{i} + G \right\rceil.$$

Using

$$\sum_{k=1}^{n} 1/k = \ln(n) + \gamma + \mathcal{O}(1/n) \tag{2.3.53}$$

we get

$$\begin{aligned} X^{(3)} &\overset{\delta}{\geq} \lceil \ln(n - n/k_u) + \gamma + \ln(n) + \gamma - \ln(n/k_u) - \gamma + G + \mathcal{O}(1/n) \rceil \\ &= \left\lceil \ln\left(\frac{n(1-1/k_u)}{n/k_u}\right) + \ln(n) + \gamma + G + \mathcal{O}(1/n) \right\rceil \\ &= \lceil \ln(k_u - 1) + \ln(n) + \gamma + G + \mathcal{O}(1/n) \rceil. \end{aligned} \tag{2.3.54}$$

We continue with the upper bound. Recall (2.3.52); starting from (2.3.51), by applying Corollary 2.3.23, for $G \sim Gumbel_\gamma$ we obtain

$$\begin{aligned} X^{(3)} &\leq \left\lceil \sum_{i=\lceil n/k_l \rceil}^{n-1} \frac{\frac{n}{n-i}}{(1+\mathcal{O}(n^{-0.2}))e(i-cn)} + \sum_{i=\lceil n/k_l \rceil}^{n-1} \frac{D_i}{(1+\mathcal{O}(n^{-0.2}))e(i-cn)} \right\rceil \\ &\overset{\delta}{=} \left\lceil (1+\mathcal{O}(n^{-0.2})) \sum_{i=\lceil n/k_l \rceil}^{n-1} \frac{1}{e(i-cn)(1-i/n)} + G \right\rceil. \end{aligned}$$

Hence, using

$$\frac{1}{e(i-cn)(1-i/n)} = \frac{1}{n-i} + \frac{1}{i-cn},$$

we can infer

$$X^{(3)} \overset{\delta}{\leq} \left\lceil (1+\mathcal{O}(n^{-0.2})) \left(\sum_{i=\lceil n/k_l \rceil}^{n-1} \frac{1}{n-i} + \sum_{i=\lceil n/k_l \rceil}^{n-1} \frac{1}{i-cn} \right) + G \right\rceil.$$

Using index shifts and (2.3.53) yields

$$\begin{aligned} X^{(3)} &\overset{\delta}{\leq} \left\lceil (1+\mathcal{O}(n^{-0.2})) \left(\sum_{i=1}^{n-\lceil n/k_l \rceil} \frac{1}{i} + \sum_{i=\lceil n/k_l \rceil - \lfloor cn \rfloor}^{n-1-\lfloor cn \rfloor} \frac{1}{i} \right) + G + o(1) \right\rceil \\ &\leq \left\lceil \ln(n) + \ln(k_l - 1) - \ln(k_l) + \gamma + \ln\left(\frac{1-c}{1/k_l - c} \right) + G + o(1) \right\rceil. \end{aligned}$$

From (2.3.45) we get $k_l \overset{a\to\infty}{\to} 1$ (uniformly with respect to n) and therefore

$$\ln(k_l) \overset{a\to\infty}{\to} 0 \quad \text{and} \quad \ln\left(\frac{1-c}{1/k_l - c} \right) \overset{a\to\infty}{\to} 0.$$

Hence there is a function $h : \mathbb{N} \to \mathbb{R}^+$ that is independent of n with $h(a) \overset{a\to\infty}{\to} 0$ such that

$$X^{(3)} \overset{\delta}{\leq} \lceil \ln(n) + \ln(k_l - 1) + \gamma + G + h(a) + o(1) \rceil. \tag{2.3.55}$$

In order to refine the lower and upper bounds (2.3.54) and (2.3.55), we consider the terms $\ln(k_u - 1)$ and $\ln(k_l - 1)$ respectively. It is

$$\ln(k_u - 1) = \ln\left(\frac{n}{u_2} - 1 \right) = \ln\left(\frac{1}{f_b(u_1/n)} - 1 \right) = \ln\left(1 - f_b\left(\frac{u_1}{n} \right) \right) - \ln\left(f_b\left(\frac{u_1}{n} \right) \right).$$

According to (2.3.45) we have $\ln(f_b(u_1/n)) \overset{a\to\infty}{\to} 0$ (uniformly with respect to n). For the other summand we have

$$\ln\left(1 - f_b\left(\frac{u_1}{n} \right) \right) = \ln\left(g_b\left(1 - \frac{u_1}{n} \right) \right) = \ln\left(g_b\left(1 - \frac{2^{-a} 2^{\lfloor \log_2(n) \rfloor}}{n} \right) \right) = \ln(g_b(1 - 2^{-a-x(n)})).$$

This, together with (2.3.54), yields a lower bound for $X^{(3)}$; as according to Corollary 2.3.25 we have (again uniformly with respect to n)

$$\ln(k_l - 1) - \ln(k_u - 1) = \ln\left(\frac{n}{l_2} - 1 \right) - \ln\left(\frac{n}{u_2} - 1 \right) \overset{a\to\infty}{\to} 0,$$

with (2.3.55) we also obtain an upper bound; to be specific, there is a function $h_1 : \mathbb{N} \to \mathbb{R}^+$ that is independent of n with $h_1(a) \overset{a\to\infty}{\to} 0$ such that

$$\begin{aligned} X^{(3)} &\overset{\delta}{\geq} \left\lceil \ln(n) + \ln(g_b(1 - 2^{-a-x(n)})) + \gamma + G - h_1(a) + o(1) \right\rceil \quad \text{and} \\ X^{(3)} &\overset{\delta}{\leq} \left\lceil \ln(n) + \ln(g_b(1 - 2^{-a-x(n)})) + \gamma + G + h_1(a) + o(1) \right\rceil. \end{aligned}$$

Hence, combining the three phases according to (2.3.50) and using that $a, b \in \mathbb{N}$ yields, still conditioned on $L := L_1 \cup L_2$,

$$X_n \overset{\delta}{\geq} \lfloor \log_2(n) \rfloor + \lceil \ln(n) - a + b + \ln(g_b(1 - 2^{-a-x(n)})) + \gamma + G - h_1(a) + o(1) \rceil \tag{2.3.56}$$

and

$$X_n \overset{\delta}{\leq} \lfloor \log_2(n) \rfloor + \lceil \ln(n) - a + b + \ln(g_b(1 - 2^{-a-x(n)})) + \gamma + G + h_1(a) + o(1) \rceil. \quad (2.3.57)$$

Lemma 2.3.18 *a*) yields that there is a function $\tilde{h}_2 = \tilde{h}_2^{(a)} : \mathbb{N} \to \mathbb{R}^+$ that is independent of n with $\tilde{h}_2(b) \overset{b\to\infty}{\to} 0$ such that for all $n \in \mathbb{N}$

$$\left| b + \ln(g_b(1 - 2^{-a-x(n)})) - \lim_{\tilde{b}\to\infty, \tilde{b}\in\mathbb{N}} \left(\tilde{b} + \ln(g_{\tilde{b}}(1 - 2^{-a-x(n)})) \right) \right| \leq \tilde{h}_2(b).$$

Recall that $b = b(a)$ is bounded from below by $b^*(a)$ (where $b^*(a)$ is defined as in Lemma 2.3.24) and note that $b^*(a) \overset{a\to\infty}{\to} \infty$. As long as we respect this lower bound, we can choose $b = b(a)$ to meet our requirements. While $\tilde{h}_2$ may depend on a, for any fixed $a \in \mathbb{N}$, we have $\tilde{h}_2^{(a)}(b) \overset{b\to\infty}{\to} 0$. Therefore, as we can choose $b = b(a)$ sufficiently large (depending on a), we can infer that for a suitable choice of $b = b(a)$ there is a function $\hat{h}_2 : \mathbb{N} \to \mathbb{R}^+$ that is independent of n with $\hat{h}_2(a) \overset{a\to\infty}{\to} 0$ such that for all $n \in \mathbb{N}$

$$\left| b + \ln(g_b(1 - 2^{-a-x(n)})) - \lim_{\tilde{b}\to\infty, \tilde{b}\in\mathbb{N}} \left(\tilde{b} + \ln(g_{\tilde{b}}(1 - 2^{-a-x(n)})) \right) \right| \leq \hat{h}_2(a). \quad (2.3.58)$$

The function $c : \mathbb{R} \to \mathbb{R}$ is given by

$$c(x) = \lim_{a\to\infty, a\in\mathbb{N}} \lim_{b\to\infty, b\in\mathbb{N}} -a + b + \ln(g_b(1 - 2^{-a-x}))$$

and, according to Lemma 2.3.18 *b*), it is well-defined, i.e. the limit expression converges. Lemma 2.3.18 *b*), together with Corollary 2.3.21, yields that there is a function $\bar{h}_2 : \mathbb{N} \to \mathbb{R}^+$ that is independent of n (Corollary 2.3.21 guarantees this independence) with $\bar{h}_2 \overset{a\to\infty}{\to} 0$ such that for all $n \in \mathbb{N}$

$$\left| -a + \left(\lim_{\tilde{b}\to\infty, \tilde{b}\in\mathbb{N}} \left(\tilde{b} + \ln(g_{\tilde{b}}(1 - 2^{-a-x(n)})) \right) \right) - c(x(n)) \right| \leq \bar{h}_2(a). \quad (2.3.59)$$

From (2.3.58) and (2.3.59) we can infer that there is a function $\check{h}_2 : \mathbb{N} \to \mathbb{R}^+$ that is independent of n with $\check{h}_2(a) \overset{a\to\infty}{\to} 0$ (for concreteness, $\check{h}_2 := \hat{h}_2 + \bar{h}_2$) such that for all $n \in \mathbb{N}$

$$\left| -a + b + \ln(g_b(1 - 2^{-a-x(n)})) - c(x(n)) \right| \leq \check{h}_2(a).$$

Therefore, using (2.3.56) and (2.3.57), we obtain that there is a function $h_2 : \mathbb{N} \to \mathbb{R}^+$ that is independent of n with $h_2(a) \overset{a\to\infty}{\to} 0$ (for concreteness, $h_2 := h_1 + \check{h}_2$) such that

$$X_n \overset{\delta}{\geq} \lfloor \log_2(n) \rfloor + \lceil \ln(n) + c(x(n)) + \gamma + G - h_2(a) + o(1) \rceil \quad \text{and}$$
$$X_n \overset{\delta}{\leq} \lfloor \log_2(n) \rfloor + \lceil \ln(n) + c(x(n)) + \gamma + G + h_2(a) + o(1) \rceil.$$

Thus we can infer that there are functions $q_1, q_2 : \mathbb{N} \to \mathbb{R}^+$ with $q_1, q_2 = o(1)$ (where, for $i \in \{1, 2\}$, q_i might also depend on a, but for fixed $a \in \mathbb{N}$ it holds $q_i(n) = q_i(n, a) \overset{n\to\infty}{\to} 0$) such that for all $n, k, a \in \mathbb{N}$

$$P_L[X_n \geq k] - P\left[\lfloor \log_2(n) \rfloor + \left\lceil \ln(n) + c(x(n)) + \gamma + G - h_2(a) - q_2(n) \right\rceil \geq k \right] \geq -q_1(n) \quad (2.3.60)$$

and

$$P_L[X_n \geq k] - P\Big[\lfloor \log_2(n) \rfloor + \Big\lceil \ln(n) + c(x(n)) + \gamma + G + h_2(a) + q_2(n) \Big\rceil \geq k\Big] \leq q_1(n). \quad (2.3.61)$$

We claim that this implies that for any $\varepsilon > 0$ there is an $n_0 \in \mathbb{N}$ such that for all $n \in \mathbb{N}$ with $n \geq n_0$, for all $k \in \mathbb{N}$

$$\Big| P_L[X_n \geq k] - P\Big[\lfloor \log_2(n) \rfloor + \Big\lceil \ln(n) + c(x(n)) + \gamma + G \Big\rceil \geq k\Big] \Big| \leq \varepsilon. \quad (2.3.62)$$

To verify this claim, let $\varepsilon > 0$. Let $\tilde{\varepsilon} > 0$ be such that

$$\sup_{x \in [0,1]} \sum_{j=-\infty}^{\infty} P\Big[G \in [x + j - \tilde{\varepsilon}, x + j + \tilde{\varepsilon}]\Big] < \varepsilon/2.$$

It is straightforward to verify that such an $\tilde{\varepsilon}$ exists. Let $a_0 \in \mathbb{N}$ be such that $h_2(a_0) < \tilde{\varepsilon}/2$. Let $n_0 \in \mathbb{N}$ be such that for all $n \in \mathbb{N}$ with $n \geq n_0$

$$q_1(n) = q_1(n, a_0) < \varepsilon/2 \quad (2.3.63)$$

and $q_2(n) = q_2(n, a_0) < \tilde{\varepsilon}/2$. Hence, for $n \geq n_0$, it is $q_2(n) + h_2(a_0) < \tilde{\varepsilon}$. Therefore, for $n \geq n_0$,

$$\begin{aligned} P\Big[\Big[\ln(n) + c(x(n)) + \gamma + G \pm q_2(n) \pm h_2(a_0)\Big] \neq \Big[\ln(n) + c(x(n)) + \gamma + G\Big]\Big] \\ \leq \sup_{x \in [0,1]} \sum_{j=-\infty}^{\infty} P\Big[G \in [x + j - \tilde{\varepsilon}, x + j + \tilde{\varepsilon}]\Big] < \varepsilon/2. \end{aligned}$$

Thus, using (2.3.60), (2.3.61) and (2.3.63), we obtain that for all $n \in \mathbb{N}$ with $n \geq n_0$, for all $k \in \mathbb{N}$ it holds (2.3.62). Hence (recall that $d : \mathbb{N} \to \mathbb{R}$ is defined by $d(n) = \ln(n) + \gamma + c(x(n))$) there is an $\tilde{m} = o(1)$ such that for all $k, n \in \mathbb{N}$

$$|P_L[X_n \geq k] - P[\lfloor \log_2(n) \rfloor + \lceil G + d(n) \rceil \geq k]| \leq \tilde{m}(n)$$

and thus, as $P[L] = 1 + o(1)$, there is an $m = o(1)$ such that for all $k, n \in \mathbb{N}$

$$|P[X_n \geq k] - P[\lfloor \log_2(n) \rfloor + \lceil G + d(n) \rceil \geq k]| \leq m(n).$$

□

2.4 Resilience Results for *Push*

Besides its simplicity and scalability, one other important reason for the popularity of *Push* is that it is often presumed to be a very robust algorithm. In this section we quantify the robustness of *Push* against adversarial edge deletions on graphs with good expansion properties. In particular, we consider the following notion of *local resilience*: Consider an adversary that wants to increase the expected runtime by $\Omega(\ln(n))$ rounds; up to which

fraction of the edges can the adversary be allowed to delete at each node without being able to do this? Our results show that while *Push* is robust against adversarial edge deletions in the sense that it still informs *almost all* nodes as fast as without edge deletions, the number of rounds needed to inform *all* nodes can be increased significantly. We also prove respective results in the presence of independent message transmission failures. To handle the problems of this section accurately we need the following definitions that are also used in [83].

Definition 2.4.1 ($(n, \delta, \Delta, \lambda)$-graph)**.** *Let G be a connected graph with n nodes that has minimum degree δ and maximum degree Δ. Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ be the eigenvalues of the adjacency matrix of G and set $\lambda = \max_{2\leq i\leq n}|\lambda_i| = \max\{|\lambda_2|, |\lambda_n|\}$. We will call G an $(n, \delta, \Delta, \lambda)$-graph.*

Recall Notation 2.1.1; for better readability we repeat some parts of it here.

Notation 2.4.2. *As usual, we are interested in the case where the size n of G gets large, that is, when $n \to \infty$. Hence, unless otherwise specified, all asymptotic notation is with respect to $n \to \infty$; in particular, recall that "whp" (which is short for "with high probability") means with probability $1 + o(1)$ when $n \to \infty$. As in this section we consider Push, for a graph G the runtime $X(G)$ refers to $X(G, Push)$. If we additionally assume that each message transmission fails independently with probability $1 - q \in [0, 1)$, we write $X^{(q)}(G)$ instead of $X(G)$; if we write $X(G)$ this refers to the case $q = 1$. We call q the success probability of a message transmission. To lighten the notation we will write I_t instead of $I_t^{(q)}(G, Push)$. In these cases q will be clear from the context, in particular, I_t does not necessarily refer to the case $q = 1$.*

Definition 2.4.3 (Expander Sequence)**.** *Let $\mathcal{G} = (G_n)_{n\in\mathbb{N}}$ be a sequence of graphs where for each $n \in \mathbb{N}$, the graph G_n is a $(n, \delta_n, \Delta_n, \lambda_n)$-graph. We say that $\mathcal{G}$ is an expander sequence if $\Delta_n/\delta_n = 1 + o(1)$ and $\lambda_n = o(\Delta_n)$.*

If we consider an expander sequence $\mathcal{G} = (G_n)_{n\in\mathbb{N}}$, this always implicitly defines δ_n, Δ_n and λ_n as in Definition 2.4.3. If $\mathcal{G}$ is an expander sequence, then intuitively this means that for n large enough, the edges of G_n are rather uniformly distributed; more formally this is stated in the following variant of the Expander Mixing Lemma.

Lemma 2.4.4 ([83, Corollary 2.4])**.** *Let $\mathcal{G} = (G_n)_{n\in\mathbb{N}} = ((V_n, E_n))_{n\in\mathbb{N}}$ be an expander sequence. Then for $S_n \subseteq V_n$ such that $1 \leq |S_n| \leq n/2$*

$$\left| e(S_n, V_n \backslash S_n) - \frac{\Delta_n |S_n|(n - |S_n|)}{n} \right| = o(\Delta_n)|S_n|.$$

Lemma 2.4.5 is a consequence of Lemma 2.4.4; it assures, for graphs with good expansion properties, that, for $0 < \varepsilon < 1/2$, even if up to a $(1/2 - \varepsilon)$ fraction of the edges at each node is deleted, the graph is still relatively well-connected; in particular, for each set of nodes, a linear fraction of the original number of edges between the set and its complement remains present in the modified graph. We prove Lemma 2.4.5 in Subsection 2.4.2.

Lemma 2.4.5. *Let $\mathcal{G} = (G_n)_{n\in\mathbb{N}} = ((V_n, E_n))_{n\in\mathbb{N}}$ be an expander sequence. Let $0 < \varepsilon < 1/2$. Let $\tilde{\mathcal{G}} = (\tilde{G}_n)_{n\in\mathbb{N}} = ((V_n, \tilde{E}_n))_{n\in\mathbb{N}}$ be such that each $\tilde{G}_n$ is obtained from G_n by deleting edges*

such that each node keeps at least a $(1/2+\varepsilon)$ *fraction of its edges. Then there is an* $\alpha > 0$ *such that the following holds. For each* $n \in \mathbb{N}$ *let* $S_n \subseteq V_n$*, then there is an* $n_0 \in \mathbb{N}$ *such that for all* $n \geq n_0$

$$e_{\tilde{G}_n}(S_n, V_n \backslash S_n) \geq \alpha e_{G_n}(S_n, V_n \backslash S_n).$$

Theorem 2.4.6 provides the runtime of *Push* on graphs with good expansion properties; it turns out to be same as on the complete graph.

Theorem 2.4.6 ([83, Theorem 1.1])**.** *Let* $\mathcal{G} = (G_n)_{n\in\mathbb{N}}$ *be an expander sequence. Then whp*

$$X(G_n) = \log_2(n) + \ln(n) + o(\ln(n)).$$

Completely analogously to the proof of Theorem 2.4.6 given in [83] one obtains the following corresponding result for the setting where each message transmission fails independently with probability $1 - q \in [0, 1)$.

Theorem 2.4.7. *Let* $\mathcal{G} = (G_n)_{n\in\mathbb{N}}$ *be an expander sequence. Let* $q \in (0, 1]$ *be the success probability of a message transmission. Then whp*

$$X^{(q)}(G_n) = \log_{1+q}(n) + \frac{1}{q}\ln(n) + o(\ln(n)).$$

In the remainder of this section we investigate up to how many edges an adversary can be allowed to delete at each node without being able to slow down the information spreading process.

2.4.1 Results

Theorem 2.4.8 states that even if an adversary may delete up to almost half of the edges at each node, whp *almost all* nodes become informed as fast as without the deletions. However, for $0 < \varepsilon < 1/2$, Theorem 2.4.9 states that the adversary can increase the time until *all* nodes are informed significantly by deleting edges, even if he is only allowed to delete up to an ε fraction of the edges at each node.

Theorem 2.4.8. *Let* $0 < \varepsilon < 1/2$*, let* $q \in (0, 1]$ *be the success probability of a message transmission and let* $\mathcal{G} = (G_n)_{n\in\mathbb{N}}$ *be an expander sequence. Let* $\tilde{\mathcal{G}} = (\tilde{G}_n)_{n\in\mathbb{N}}$ *be such that each* $\tilde{G}_n$ *is obtained from* G_n *by deleting edges such that each node keeps at least a* $(1/2+\varepsilon)$ *fraction of its edges. Then, whp, Push informs all but* $n/\ln(n)$ *nodes of* $\tilde{G}_n$ *within*

$$\log_{1+q}(n) + o(\ln(n))$$

rounds.

Theorem 2.4.9. *Let* $0 < \varepsilon < 1/2$ *and let* $q \in (0, 1]$ *be the success probability of a message transmission. There is an expander sequence* $\mathcal{G} = (G_n)_{n\in\mathbb{N}}$ *and a sequence of graphs* $\tilde{\mathcal{G}} = (\tilde{G}_n)_{n\in\mathbb{N}}$ *such that each* $\tilde{G}_n$ *is obtained by deleting edges of* G_n *such that each node keeps at least a* $(1-\varepsilon)$ *fraction of its edges such that whp*

$$X^{(q)}(\tilde{G}_n) \geq \log_{1+q}(n) + \left(1 + \frac{\varepsilon}{2}\right)\frac{1}{q}\ln(n) + o(\ln(n)).$$

2.4.2 Proofs

Proof of Lemma 2.4.5. Without loss of generality we assume that $|S_n| \leq n/2$, otherwise we exchange the roles of S_n and its complement. We note that at most $\Delta_n(1/2-\varepsilon)|S_n|$ edges between S_n and its complement can be deleted. Hence we obtain

$$e_{\tilde{G}_n}(S_n, V_n \backslash S_n) \geq e_G(S_n, V_n \backslash S_n) - \Delta_n \left(1/2 - \varepsilon\right) |S_n|.$$

Using Lemma 2.4.4 we continue with

$$\begin{aligned} e_{G_n}(S_n, V_N \backslash S_n) - \Delta_n \left(\frac{1}{2} - \varepsilon\right) |S_n| &= \frac{\Delta_n |S_n|(n - |S|)}{n} + o(\Delta_n)|S_n| - \Delta_n \left(\frac{1}{2} - \varepsilon\right) |S_n| \\ &= \frac{\Delta_n |S_n|(n - |S_n|)}{n} \left(1 - \frac{n(1/2-\varepsilon)}{n - |S_n|} + \frac{o(\Delta_n)}{\Delta_n} \frac{n}{n - |S_n|}\right). \end{aligned}$$

As $n - |S_n| \geq n/2$ we have $(o(\Delta_n)/\Delta_n)(n/(n-|S_n|)) = o(1)$ and we also obtain that there is a constant $\tilde{\alpha} < 1$ such that $(n(1/2-\varepsilon))/(n-|S_n|) < \tilde{\alpha}$. Hence, for any $0 < \alpha < 1 - \tilde{\alpha}$ and n sufficiently large

$$\frac{\Delta_n |S_n|(n - |S_n|)}{n} \left(1 - \frac{n(1/2-\varepsilon)}{n - |S_n|} + \frac{o(\Delta_n)}{\Delta_n} \frac{n}{n - |S_n|}\right) \geq \alpha e_{G_n}(S_n, V_n \backslash S_n).$$

□

For a shorter notation let us call the setting with deleted edges "new model" and the setting without such edge deletions "old model". We prove Lemma 2.4.10 which directly implies Theorem 2.4.8. To do this we will use Lemmas 2.4.11 and 2.4.12.

Lemma 2.4.10. *Under the assumptions of Theorem 2.4.8 the following holds for the new model.*

a) There are $\tau, \tilde{\tau} = \log_{1+q}(n) + o(\ln(n))$ such that whp $|I_{\tilde{\tau}}| < n/\ln(n) < |I_\tau|$.

b) Assume that $|I_t| \geq n/\ln(n)$. Then there is a $\tau = o(\ln(n))$ such that whp $|I_{t+\tau}| \geq n - n/\ln(n)$.

Note that *a)* and *b)* from Lemma 2.4.10 together directly imply Theorem 2.4.8. Lemma 2.4.11 assures that in the old model and without transmission failures, in the beginning of the information spreading process whp the number of informed nodes essentially doubles.

Lemma 2.4.11 (See the proof of Lemma 2.5 in [83])**.** *Consider the old model. Assume that $|I_t| < n/\ln(n)$ and $q = 1$. Then*

$$P_t[|I_{t+1}| = |I_t| + (1 + o(1))|I_t|] = 1 + o(1). \tag{2.4.1}$$

Lemma 2.4.12 assures that if we already know that the number of informed nodes whp essentially doubles in a round in the setting without transmission failures, then in the setting with transmission failures the number of informed nodes whp increases by a factor of essentially $(1 + q)$.

Lemma 2.4.12. *Consider Push on a sequence of graphs* $(G_n)_{n\in\mathbb{N}} = ((V_n, E_n))_{n\in\mathbb{N}}$ *with* $|V_n| = n$ *such that (2.4.1) holds for* $q = 1$*. Assume that* $|I_t| = \omega(1)$*. Then for* $q \in (0, 1]$

$$P_t[|I_{t+1}| = |I_t| + (1 + o(1))q|I_t|] = 1 + o(1). \tag{2.4.2}$$

Moreover, if there is a $t^* = o(\ln(n))$ *such that whp* $|I_{t^*}| = \omega(1)$*, then there are* $\tau, \tilde{\tau} = \log_{1+q}(n) + o(\ln(n))$ *such that whp*

$$|I_{\tilde{\tau}}| < n/\ln(n) < |I_\tau|. \tag{2.4.3}$$

Proof. For a graph G and for $v \in I_t$ let $\chi_v(G)$ denote the node to which v attempts to push in round t (the push attempt fails with probability $1 - q$). Let

$$N_{t+1} := \{\chi_v(G_n) \mid v \in I_t\} \cap U_t.$$

Note that whp $|N_{t+1}| = (1 + o(1))|I_t|$ from (2.4.1). Each node in N_{t+1} has a probability of at least q to get informed and all these events are independent; thus the first statement follows directly by applying the Chernoff bound. Now we prove the second statement. We call a round t that does not satisfy (2.4.2) a failed round. Due to the first statement by applying the Chernoff bound we obtain the existence of a $h = o(\ln(n))$ such that whp less than $h(n)$ of the first $\log_{1+q}(n) + h(n)$ rounds fail; this implies the second statement. □

Proof of Lemma 2.4.10. We first show claim *a*). We start with verifying by contradiction that in the old model for $|I_t| \leq n/4$ we have

$$P_t\left[|I_{t+1}| > \left(1 + \frac{q}{4}\right)|I_t|\right] \geq \frac{q}{4}. \tag{2.4.4}$$

Assume that this is not the case, i.e. assume that $P_t[|I_{t+1}\backslash I_t| > q|I_t|/4] < q/4$. Since $|I_{t+1}\backslash I_t| \leq |I_t|$, we get

$$\mathbb{E}_t[|I_{t+1}\backslash I_t|] \leq P_t\left[|I_{t+1}\backslash I_t| > \frac{q}{4}|I_t|\right]|I_t| + \frac{q}{4}|I_t| < \frac{q}{4}|I_t| + \frac{q}{4}|I_t| = \frac{q}{2}|I_t|. \tag{2.4.5}$$

Let χ_v and N_{t+1} be defined as in the proof of Lemma 2.4.12. From (2.4.1) we know that whp $|N_{t+1}| = (1 + o(1))|I_t|$, hence for $q \in (0, 1]$ we have $E_t[|I_{t+1}\backslash I_t|] \geq (1 + o(1))q|I_t|$ which contradicts (2.4.5) and therefore (2.4.4) holds. Let us call a round t with $|I_{t+1}| > (1 + q/4)|I_t|$ a successful round. If we make $R \in \mathbb{N}$ rounds, then, according to (2.4.4), the number of successful rounds can be bounded from below by a binomial random variable $Bin(R, q/4)$. In particular, applying the Chernoff bound yields that, whp, if we make $R = \ln(\ln(n))$ rounds, $(1 + o(1))Rq/4$ of these rounds are successful. Thus, whp, after $\ln(\ln(n))$ rounds, $\omega(1)$ nodes are informed. Hence from now on we assume that $|I_t| = \omega(1)$. We assume $q = 1$ and prove that (2.4.1) also holds in the new model; then, according to Lemma 2.4.12, claim *a*) follows.

Let $G = (V, E)$ be a graph; for $u \in V$ let $c_u(G) := |\{v \in I_t \mid \chi_v(G) = u\}|$ denote the number of times u is pushed in round t. Let

$$\mathcal{Y}_t(G) := \{v \in I_t \mid c_v(G) = 1\} \quad \text{and} \quad \mathcal{H}_t(G) := \{v \in I_t \mid c_v(G) \geq 1\}$$

denote the set of informed nodes that are pushed exactly once in round t and the set of informed nodes that are pushed at least once in round t respectively. Let

$$\mathcal{Z}_t(G) := \{v \in V \mid c_v(G) \geq 2\}$$

denote the set of nodes that are pushed more than once in round t. Set $Y_t(G) := |\mathcal{Y}_t(G)|$ and $H_t(G) := |\mathcal{H}_t(G)|$ and, in slight abuse of notation, let $Z_t(G) := \sum_{k\geq 2}(k-1) \cdot |\{v \in V \mid c_v(G) = k\}|$ denote the number of nodes that are pushed multiple times in round t counted with multiplicity. We want to show that (2.4.1) does hold in the new model; for contradiction we assume that this is not the case. Hence we can infer that there is a constant $c > 0$ such that

$$\limsup_{n\to\infty} P_t[Y_t(\tilde{G}_n) \geq c|I_t|] > 0 \quad \text{or} \quad \limsup_{n\to\infty} P_t[Z_t(\tilde{G}_n) \geq c|I_t|] > 0.$$

Thus, w.l.o.g., we can assume that there is $f^* > 0$ and $n_0 \in \mathbb{N}$ such that

$$P_t[Y_t(\tilde{G}_n) \geq c|I_t|] > f^* \text{ for all } n \geq n_0 \quad \text{or} \quad P_t[Z_t(\tilde{G}_n) \geq c|I_t|] > f^* \text{ for all } n \geq n_0,$$

if this is not the case we can restrict ourselves to a suitable subsequence of $(n)_{n\in\mathbb{N}}$ on which it is true. We couple the new and the old model: For any node v consider $\chi_v(G_n)$. If $\chi_v(G_n) \in N_{\tilde{G}_n}(v)$, set $\chi_v(\tilde{G}_n) := \chi_v(G_n)$ otherwise choose $\chi_v(\tilde{G}_n)$ uniformly at random from $N_{\tilde{G}_n}(v)$. The marginal distributions are correct by construction.

We start with the case that $P_t[Y_t(\tilde{G}_n) \geq c|I_t|] > f^*$. We consider $w \in \mathcal{Y}_t(\tilde{G}_n)$. This means that there is a unique node $v \in N_{\tilde{G}_n}(w) \cap I_t$ with $\chi_v(\tilde{G}_n) = w$. We obtain

$$\begin{aligned} P_t[w \in \mathcal{H}_t(G_n) \mid \chi_v(\tilde{G}_n) = w] &\geq P_t[\chi_v(G_n) = \chi_v(\tilde{G}_n) \mid \chi_v(\tilde{G}_n) = w] \\ &= P_t[\chi_v(G_n) = \chi_v(\tilde{G}_n)] \geq 1/2. \end{aligned}$$

As this holds independently for all $w \in \mathcal{Y}_t(\tilde{G}_n)$ we can infer that $P_t[H_t(G_n) \geq c/2|I_t|] \geq f^*/2$ which contradicts Lemma 2.4.11.

We continue with the case that $P_t[Z_t(\tilde{G}_n) \geq c|I_t|] > f^*$. Consider $w \in \mathcal{Z}_t(\tilde{G}_n)$, i.e. $k := c_w(\tilde{G}_n) \geq 2$. In particular, there are nodes $v_1, \ldots, v_k$ with $\chi_{v_i}(\tilde{G}_n) = w$ for $i \in \{1, \ldots, k\}$. Like in the previous case for each v_i independently $P_t[\chi_{v_i}(G_n) = \chi_{v_i}(\tilde{G}_n) \mid \chi_{v_i}(\tilde{G}_n) = w] \geq 1/2$. Intuitively this means that for each push to w in round t in the new model, with probability at least $1/2$ the same push was also made in the old model; in particular, for $k = 2$ with probability $1/4$ both pushes have also been made in the old model and for $k \geq 3$ with probability at least $1/2$ at least $\lceil k/2 \rceil$ pushes were the same in the old model. As this holds independently for all $w \in Z_t(\tilde{G}_n)$ we obtain $P_t[Z_t(G_n) \geq c/3|I_t|] \geq f^*/4$ which contradicts Lemma 2.4.11.

Next we prove claim *b*). This works analogously to Phase (II) in the proof of Lemma 2.5 in [83]. Yet, for completeness, we include the details. We assume that $|I_t| \in [n/\ln(n), n - n/\ln(n)]$. We further divide this phase into two cases, namely $|I_t| \in [n/\ln(n), n/2]$ and $|I_t| \in [n/2, n - n/\ln(n)]$. We start with the first case, i.e. $|I_t| \in [n/\ln(n), n/2]$. Using Lemmas 2.4.4 and 2.4.5 and the assumption that $\Delta_n/\delta_n = 1 + o(1)$ we obtain that there is an $\alpha > 0$ such that, for n sufficiently large,

$$e(I_t, U_t) > \alpha\delta_n|I_t|. \tag{2.4.6}$$

Now, using Fact 2.3.3, we bound the expected number of nodes that become informed in round t from below.

$$E_t[|I_{t+1}\backslash I_t|] \geq \sum_{u\in N(I_t)\backslash I_t}\left[1-\prod_{v\in N(u)\cap I_t}\left(1-\frac{q}{\Delta_n}\right)\right] \geq \sum_{u\in N(I_t)\backslash I_t} 1-e^{-|N(u)\cap I_t|q/\Delta_n}.$$

Using the fact that for any $x\in(0,1)$ it holds $e^{-x}\leq 1-x/2$ and (2.4.6) we continue with

$$E_t[|I_{t+1}\backslash I_t|] \geq \sum_{u\in N(I_t)\backslash I_t}\frac{q|N(u)\cap I_t|}{2\Delta_n} = \frac{qe(I_t,V\backslash I_t)}{2\Delta_n} \geq \frac{\alpha q\delta_n}{2\Delta_n}|I_t|.$$

Since $|I_{t+1}|\leq 2|I_t|$, we obtain as long as $|I_t|\leq n/2$ that there are constants $\beta,\gamma>0$ so that

$$P_t[|I_{t+1}|\geq(1+\gamma)|I_t|]\geq\beta. \tag{2.4.7}$$

Similarly to before, now we call a round t successful if $|I_{t+1}|\geq(1+\gamma)|I_t|$ and analogously to before, if we make $R\in\mathbb{N}$ rounds, then according to (2.4.7) the number of successful rounds can be bounded from below by a binomial random variable $Bin(R,\beta)$. Therefore, by applying the Chernoff bound, we can infer that if we make $R=\ln(n)/\ln(\ln(n))$ rounds, whp at least $(1+o(1))\beta R$ of these rounds are successful. Thus, if at least $n/\ln(n)$ nodes are informed and we make $\ln(n)/\ln(\ln(n))$ additional rounds, then afterwards whp at least $n/2$ nodes are informed.

Now we consider the case $|I_t|\in[n/2,n-n/\ln(n)]$. In this case, we consider the shrinking of U_t. Again, as $|U_t|\leq n/2$, using Lemmas 2.4.4 and 2.4.5 we obtain that there is an $\alpha>0$ such that, for n sufficiently large, $e(I_t,U_t)>\alpha\delta_n|U_t|$. Hence, again using Fact 2.3.3 and that for any $x\in(0,1)$ it holds $e^{-x}\leq 1-x/2$, we obtain

$$\begin{aligned}E_t[|U_{t+1}|] &= \sum_{u\in U_t}\prod_{v\in N(u)\cap I_t}\left(1-\frac{q}{d(v)}\right) \leq \sum_{u\in U_t} e^{-|N(u)\cap I_t|q/\Delta_n}\\ &\leq \sum_{u\in U_t} 1-\frac{q|N(u)\cap I_t|}{2\Delta_n} \leq |U_t|-\frac{\alpha q\delta_n}{2\Delta_n}|U_t| = \left(1-\frac{\alpha q\delta_n}{2\Delta_n}\right)|U_t|.\end{aligned}$$

A simple inductive argument gives

$$E_t[|U_{t+\tau}|]\leq\left(1-\frac{\alpha q\delta_n}{2\Delta_n}\right)^{\tau}|U_t|,\quad \tau\in\mathbb{N}.$$

Thus, for $\tau:=-2\ln(\ln(n))/\ln(1-\alpha q\delta_n/(2\Delta_n))=o(\ln(n))$ we have $E_t[|U_{t+\tau}|]=o(n/\ln(n))$. Hence, by Markov's inequality, $P_t[|U_{t+\tau}|\geq n/\ln(n)]=o(1)$. □

Proof of Theorem 2.4.9. We will construct an explicit example of a graph that has the desired property. In particular, for any $0<\varepsilon<1/2$, each $q\in(0,1]$ and $n\in\mathbb{N}$ we will define a graph $G=G_n$ that is obtained by deleting edges from the complete graph on n nodes such that each node keeps at least a $(1-\varepsilon)$ fraction of its edges and such that *Push* slows down by terms of logarithmic order. We define a graph $G=(V_1\cup V_2,E)$ with node

set $V = V_1 \cup V_2$, where $V_1 := \{1, \ldots, \lfloor n/2 \rfloor\}$ and $V_2 := \{\lfloor n/2 \rfloor + 1, \ldots, n\}$, as follows. We include in E all pairs of nodes that intersect V_1 and moreover, we add edges (that now have endpoints only in V_2) such that all nodes in V_2 have degree $\lceil (1-\varepsilon)n \rceil + 1 \pm 1$. Let $U_t^{(2)} := U_t \cap V_2$. According to Lemma 2.4.10 *a)* there is a $t = \log_{1+q}(n) + o(\log n)$ such that whp $|I_t| < n/\ln(n)$. Hence we can assume that $|I_t| < n/\ln(n)$ and it suffices to show that it takes whp at least $(1+\varepsilon/2)q^{-1}\ln(n) + o(\ln(n))$ rounds to inform all remaining nodes. As $|I_t| < n/\ln(n)$ we have $|U_t^{(2)}| \geq n/4$. To show that *Push* has indeed slowed down, we use an argument that is similar to the one used in the proof of the lower bound for Theorem 1.1 in [83]. In the remainder of this proof we will consider a modified process in which nodes have a higher chance of getting informed; in particular, we assume that in each round, *every* node chooses a neighbour independently and uniformly at random and after this round the chosen nodes are informed. The runtime in this modified model constitutes a lower bound for the runtime of *Push*. Let E_u denote the event that $u \in U_t$ does not get informed within the next $\tau := (1+\varepsilon/2)q^{-1}\ln(n)$ rounds in the modified model and let $\overline{E_u}$ denote the complementary event. It suffices to show that

$$P_t\Big[\bigwedge_{u \in U_t^{(2)}} \overline{E_u}\Big] = o(1).$$

Each node $u \in U_t^{(2)}$ has $\lfloor n/2 \rfloor$ neighbours that have degree $n-1$, at most $(1-\varepsilon)n+3$ neighbours that have at least degree $(1-\varepsilon)n$ and no further neighbours. Therefore, using Fact 2.3.3, we obtain for each $u \in U_t^{(2)}$

$$\begin{aligned} P_t[E_u] &\geq \left(\left(1-\frac{q}{n-1}\right)^{n/2}\left(1-\frac{q}{(1-\varepsilon)\,n}\right)^{(1/2-\varepsilon)n+3}\right)^{\tau} = (1+o(1))\left(e^{-q\left(1/2+\frac{1/2-\varepsilon}{1-\varepsilon}\right)}\right)^{\tau} \\ &= (1+o(1))e^{-\frac{4-4\varepsilon-3\varepsilon^2}{4-4\varepsilon}\ln(n)} = \omega(n^{-1}). \end{aligned}$$

We observe

$$\sum_{u \in U_t^{(2)}} P_t[E_u] \geq n/4\omega(n^{-1}) = \omega(1).$$

As we assume that uninformed nodes can obtain the piece of information not only from informed neighbours but from all their neighbours, the events $\{\overline{E_u} \mid u \in U_t^{(2)}\}$ are negatively correlated in the sense that for all $v, u_1, ..., u_k \in U_t^{(2)}$ it holds $P_t[\overline{E_v}] \geq P_t[\overline{E_v} \mid \overline{E_{u_1}} \wedge \ldots \wedge \overline{E_{u_k}}]$ ([93]). Thus, using that for all $x \in \mathbb{R}$ it holds $1 - x \leq e^{-x}$, we arrive at

$$P_t\Big[\bigwedge_{u \in U_t^{(2)}} \overline{E_u}\Big] \leq \prod_{u \in U_t^{(2)}} P_t[\overline{E_u}] = \prod_{u \in U_t^{(2)}} (1 - P_t[E_u]) \leq \exp\left(-\sum_{u \in U_t^{(2)}} P_t[E_u]\right) = o(1).$$

□

2.5 Information Spreading on Random Evolving Graphs

2.5.1 Introduction

Recently Clementi et al. have investigated *Push* on random evolving graphs ([21]), i.e. in a setting where the underlying graph is not fixed but changes over time; this is motivated by the fact that often real-world networks are not static; compare also the paragraph "Evolving Graphs" in Subsection 2.2.2. One setting they treated is the following: Each round the underlying graph is a newly (and independently of the previous graphs) sampled Erdős-Rényi random graph $G(n,p)$. We are interested in large graphs, thus, as usual, all asymptotic notation is with respect to $n \to \infty$ if not explicitly stated differently; in particular, recall that "whp" (which is short for "with high probability") means with probability $1+o(1)$ when $n \to \infty$. Among other results, in [21] it is shown that if $p \geq 1/n$ then whp the runtime of *Push* is $\mathcal{O}(\ln(n))$.

For $a > 0$ and $p = a/n$, Doerr and Kostrygin have improved this bound ([34, 70]). According to [70], $p = a/n$ is the most interesting regime because a respective random graph whp is not connected but has nodes with degrees varying between 0 and $\Theta(\ln(n)/\ln(\ln(n)))$ and, for $\varepsilon > 0$ if $p \geq (1+\varepsilon)/n$, whp a giant component containing a linear fraction of the nodes exists. They have shown that the expected runtime of *Push* is $E[X_n] = \log_{2-e^{-a}}(n) + 1/(1-e^{-a})\ln(n) + \mathcal{O}(1)$; moreover, it is shown that constants $\alpha, A > 0$ exist such that for all $r, n \in \mathbb{N}$ for the runtime X_n it holds $P[|X_n - E[X_n]| \geq r] \leq A\exp(-\alpha r)$. This was shown by applying a general framework developed in [34]. This framework exploits that many information spreading algorithms are sufficiently characterised by the probability p_k of a node to become informed in a round that starts with k informed nodes and a bound on the covariances between the indicator variables each indicating whether an uninformed node becomes informed in that round. By bounding p_k and these covariances, the framework allows to obtain the expected runtime up to constant additive terms as well as large deviation bounds.

We use this framework to investigate *Pull* and *Push&Pull* on random evolving graphs. We show that the expected runtime of *Pull* in the setting described above (i.e. each round a new Erdős-Rényi random graph $G(n,a/n)$ is sampled independently of what happened before) is $\log_{2-e^{-a}}(n) + 1/a\ln(n) + \mathcal{O}(1)$. Let $\kappa = 2(1-e^{-a}) - (1-e^{-a})^2/a$; we prove that the expected runtime of *Push&Pull* is $\log_{1+\kappa}(n) + 1/a\ln(n) + \mathcal{O}(1)$. As a byproduct, we also obtain large deviation bounds.

For *Push&Pull* we will observe that while both, *Push* and *Pull*, need logarithmic time for the last phase of the information spreading process, when combining them in *Push&Pull*, *Push* becomes useless in the last phase which might be unexpected. Another interesting aspect is that in the first phase, when almost no nodes are informed, *Push* and *Pull* get in each other's way in the sense that many nodes obtain the piece of information by a push as well as by a pull operation which makes one of both operations useless. In Remark 2.5.15 we provide an explanation for these observations.

Note that we have made the contributions of this section also available in [23].

Notation 2.5.1. *Recall that X_n denotes the runtime of an information spreading algorithm on a graph with n nodes if the remaining information (i.e. the underlying graph etc.) is clear from the context. In this section we will use a slightly more general notation: Let*

$c, d \in \{1, 2, \ldots, n\}$ with $c < d$. Then $X_n(c, d)$ denotes the number of rounds that are needed if one starts with c informed nodes until at least d nodes are informed. When we consider a sequence of random graphs, then we implicitly mean that the graphs of the sequence are sampled independently of each other.

2.5.2 Preliminaries

We start with stating the framework from [34]. We consider homogeneous information spreading processes characterised as follows: We consider graphs with n nodes, in the beginning one node is informed, the other nodes are uninformed. Once a node is informed, it remains informed. The process is partitioned into rounds, in each round each uninformed node can become informed. Whenever a round starts with k informed nodes, we assume that there is a $p_k \in (0, 1)$ (only depending on k) such that each uninformed node becomes informed in that round with probability p_k; thus p_k is called the success probability. An information spreading process as described is called homogeneous ([34]). By suitably bounding the success probabilities and the covariance numbers defined as follows, bounds on the runtime can be obtained. Whenever we write k, this refers to the number of informed nodes the current round of the information spreading process starts with; we set $u := n - k$.

Definition 2.5.2 (Covariance numbers, [34])**.** *For a given homogeneous information spreading process and $k \in \{1, \ldots, n-1\}$ let c_k be the smallest number such that, whenever a round starts with k informed nodes, for any two uninformed nodes x, y, the indicator random variables X, Y for the events that x or y respectively becomes informed in this round satisfy $Cov[X, Y] \leq c_k$.*

If the exponential growth conditions given by Definition 2.5.3 are fulfilled, then Theorem 2.5.4 states that there is an exponential growth phase, i.e. if sufficiently few nodes are informed, then the number of informed nodes essentially increases by a constant factor each round and the runtime can be bounded respectively.

Definition 2.5.3 (Exponential growth conditions, [34])**.** *Let κ_n be bounded between two positive constants. Let $a, b, c \geq 0$ and $0 < f < 1$. We say that a homogeneous information spreading process satisfies the upper (respectively lower) exponential growth conditions in $[1, fn[$ if there is an $n_0 \in \mathbb{N}$ such that for all $n \in \mathbb{N}$ with $n \geq n_0$ the following properties are satisfied for any $k < fn$:*

- $p_k \geq \kappa_n \frac{k}{n}\left(1 - a\frac{k}{n} - \frac{b}{\ln(n)}\right)$ (*respectively* $p_k \leq \kappa_n \frac{k}{n}\left(1 + a\frac{k}{n} + \frac{b}{\ln(n)}\right)$).
- $c_k \leq c\frac{k}{n^2}$.

In the case of the upper exponential growth condition, we also require $af < 1$.

Theorem 2.5.4 ([34])**.** *If a homogeneous information spreading process satisfies the upper (lower) exponential growth conditions in $[1, fn[$, then there are constants $A, \alpha > 0$ such that*

$$E[X_n(1, fn)] \underset{(\geq)}{\leq} \log_{1+\kappa_n}(n) + \mathcal{O}(1) \quad \text{and}$$

$$P[X_n(1, fn) \underset{(\leq)}{\geq} \log_{1+\kappa_n}(n) \underset{(-)}{+} r] \leq A\exp(-\alpha r) \text{ for all } r, n \in \mathbb{N}.$$

When the lower exponential growth conditions are satisfied, then also there is an $f < f' < 1$ such that with probability $1 + \mathcal{O}(1/n)$ at most $f'n$ nodes are informed at the end of round $X_n(1, fn)$.

If the exponential shrinking conditions given by Definition 2.5.5 are fulfilled, then Theorem 2.5.6 states that there is an exponential shrinking phase, i.e. if sufficiently many nodes are informed, then the number of uninformed nodes essentially decreases by a constant factor each round and the runtime can be bounded respectively.

Definition 2.5.5 (Exponential shrinking conditions, [34])**.** *Let ρ_n be bounded between two positive constants. Let $0 < g < 1$, and $a, c \in \mathbb{R}_0^+$. We say that a homogeneous information spreading process satisfies the upper (respectively lower) exponential shrinking conditions if there is an $n_0 \in \mathbb{N}$ such that for all $n \in \mathbb{N}$ with $n \geq n_0$ the following properties are satisfied for all $u = n - k \leq gn$:*

- $1 - p_k = 1 - p_{n-u} \leq e^{-\rho_n} + a\frac{u}{n}$ *(respectively* $1 - p_k = 1 - p_{n-u} \geq e^{-\rho_n} - a\frac{u}{n}$*).*
- $c_k = c_{n-u} \leq \frac{c}{u}$.

For the upper exponential shrinking conditions, we also assume that $e^{-\rho_n} + ag < 1$.

Theorem 2.5.6 ([34])**.** *If a homogeneous information spreading process satisfies the upper (lower) exponential shrinking conditions, then there are $A, \alpha > 0$ such that*

$$E[X_n(n - \lfloor gn \rfloor, n)] \underset{(\geq)}{\leq} \frac{1}{\rho_n} \ln(n) + \mathcal{O}(1) \quad \text{and}$$

$$P\left[X_n(n - \lfloor gn \rfloor, n) \underset{(\leq)}{\geq} \frac{1}{\rho_n} \ln(n) \underset{(-)}{+} r\right] \leq A \exp(-\alpha r) \text{ for all } r, n \in \mathbb{N}.$$

Remark 2.5.7. *It suffices to compute κ_n and ρ_n from Theorems 2.5.4 and 2.5.6 respectively up to additive $\mathcal{O}(1/\ln(n))$ terms; then the conclusions of Theorems 2.5.4 and 2.5.6 remain the same.*

We will use the following well-known fact in our proofs; it is a direct consequence of Fact 2.3.3.

Fact 2.5.8. *Let $a > 0$. Consider an Erdős-Rényi random graph $G = G(n, a/n)$. The probability that a specific node is isolated is $e^{-a} + \mathcal{O}(1/n)$.*

Theorem 2.5.9 considers the number of rounds *Push* needs in the described setting. While we do not need it for the proofs of our results, we state it for completeness.

Theorem 2.5.9 ([34])**.** *Let $a > 0$ and assume that each round a newly sampled Erdős-Rényi random graph $G(n, a/n)$ is the underlying graph. Then for the runtime of Push, X_n, we have*

$$E[X_n] = \log_{2-e^{-a}}(n) + \frac{1}{1 - e^{-a}} \ln(n) + \mathcal{O}(1)$$

and there are constants $A, \alpha > 0$ such that for all $r, n \in \mathbb{N}$

$$P[|X_n - E[X_n]| \geq r] \leq A \exp(-\alpha r).$$

In [34] it is observed that the obtained runtime is the same as if the underlying graph is a complete graph but message transmissions fail independently with probability e^{-a} which, up to additive $\mathcal{O}(1/n)$ terms, is the probability that a node is isolated. We will see that this also holds for *Pull*. Interestingly it does not hold for *Push&Pull*; we provide an explanation in Remark 2.5.15.

2.5.3 *Pull* on Random Evolving Graphs

We prove the following theorem that provides the expected runtime of *Pull* up to a constant number of rounds as well as large deviation bounds where in each round the underlying graph is a newly sampled Erdős-Rényi random graph.

Theorem 2.5.10. *Let $a > 0$ and assume that each round a newly sampled Erdős-Rényi random graph $G(n, a/n)$ is the underlying graph. Then for the runtime of Pull, X_n, we have*

$$E[X_n] = \log_{2-e^{-a}}(n) + \frac{1}{a}\ln(n) + \mathcal{O}(1)$$

and there are constants $A, \alpha > 0$ such that for all $r, n \in \mathbb{N}$

$$P[|X_n - E[X_n]| \geq r] \leq A\exp(-\alpha r).$$

Proof. We want to apply the framework from [34]. In order to do this, we consider a round of the information spreading process that starts with k informed and $u = n - k$ uninformed nodes; we will refer to this round as the *current round*. We can assume that at the start of each round, the edges of the Erdős-Rényi random graph $G(n, a/n)$ are not yet sampled. Before the random graph is sampled, each uninformed node has the same probability of getting informed, hence the information spreading process is homogeneous. First we consider the covariance numbers. To do this, consider two uninformed nodes x and y and let X and Y denote the indicator random variables indicating whether x or y respectively gets informed in this round. Note that, as the edges are not yet sampled, there is some positive correlation between X and Y: If we condition on the event that the uninformed node x becomes informed, then it is slightly less likely that x and the uninformed node y are neighbours; this increases the probability that y has a higher fraction of informed neighbours and therefore y pulls the information more likely. However, the framework from [34] allows for some positive correlation. We will bound the covariance accordingly. Let $\mathsf{X} :=$ "$X = 1$" and $\mathsf{Y} :=$ "$Y = 1$" denote the events that x or y respectively becomes informed in the current round. Let $E(G)$ denote the edge set of the random graph for the current round; let $\mathsf{E} :=$ "$\{x, y\} \in E(G)$" denote the event that x and y become neighbours in the current round. It is

$$Cov(X, Y) = P[\mathsf{X} \cap \mathsf{Y}] - P[\mathsf{X}]P[\mathsf{Y}] = P[\mathsf{X}]P[\mathsf{Y} \mid \mathsf{X}] - P[\mathsf{X}]P[\mathsf{Y}] = P[\mathsf{X}](P[\mathsf{Y} \mid \mathsf{X}] - P[\mathsf{Y}]).$$

We have

$$P[\mathsf{Y} \mid \mathsf{X}] \leq P[\mathsf{Y} \mid \neg\mathsf{E}] = \frac{P[\mathsf{Y} \cap \neg\mathsf{E}]}{P[\neg\mathsf{E}]} \leq \frac{P[\mathsf{Y}]}{P[\neg\mathsf{E}]} = \frac{P[\mathsf{Y}]}{1 - a/n} = P[Y] + \mathcal{O}(1/n)$$

and hence

$$Cov(X,Y) \leq P[\mathsf{X}]\mathcal{O}(1/n) \leq \frac{k}{n}\mathcal{O}(1/n).$$

Therefore the covariance conditions are fulfilled for the exponential growth and shrinking conditions.

Next we estimate the probability p_k for an uninformed node to become informed in a round starting with k informed nodes. If an uninformed node has a neighbour, i.e. if it is not isolated, then with probability $k/(n-1)$ it becomes informed. However, if it is isolated, which according to Fact 2.5.8 is the case with probability $e^{-a}+\mathcal{O}(1/n)$, the node does not become informed in this round deterministically. Thus $p_k = (1-e^{-a}+\mathcal{O}(1/n))k/n$. Hence both, upper and lower, exponential growth conditions are fulfilled for an arbitrary $0<f<1$ with $\kappa_n = 1-e^{-a}+\mathcal{O}(1/n)$. Recall that according to Remark 2.5.7, the $\mathcal{O}(1/n)$ term is negligible. Theorem 2.5.4 therefore yields

$$E[X_n(1,fn)] = \log_{2-e^{-a}}(n)+\mathcal{O}(1)$$

and that there are $A_1, \alpha_1 > 0$ such that for all $r, n \in \mathbb{N}$

$$P[|X_n(1,fn) - \log_{2-e^{-a}}(n)| \geq r] \leq A_1 \exp(-\alpha_1 r).$$

Moreover, it yields that there is an $f < f' < 1$ such that with probability $1+\mathcal{O}(1/n)$ at most $f'n$ nodes are informed at the end of round $X_n(1,fn)$.

Now, for the exponential shrinking conditions, we consider $1-p_k = 1-p_{n-u}$. We have

$$1-p_{n-u} = 1-\frac{n-u}{n-1}(1-e^{-a}+\mathcal{O}(1/n)) = e^{-a}+(1-e^{-a})\frac{u}{n}+\mathcal{O}(1/n).$$

Hence the upper and lower exponential shrinking conditions are fulfilled with $\rho_n = a+\mathcal{O}(1/n)$ (because $e^{-a}+\mathcal{O}(1/n) = e^{-a+\mathcal{O}(1/n)}$) for an arbitrary $0<g<1$. Note that according to Remark 2.5.7, the term $\mathcal{O}(1/n)$ is negligible. Theorem 2.5.6 therefore yields

$$E[X_n(n-\lfloor gn \rfloor, n)] = \frac{1}{a}\ln(n)+\mathcal{O}(1)$$

and that there are $A_2, \alpha_2 > 0$ such that for all $r, n \in \mathbb{N}$

$$P\left[\left|X_n(n-\lfloor gn\rfloor, n) - \frac{1}{a}\ln(n)\right| \geq r\right] \leq A_2 \exp(-\alpha_2 r).$$

Thus, considering the exponential growth phase and the exponential shrinking phase together, we obtain the claim. □

If the underlying graph is the complete graph on n nodes and each message transmission fails independently with probability e^{-a} (which, up to an additive $\mathcal{O}(1/n)$ term, is the probability that a specific node is isolated in an Erdős-Rényi random graph $G(n, a/n)$), then the expected runtime of *Pull* is $\log_{2-e^{-a}}(n) + 1/a\ln(n) + \mathcal{O}(1)$ ([34]). As we have seen, this is the same as in the random evolving graph setting that we investigated. An analogous observation was made for *Push* in [34].

2.5.4 *Push&Pull* on Random Evolving Graphs

We prove the following theorem that provides the expected runtime of *Push&Pull* up to a constant number of rounds as well as large deviation bounds where in each round the underlying graph is a newly sampled Erdős-Rényi random graph.

Theorem 2.5.11. *Let $a > 0$ and assume that each round a newly sampled Erdős-Rényi random graph $G(n, a/n)$ is the underlying graph. Let $\kappa := 2(1 - e^{-a}) - (1 - e^{-a})^2/a$. Then for the runtime of Push&Pull, X_n, we have*

$$E[X_n] = \log_{1+\kappa}(n) + \frac{1}{a}\ln(n) + \mathcal{O}(1)$$

and there are constants $A, \alpha > 0$ such that for all $r, n \in \mathbb{N}$

$$P[|X_n - E[X_n]| \geq r] \leq A\exp(-\alpha r).$$

Before we prove Theorem 2.5.11, we introduce some notation.

Notation 2.5.12. *Consider an uninformed node y at the beginning of a round that starts with $k = \mu n$ informed nodes; we will refer to this round as the* current round. *Let PH_y denote the event that y is pushed by an informed node in the current round. Analogously let PL_y denote the event that y pulls the piece of information in the current round from an informed node. Further set $\mathsf{PP}_y = \mathsf{PH}_y \cup \mathsf{PL}_y$, i.e. PP_y denotes the event that y is pushed or pulls the piece of information in the current round. For $j \in \{0, 1, 2, \ldots, k\}$ let $\mathsf{INF}_y(j)$ denote the event that y has exactly j informed neighbours $x_1, \ldots, x_j$. When we write $\mathsf{INF}_y(j)$ this implicitly defines $x_1, \ldots, x_j$. Let x be an informed node; let $\mathsf{PH}_y(x)$ denote the event that y is pushed by x in the current round. Similarly, let $\mathsf{PL}_y(x)$ denote the event that y pulls the information from x in the current round. When an index is clear from the context, it may be omitted.*

We will use Lemma 2.5.13 to prove Theorem 2.5.11; it quantifies the probability that an uninformed node pulls the information in the current round conditioned on the event that it gets also pushed by an informed node.

Lemma 2.5.13. *Let $a > 0$ and assume that each round a newly sampled Erdős-Rényi random graph $G(n, a/n)$ is the underlying graph. Consider a round that starts with k informed nodes and set $\mu := k/n$. Assume that the edges are not yet sampled. Let y be an uninformed node. Then*

$$P[\mathsf{PL}_y \mid \mathsf{PH}_y] = \frac{1 - e^{-a}}{a} + \mathcal{O}(\mu) \text{ for } \mu \to 0.$$

In order to prove Lemma 2.5.13 we will use Lemma 2.5.14 that provides a closed form for a certain sum.

Lemma 2.5.14. *Let $n \in \mathbb{N}$, $\mu \in (0, 1)$ with $\mu n \in \mathbb{N}$ and let $a \in \mathbb{R}^+$. Then*

$$\sum_{i=0}^{(1-\mu)n-1} \binom{(1-\mu)n-1}{i} \left(\frac{a}{n}\right)^i \left(1 - \frac{a}{n}\right)^{(1-\mu)n-1-i} \frac{1}{i+1} = \frac{1 - (1 - \frac{a}{n})^{(1-\mu)n}}{a(1-\mu)}.$$

Proof. It is

$$\begin{aligned}
&\sum_{i=0}^{(1-\mu)n-1} \binom{(1-\mu)n-1}{i} \left(\frac{a}{n}\right)^i \left(1-\frac{a}{n}\right)^{(1-\mu)n-1-i} \frac{1}{i+1} \\
&\quad = \sum_{i=0}^{(1-\mu)n-1} \frac{((1-\mu)n-1)!}{(i+1)!\,((1-\mu)n-1-i)!} \left(\frac{a}{n}\right)^i \left(1-\frac{a}{n}\right)^{(1-\mu)n-1-i} \\
&\quad = \sum_{i=1}^{(1-\mu)n} \frac{((1-\mu)n-1)!}{i!\,((1-\mu)n-i)!} \left(\frac{a}{n}\right)^{i-1} \left(1-\frac{a}{n}\right)^{(1-\mu)n-i} \\
&\quad = \frac{n}{a}\left(-\frac{(1-\frac{a}{n})^{(1-\mu)n}}{(1-\mu)n} + \frac{1}{(1-\mu)n} \sum_{i=0}^{(1-\mu)n} \frac{((1-\mu)n)!}{i!\,((1-\mu)n-i)!} \left(\frac{a}{n}\right)^i \left(1-\frac{a}{n}\right)^{(1-\mu)n-i}\right).
\end{aligned}$$

Let $X \sim Bin((1-\mu)n, a/n)$. We have

$$1 = \sum_{i=0}^{(1-\mu)n} P[X=i] = \sum_{i=0}^{(1-\mu)n} \frac{((1-\mu)n)!}{i!\,((1-\mu)n-i)!} \left(\frac{a}{n}\right)^i \left(1-\frac{a}{n}\right)^{(1-\mu)n-i}.$$

Hence we arrive at

$$\begin{aligned}
\sum_{i=0}^{(1-\mu)n-1} \binom{(1-\mu)n-1}{i} \left(\frac{a}{n}\right)^i \left(1-\frac{a}{n}\right)^{(1-\mu)n-1-i} \frac{1}{i+1} &= \frac{n}{a}\left(-\frac{(1-\frac{a}{n})^{(1-\mu)n}}{(1-\mu)n} + \frac{1}{(1-\mu)n}\right) \\
&= \frac{1-(1-\frac{a}{n})^{(1-\mu)n}}{a(1-\mu)}.
\end{aligned}$$

□

Proof of Lemma 2.5.13. We will omit y as an index in this proof, i.e. we will write PH instead of PH_y etc. First we verify that for all $j \in \{0, 1, \ldots, \mu n\}$

$$P[\mathsf{PH} \mid \mathsf{INF}(j)] \leq jP[\mathsf{PH} \mid \mathsf{INF}(1)]. \tag{2.5.1}$$

It is

$$P[\mathsf{PH} \mid \mathsf{INF}(j)] = P[\mathsf{PH}(x_1) \cup \ldots \cup \mathsf{PH}(x_j) \mid \mathsf{INF}(j)].$$

Hence, by applying the union bound,

$$P[\mathsf{PH} \mid \mathsf{INF}(j)] \leq jP[\mathsf{PH}(x_1) \mid \mathsf{INF}(j)] = jP[\mathsf{PH}(x_1) \mid \mathsf{INF}(1)] = jP[\mathsf{PH} \mid \mathsf{INF}(1)],$$

thus (2.5.1) holds. Next we prove that for all $j \in \{1, 2, \ldots, \mu n\}$

$$\frac{P[\mathsf{INF}(j) \mid \mathsf{PH}]}{P[\mathsf{INF}(1) \mid \mathsf{PH}]} \leq j\frac{P[\mathsf{INF}(j)]}{P[\mathsf{INF}(1)]}. \tag{2.5.2}$$

Using Bayes' Theorem and (2.5.1) we obtain

$$P[\mathsf{INF}(j) \mid \mathsf{PH}] = \frac{P[\mathsf{PH} \mid \mathsf{INF}(j)]P[\mathsf{INF}(j)]}{P[\mathsf{PH}]} \leq \frac{jP[\mathsf{PH} \mid \mathsf{INF}(1)]P[\mathsf{INF}(j)]}{P[\mathsf{PH}]}.$$

Hence, by again applying Bayes' Theorem, we obtain

$$P[\mathsf{INF}(j) \mid \mathsf{PH}] \leq j\frac{P[\mathsf{INF}(1) \mid \mathsf{PH}]P[\mathsf{INF}(j)]}{P[\mathsf{INF}(1)]}$$

which implies (2.5.2). We have

$$\begin{aligned}\frac{P[\mathsf{INF}(j)]}{P[\mathsf{INF}(1)]} &= \frac{\binom{\mu n}{j}(\frac{a}{n})^j(1-\frac{a}{n})^{\mu n-j}}{\binom{\mu n}{1}\frac{a}{n}(1-\frac{a}{n})^{\mu n-1}} = \frac{(\mu n)!}{j!\,(\mu n-j)!\,\mu n}\left(\frac{a}{n}\right)^{j-1}\left(1-\frac{a}{n}\right)^{-j+1} \\ &= \frac{\mu n-1}{n}\frac{\mu n-2}{n}\cdot\ldots\cdot\frac{\mu n-j+1}{n}\cdot\frac{a^{j-1}}{j!}\left(1-\frac{a}{n}\right)^{-j+1} \leq \mu^{j-1}\frac{a^{j-1}}{j!}\left(1-\frac{a}{n}\right)^{-j+1}.\end{aligned}$$

Hence, using (2.5.2), we can infer that for all $j \in \{1, 2, \ldots, \mu n\}$

$$\frac{P[\mathsf{INF}(j) \mid \mathsf{PH}]}{P[\mathsf{INF}(1) \mid \mathsf{PH}]} \leq \mu^{j-1}\frac{a^{j-1}}{(j-1)!}\left(1-\frac{a}{n}\right)^{-j+1} = \mathcal{O}(\mu^{j-1}) \text{ for } \mu \to 0. \tag{2.5.3}$$

In the following, $\mathcal{O}(\mu)$ refers to $\mu \to 0$. We prove

$$P[\mathsf{INF}(1) \mid \mathsf{PH}] = 1 + \mathcal{O}(\mu). \tag{2.5.4}$$

Using (2.5.3) we get

$$\frac{P[\mathsf{INF}(2) \mid \mathsf{PH}] + P[\mathsf{INF}(3) \mid \mathsf{PH}] + \cdots + P[\mathsf{INF}(\mu n) \mid \mathsf{PH}]}{P[\mathsf{INF}(1) \mid \mathsf{PH}]} = \mathcal{O}(\mu).$$

Therefore

$$P[\mathsf{INF}(2) \mid \mathsf{PH}] + P[\mathsf{INF}(3) \mid \mathsf{PH}] + \cdots + P[\mathsf{INF}(\mu n) \mid \mathsf{PH}] = P[\mathsf{INF}(1) \mid \mathsf{PH}] \cdot \mathcal{O}(\mu)$$

and thus

$$1 = P[\mathsf{INF}(1) \mid \mathsf{PH}] + P[\mathsf{INF}(2) \mid \mathsf{PH}] + \cdots + P[\mathsf{INF}(\mu n) \mid \mathsf{PH}] = P[\mathsf{INF}(1) \mid \mathsf{PH}] \cdot (1 + \mathcal{O}(\mu))$$

which implies (2.5.4). Using (2.5.4) and that $P[\mathsf{PL} \mid \mathsf{PH} \cap \mathsf{INF}(1)] = P[\mathsf{PL} \mid \mathsf{INF}(1)]$ we obtain

$$P[\mathsf{PL} \mid \mathsf{PH}] = P[\mathsf{PL} \mid \mathsf{INF}(1)] + \mathcal{O}(\mu). \tag{2.5.5}$$

Thus, to finish the proof, it suffices to show

$$P[\mathsf{PL} \mid \mathsf{INF}(1)] = \frac{1-e^{-a}}{a} + \mathcal{O}(\mu). \tag{2.5.6}$$

For each $j \in \{0, 1, \ldots, (1-\mu)n-1\}$ let $\mathsf{UNF}(j)$ denote the event that y has exactly j uninformed neighbours in the current round. Note that at the beginning of the round there is a

fixed number of informed nodes, namely μn, and a fixed number of uninformed nodes, namely $(1-\mu)n$. In particular, as all edges are sampled independently, for any $j \in \{0, 1, \dots, (1-\mu)n-1\}$, $\mathsf{UNF}(j)$ and $\mathsf{INF}(1)$ are independent. Hence, as $P[\mathsf{PL} \mid \mathsf{INF}(1) \cap \mathsf{UNF}(j)] = 1/(j+1)$,

$$\begin{aligned} P[\mathsf{PL} \mid \mathsf{INF}(1)] &= \sum_{j=0}^{(1-\mu)n-1} P[\mathsf{UNF}(j)] \frac{1}{j+1} \\ &= \sum_{j=0}^{(1-\mu)n-1} \binom{(1-\mu)n-1}{j} \left(\frac{a}{n}\right)^j \left(1-\frac{a}{n}\right)^{(1-\mu)n-1-j} \frac{1}{j+1}. \end{aligned}$$

Thus, using Lemma 2.5.14, we can infer

$$P[\mathsf{PL} \mid \mathsf{INF}(1)] = \frac{1-(1-\frac{a}{n})^{(1-\mu)n}}{a(1-\mu)}.$$

Using Fact 2.3.3, this gives

$$P[\mathsf{PL} \mid \mathsf{INF}(1)] = \frac{1-e^{a(\mu-1)}}{(1-\mu)a} + \mathcal{O}(\mu).$$

Thus, using the series representation of the exponential function, we obtain

$$P[\mathsf{PL} \mid \mathsf{INF}(1)] = \frac{1-e^{-a}}{a} + \mathcal{O}(\mu)$$

which shows (2.5.6) and hence completes the proof. □

Proof of Theorem 2.5.11. We want to use the framework from [34]. In order to do this, we consider a round of the information spreading process that starts with k informed and $u = n-k$ uninformed nodes; we will refer to this round as the *current round*. Let $\mu := k/n$. We can assume that at the start of the round, the edges of the Erdős-Rényi random graph $G(n, a/n)$ are not yet sampled. Before the random graph is sampled, each uninformed node has the same probability of getting informed, hence the information spreading process is homogeneous. We start with showing that the covariance conditions are fulfilled. To do this, consider two uninformed nodes x and y. As before, E denotes the event that x and y become neighbours in the current round. We have

$$Cov(\mathbb{1}_{\mathsf{PP}_x}, \mathbb{1}_{\mathsf{PP}_y}) = P[\mathsf{PP}_x \cap \mathsf{PP}_y] - P[\mathsf{PP}_x]P[\mathsf{PP}_y] = P[\mathsf{PP}_x](P[\mathsf{PP}_y \mid \mathsf{PP}_x] - P[\mathsf{PP}_y]).$$

It is

$$P[\mathsf{PP}_y \mid \mathsf{PP}_x] \leq P[\mathsf{PP}_y \mid \neg\mathsf{E}] = \frac{P[\mathsf{PP}_y \cap \neg\mathsf{E}]}{P[\neg\mathsf{E}]} \leq \frac{P[\mathsf{PP}_y]}{P[\neg\mathsf{E}]} = \frac{P[\mathsf{PP}_y]}{1-a/n} = P[\mathsf{PP}_y] + \mathcal{O}(1/n).$$

Hence

$$Cov(\mathbb{1}_{\mathsf{PP}_x}, \mathbb{1}_{\mathsf{PP}_y}) = P[\mathsf{PP}_x] \cdot \mathcal{O}(1/n). \tag{2.5.7}$$

From [34] it is known that

$$P[\mathsf{PH}_x] \leq \frac{k}{n}(1 - e^{-a} + \mathcal{O}(1/n))$$

and therefore

$$P[\mathsf{PP}_x] \leq P[\mathsf{PH}_x] + P[\mathsf{PL}_x] \leq (1 - e^{-a} + \mathcal{O}(1/n))\frac{k}{n} + (1 + \mathcal{O}(1/n))\frac{k}{n} = (2 - e^{-a} + \mathcal{O}(1/n))\frac{k}{n}.$$

This, together with (2.5.7), yields

$$Cov(\mathbb{1}_{\mathsf{PP}_x}, \mathbb{1}_{\mathsf{PP}_y}) = \frac{k}{n}\mathcal{O}(1/n).$$

Hence the covariance conditions are fulfilled for the exponential growth and shrinking conditions.

For the exponential growth phase, we have to estimate the success probability $p_k = P[\mathsf{PP}_y]$ that an uninformed node y becomes informed in the current round that starts with k informed nodes. In the following, we write PP, PH and PL instead of $\mathsf{PP}_y, \mathsf{PH}_y$ and PL_y respectively. Note that PH and PL are not independent (as the edges are not yet sampled at the beginning of the round). It is

$$P[\mathsf{PP}] = P[\mathsf{PH} \cup \mathsf{PL}] = P[\mathsf{PH}] + P[\mathsf{PL}] - P[\mathsf{PH} \cap \mathsf{PL}]. \tag{2.5.8}$$

To compute $P[\mathsf{PP}]$ we consider the three summands of (2.5.8) individually:

Term 1 $P[\mathsf{PH}]$: From [34] it is known that

$$\mu(1 - e^{-a})\left(1 - \frac{k + \mathcal{O}(1)}{2n}(1 - e^{-a})\right) \leq P[\mathsf{PH}] \leq \mu(1 - e^{-a} + \mathcal{O}(1/n)). \tag{2.5.9}$$

Term 2 $P[\mathsf{PL}]$: According to Fact 2.5.8, y is isolated with probability $e^{-a} + \mathcal{O}(1/n)$. Thus

$$P[\mathsf{PL}] = (1 - e^{-a} + \mathcal{O}(1/n))\mu. \tag{2.5.10}$$

Term 3 $P[\mathsf{PL} \cap \mathsf{PH}]$: We have

$$P[\mathsf{PL} \cap \mathsf{PH}] = P[\mathsf{PL} \mid \mathsf{PH}]P[\mathsf{PH}]. \tag{2.5.11}$$

Thus, using Lemma 2.5.13 and (2.5.9) we obtain

$$P[\mathsf{PL} \cap \mathsf{PH}] = \mu\frac{(1 - e^{-a})^2}{a} + \mathcal{O}(\mu^2) \text{ for } \mu \to 0.$$

Combining the three terms in (2.5.8), where asymptotic notation is with respect to $\mu \to 0$, we obtain

$$P[\mathsf{PP}] = \left(2(1 - e^{-a}) - \frac{(1 - e^{-a})^2}{a} + \mathcal{O}(\mu)\right)\mu = \left(2(1 - e^{-a}) - \frac{(1 - e^{-a})^2}{a}\right)(1 + \mathcal{O}(\mu))\mu.$$

In particular, there is an $a^* \geq 0$ such that

$$p_k = P[\mathsf{PP}] \underset{\leq}{\geq} \left(2(1-e^{-a}) - \frac{(1-e^{-a})^2}{a}\right) \frac{k}{n} \left(1 \underset{+}{-} a^* \frac{k}{n}\right).$$

Hence there is a constant $f > 0$ such that the exponential growth conditions are fulfilled for $\kappa = 2(1-e^{-a}) - (1-e^{-a})^2/a$. Thus Theorem 2.5.4 yields

$$E[X_n(1, fn)] = \log_{1+\kappa}(n) + \mathcal{O}(1)$$

and that there are constants $A_1, \alpha_1 > 0$ such that for all $r, n \in \mathbb{N}$

$$P[|X_n(1, fn) - \log_{1+\kappa}(n)| \geq r] \leq A_1 \exp(-\alpha_1 r).$$

Moreover, it yields that there is an $f < f' < 1$ such that with probability $1 + \mathcal{O}(1/n)$ at most $f'n$ nodes are informed at the end of round $X_n(1, fn)$.

Let $g \in (0,1)$ be an arbitrary constant. To complete the proof we show that $1/a \ln(n) + \mathcal{O}(1)$ is a lower bound for the number of rounds needed to inform all remaining nodes, starting with $n - \lfloor gn \rfloor$ informed nodes; then the claim follows as *Pull* provides a matching upper bound for the exponential shrinking phase. Consider an uninformed node y. According to Fact 2.5.8, y is isolated in the current round with probability $e^{-a} + \mathcal{O}(1/n) = e^{-a+\mathcal{O}(1/n)}$. If y is isolated, then it cannot be informed in the current round. Therefore

$$1 - P[\mathsf{PP}] \geq e^{-a} + \mathcal{O}(1/n).$$

Thus the lower exponential shrinking conditions are fulfilled for $\rho_n = a + \mathcal{O}(1/n)$ and arbitrary $g \in (0,1)$. Therefore Theorem 2.5.6 yields

$$E[X_n(n - \lfloor gn \rfloor, n)] \geq \frac{1}{a} \ln(n) + \mathcal{O}(1)$$

and that there are $A_2, \alpha_2 > 0$ such that for all $r, n \in \mathbb{N}$

$$P\left[X_n(n - \lfloor gn \rfloor, n) \leq \frac{1}{a} \ln(n) - r\right] \leq A_2 \exp(-\alpha_2 r).$$

Together with the upper bounds that we obtain by considering the exponential shrinking phase of *Pull*, this completes the proof. □

Remark 2.5.15. *It is interesting that Push&Pull does (unlike Push and Pull) behave differently on random evolving graphs than on the complete graph with message transmission success probability $1 - e^{-a}$ (which is, up to an additive $\mathcal{O}(1/n)$ term, the probability that a node is not isolated): The expected runtime in the letter case is only $\log_{3-2e^{-a}}(n) + 1/(1 - e^{-a} + a) \ln(n) + \mathcal{O}(1)$ ([34]).*

One reason for this differing behaviour is that, when using Push&Pull where the underlying graph is in each round a newly sampled Erdős-Rényi random graph $G(n, a/n)$, push and pull operations get in each other's way, i.e. it has a substantial impact that some nodes get informed in the same round by a push as well as by a pull operation which makes one of those operations useless. This is in contrast to the situation on the complete graph with

message transmission success probability $1 - e^{-a}$: There, in the beginning of the information spreading process, Push and Pull essentially do not get in each other's way, i.e. only very few nodes get informed by a push as well as by a pull operation. This is because in the complete graph, each node has $n - 1$ neighbours while in the random evolving setting considered here, the expected number of neighbours of a node is in each round $a + \mathcal{O}(1/n)$ and therefore here it is much more likely that a relevant fraction of the edges is used by Push as well as by Pull.

The other reason is the behaviour of Push&Pull in the last phase of the process: As in the investigated setting both, Push and Pull, need logarithmic time for the exponential shrinking phase, one might conjecture that in the Push&Pull setting Push as well as Pull contribute substantially to the last phase. The reason why this is not the case, i.e. why only Pull contributes substantially to the last phase, is the following: Consider an uninformed node x in the last phase, i.e. if most nodes are informed already. In each round we can first sample whether x is isolated which is the case with probability $e^{-a} + \mathcal{O}(1/n)$. If x is isolated, it cannot be informed in that round, neither by a push nor by a pull operation. However, if x is not isolated it is very probable that it becomes informed by a pull attempt. In particular, the case that it does become informed by a push operation but not simultaneously also by a pull operation is very unlikely. So the problem essentially is that both, pull and push attempts, have to clear the same hurdle, i.e. wait for a round where x is not isolated. But after taking this hurdle, it is extremely unlikely that a pull operation does not succeed while a push operation does succeed. In contrast to this, on the complete graph with message transmission success probability $1 - e^{-a}$, both, Push and Pull, contribute essentially to the last phase; this is due to the fact that the message transmissions fail independently of each other.

2.6 Outlook

Many subjects that are related to the problems that we investigated in this chapter constitute interesting topics for future research. For example, it is desirable to obtain resilience results for more graph classes than the investigated expander graphs. Also in the setting with random evolving graphs there are various directions for future research: First, one could extend the results to a broader range of p (though, e.g. according to [34], $p = a/n$ is the most interesting regime). Second, it is an interesting task to obtain results for the runtimes of *Push*, *Pull* and *Push&Pull* also for the edge-Markovian setting introduced in [21] (cf. the paragraph "Evolving Graphs" in Subsection 2.2.2): For *Push*, in [21], rather loose bounds are shown which should be sharpened and for *Pull* and *Push&Pull* to our best knowledge no results at all are yet proven in the edge-Markovian setting.

Chapter 3

On a Graph Theoretical Model for Opinion Spreading

3.1 Introduction

How opinions are formed is of great interest. It affects which products are popular, how elections turn out and countless more aspects of our lives. The internet can be considered as a catalyst for the spreading of opinions. One step towards understanding how opinions spread within social groups was made by Rogers [94]: According to him, new ideas spread starting from so-called early adopters which have huge influence on others. Motivated by this assumption, in [5], Alon et al. introduced a model that describes how opinions spread in a social network (which is modelled by a graph). For different kinds of adversaries that try to convince the majority of a falsehood (blue), they investigate under which conditions the truth (red) wins despite the adversary's efforts. Let $\mathcal{G} = (G_n)_{n\in\mathbb{N}} = ((V_n, E_n))_{n\in\mathbb{N}}$ denote a sequence of graphs with $|V_n| = n$. $\mathcal{G}$ is called robust against a certain adversary if with probability $1 + o(1)$ (as $n \to \infty$) at the end of the dissemination process the majority of the nodes in V_n will believe the truth (i.e. will become red) despite the adversary's efforts.

If a graph or a sequence of graphs possesses a property, a natural question is: "How strongly does it posses the property?" This question is formalised by the concept of local resilience, compare for example the article by Sudakov and Vu ([99]). Local resilience of a graph with respect to a certain property measures, up to how many edges one has to be allowed to delete at each node to be able to destroy the property. If it is a non-monotonous property, also adding edges can be taken under consideration. The concept of local resilience easily generalises to sequences of graphs. Thus, if a graph is robust against a certain kind of adversary, this motivates to ask how strongly is it robust against that adversary or, more specifically, what is the local resilience of the graph with respect to being robust against that adversary? This can be of practical relevance, too. For example, an adversary may be able to change a graph to some extent by deleting edges, e.g. by filtering information someone obtains.

Another interesting question comes up if we consider the problem from the adversary's perspective. Is it NP-hard to determine an optimal solution? We answer this affirmatively even for a seemingly simpler special case.

3.2 The Model and Related Results

In the following, we describe the model that we will use. It was introduced in [5]. For a graph $G = (V, E)$, at the beginning of the dissemination process, each node $v \in V$ can either have an opinion or have no opinion; at the end, every node has an opinion. There are two opinions, namely *red* (which is considered to be true) and *blue* (which is considered to be false). A node that has an opinion keeps this opinion forever. Let $0 < \mu < 1/2$. In the beginning only $\mu|V|$ so called experts $\mathcal{E} \subseteq V$ have an opinion. Now the opinions are disseminated as follows: Every non-expert takes the opinion that the majority of its expert neighbours has, ties are broken independently and uniformly at random, including the case when a node has no expert neighbours. In particular, after one round, every node has an opinion.

We will investigate the following two kinds of adversaries introduced in [5] which have influence on the expert set $\mathcal{E}$. Let $0 < \delta < 1/2$. The weak adversary is allowed to choose $\mathcal{E}$. Then every node from $\mathcal{E}$ turns red with probability $1/2 + \delta$ and blue otherwise. The strong adversary can choose $\mathcal{E}$ as well. However, additionally he may assign the colours to the experts in $\mathcal{E}$ as long as he respects the ratio of red to blue experts to be $(1/2+\delta)/(1/2-\delta)$. A sequence of graphs $\mathcal{G} = (G_n)_{n\in\mathbb{N}}$ (where G_n has n nodes) is called *robust against a certain kind of adversary* if, with probability $1 + o(1)$ (as $n \to \infty$), the majority of the nodes of G_n (i.e. more than $n/2$ nodes) are red at the end of the spreading process despite the adversary's efforts.

In [5], several results for this model are proven. For the precise statements we refer to their article, here we just summarise which kinds of results they provide:

- If the maximum degree of a sufficiently large graph is suitably bounded from above, then the probability that the weak adversary wins is upper bounded close to zero.
- In graphs with good expansion properties, the strong adversary loses.
- If $\mathcal{G}$ is a sequence of Erdős-Rényi random graphs, i.e. $G_n = G(n, p(n))$, and if the expected degree is suitably lower bounded, then $\mathcal{G}$ is robust against the strong adversary.
- An alternative, iterative dissemination process is introduced and briefly investigated.

Notation 3.2.1. *Consider a sequence of graphs $\mathcal{G} = (G_n)_{n\in\mathbb{N}} = ((V_n, E_n))_{n\in\mathbb{N}}$ where $|V_n| = n$. As in Chapter 2, we are interested in large values of n, hence, as before, if not stated differently, asymptotic notation is with respect to $n \to \infty$ and in particular, "whp" (which is short for "with high probability") means with probability $1 + o(1)$ (as $n \to \infty$). When we consider a probability $p(n)$ that depends on n, we sometimes write p instead of $p(n)$ if n is clear from the context. We consider opinion spreading according to the described model: We will write $\mathcal{E} = \mathcal{E}^{(n)} \subseteq V_n$ for the set of experts and $\mathcal{N} = \mathcal{N}^{(n)} = V_n \backslash \mathcal{E}^{(n)}$ for the set of non-experts; we use $\mathcal{R} = \mathcal{R}^{(n)} \subseteq V_n$ to denote the (random) set of nodes that are red when the dissemination process has finished and we use $\mathcal{B} = \mathcal{B}^{(n)} = V_n \backslash \mathcal{R}^{(n)}$ to denote the (random) set of nodes that are blue at the end of the process. For $v \in V_n$ let $R(v)$ and $B(v)$ denote the events that $v \in \mathcal{R}$ and $v \in \mathcal{B}$ respectively. For simplicity of exposition we ignore rounding issues if they do not affect the result. For a subset $A \subseteq \mathbb{R}$ that has no upper bound we set $\sup A = +\infty$. With μ and δ we always refer to quantities $\mu, \delta \in (0, 1/2)$ as in the model definition. As in Chapter 2, we use the following notation for the basic*

graph theoretical terms. For a graph $G = (V, E)$ and $v \in V$ let $N(v) = N_G(v)$ denote the neighbourhood of v; let $d(v) = d_G(v) = |N_G(v)|$ denote the degree of v. For $U, W \subseteq V$ with $U \cap W = \emptyset$ let $E(U, W) = E_G(U, W) \subseteq E$ denote the set of edges with one node in U and one node in W and set $e(U, W) = e_G(U, W) = |E_G(U, W)|$. Consider two graphs $H = (V, E)$ and $\tilde{H} = (V, \tilde{E})$ defined on the same node set V and a node $v \in V$; define $d_{H,\tilde{H}}(v) := |N_H(v) \triangle N_{\tilde{H}}(v)| = |(N_H(v) \backslash N_{\tilde{H}}(v)) \cup (N_{\tilde{H}}(v) \backslash N_H(v))|$.

3.3 Local Resilience

In [99], Sudakov and Vu initiated a systematic study of graph resilience. Since then, the concept received a lot of attention. Loosely speaking, the local resilience of a graph with respect to a property of graphs is the number (or fraction) of edges that an adversary can be allowed to delete at each node without being able to destroy the property. In the following, we provide the definition of local resilience that we will use; note that it slightly differs from the definition provided in [99] where at each node the absolute number of edges that may be deleted is considered. In contrast, we, like e.g. also Dellamonica et al. ([29]), consider the fraction of edges that can be deleted. However, as we consider Erdős-Rényi random graphs, due to the Chernoff bounds, for the considered range of p, the maximum degree is close to the minimum degree. Hence in the present setting, it does not make a crucial difference which of both definitions is used.

Definition 3.3.1 (Local resilience of a graph)**.** *Let $G = (V, E)$ be a graph without isolated nodes and P a property of graphs. Then the local resilience $r = r_P(G)$ of G with respect to P is defined as*

$$r := \sup_{q \in [0,1]} \left\{ q \mid \nexists \tilde{G} = (V, \tilde{E}), \tilde{E} \subseteq E : \forall v \in V : 1 - \tfrac{d_{\tilde{G}}(v)}{d_G(v)} < q \text{ and } P(\tilde{G}) \text{ does not hold} \right\}.$$

Definition 3.3.2 generalises the concept of local resilience to sequences of Erdős-Rényi random graphs; cf. [99].

Definition 3.3.2 (Local resilience of Erdős-Rényi random graphs)**.** *Let $p : \mathbb{N} \to (0, 1]$ with $p = \omega(\ln(n)/n)$, let $\mathcal{G} = (G_n)_{n \in \mathbb{N}}$ be a sequence of graphs where, for $n \in \mathbb{N}$, $G_n = G(n, p(n))$ is an Erdős-Rényi random graph and let P be a property of graphs. The local resilience $r^* = r^*_P(\mathcal{G})$ of $\mathcal{G}$ with respect to P is defined as*

$$r^* := \sup_{q \in [0,1]} \{q \mid \text{whp } r_P(G_n) \geq q\}.$$

In [99] it is noted that for non-monotonous properties also edge insertions should be considered, cf. Definitions 1.3 and 2.1 in [99]. A notion of local resilience with deletions and insertions is given in Definition 3.3.3; Definition 3.3.4 generalises the concept of local resilience with deletions and insertions to sequences of Erdős-Rényi random graphs. Definitions 3.3.3 and 3.3.4 are very similar to Definitions 1.3 and 2.1 in [99] respectively, with the slight difference that here we consider relative fractions of the edges instead of absolute numbers. However, as mentioned above, in the settings that we will consider, the maximum and the minimum degree are close together, hence it does not make a crucial difference.

Definition 3.3.3 (Local resilience with deletions and insertions)**.** *Consider a graph $G = (V, E)$ without isolated nodes and let P be a property of graphs. Then the local resilience $\tilde{r} = \tilde{r}_P(G)$ with deletions and insertions of G with respect to P is defined as*

$$\tilde{r} := \sup_{q \geq 0} \left\{ q \mid \nexists \tilde{G} = (V, \tilde{E}) : \forall v \in V : \frac{d_{G,\tilde{G}}(v)}{d_G(v)} < q \text{ and } P(\tilde{G}) \text{ does not hold} \right\}.$$

Definition 3.3.4 (Local resilience with deletions and insertions of Erdős-Rényi random graphs)**.** *Let $p : \mathbb{N} \to (0, 1]$ with $p = \omega(\ln(n)/n)$, let $\mathcal{G} = (G_n)_{n \in \mathbb{N}}$ be a sequence of graphs where, for $n \in \mathbb{N}$, $G_n = G(n, p(n))$ is an Erdős-Rényi random graph and let P be a property of graphs. The local resilience with deletions and insertions $\tilde{r}^* = \tilde{r}^*_P(\mathcal{G})$ of $\mathcal{G}$ with respect to P is defined as*

$$\tilde{r}^* := \sup_{q \geq 0} \{ q \mid \text{whp } \tilde{r}_P(G_n) \geq q \}.$$

Remark 3.3.5. *Note that, although there we did not explicitly use the notion of local resilience, also Theorems 2.4.8 and 2.4.9 are results regarding robustness in the sense of local resilience.*

3.4 Results

Theorem 3.4.1 states that a sufficiently large minimum degree guarantees robustness against the weak adversary. Theorem 3.4.2 assures that Theorem 3.4.1 is tight for $\mu \leq 1/(3 - 2\delta)$. Note that this covers the relevant range of μ as up to more than one third of the nodes can be experts which constitutes a very large fraction.

Theorem 3.4.1. *Let $f : \mathbb{N} \to \mathbb{R}_0^+$ with $f = \omega(\sqrt{n})$ and let $\mathcal{G} = (G_n)_{n \in \mathbb{N}} = ((V_n, E_n))_{n \in \mathbb{N}}$ be a sequence of graphs with $|V_n| = n$. For each $n \in \mathbb{N}$ let $m = m(n)$ denote the minimum degree of G_n and assume that $m(n) \geq n(1 + \mu - 2\delta\mu)/2 + f(n)$. Then $\mathcal{G}$ is robust against the weak adversary.*

Theorem 3.4.2. *Let $0 < \mu \leq 1/(3 - 2\delta)$. There is a constant $p^* > 0$ such that for any $n \in \mathbb{N}$ a graph G with n nodes and minimum degree $m \geq n(1 + \mu - 2\delta\mu)/2 + \mathcal{O}(1)$ exists such that there is a strategy for the weak adversary such that with probability at least p^* more than $n/2$ nodes are blue at the end of the spreading process.*

Theorem 3.4.3 provides a lower bound for the local resilience of Erdős-Rényi random graphs with respect to robustness against the strong adversary. Theorem 3.4.4 shows that this bound is tight.

Theorem 3.4.3. *Let $p : \mathbb{N} \to (0, 1]$ with $p = \omega(\ln(n)/n)$. Let $\mathcal{G} = (G_n)_{n \in \mathbb{N}}$ be a sequence of graphs where G_n is an Erdős-Rényi random graph, $G_n = G(n, p(n))$. The local resilience of $\mathcal{G}$ with respect to robustness against the strong adversary is at least $2(1 - \mu + 2\delta\mu)\delta/(1 + 2\delta)$.*

Note that it always holds $2(1 - \mu + 2\delta\mu)\delta/(1 + 2\delta) > 2\mu\delta$.

Theorem 3.4.4. *Let $p : \mathbb{N} \to (0,1]$ with $p = \omega(\ln(n)/n)$. Let $\mathcal{G} = (G_n)_{n\in\mathbb{N}}$ be a sequence of graphs where G_n is an Erdős-Rényi random graph, $G_n = G(n, p(n))$. Suppose that for each $n \in \mathbb{N}$ the strong adversary is allowed to delete up to a $(1 + o(1))2(1 - \mu + 2\delta\mu)\delta/(1 + 2\delta)$ fraction of the edges at each node of G_n. Then there is a strategy for the strong adversary such that he wins whp. If G_n is the complete graph on n nodes, then there is a strategy such that he wins deterministically.*

Theorem 3.4.5 considers the setting where also edge insertions are allowed.

Theorem 3.4.5. *Let $p : \mathbb{N} \to (0,1]$ with $p = \omega(\ln(n)/n)$. Let $\mathcal{G} = (G_n)_{n\in\mathbb{N}}$ be a sequence of graphs where G_n is an Erdős-Rényi random graph, $G_n = G(n, p(n))$. Then the local resilience with deletions and insertions of $\mathcal{G}$ with respect to robustness against the strong adversary is*

$$(1 - \mu + 2\delta\mu)\left(\delta + \frac{1-2\delta}{2+4\delta}\liminf_{n\to\infty}\left(\max\left\{0, 2\delta - \frac{1-p(n)}{p(n)}\right\}\right)\right).$$

Theorem 3.4.6 considers the problem from the adversary's perspective. Even if we only consider graphs that are unions of disjoint cliques, the problem to find an optimal solution is NP-hard.

Theorem 3.4.6. *Let $G = (V, E)$ be a graph. Let us denote the problem to choose the red and blue experts $\mathcal{E} \cap \mathcal{R}, \mathcal{E} \cap \mathcal{B} \subseteq V$ such that the expected number of red nodes at the end of the spreading process, $E[|\mathcal{R}|]$, is minimised as the "strong adversary problem" (SAP). Then SAP is NP-hard even if we only consider graphs that are unions of disjoint cliques.*

3.5 Proofs

We will need the following Chernoff bounds which follow immediately from, e.g., [67, Thm. 2.1].

Theorem 3.5.1 (Chernoff Bounds)**.** *Let $X_1, \ldots, X_n$ be independent random variables. For $i \in \{1, \ldots, n\}$ assume that $0 \le X_i \le 1$. Let $X = \sum_{1\le i\le n} X_i$ and let $\mu = E[X] = \sum_{1\le i\le n} E[X_i]$. Then, for any $\varepsilon \ge 0$,*

$$P[X \ge (1+\varepsilon)\mu] \le \exp\left(-\frac{\varepsilon^2}{2+\varepsilon}\mu\right) \quad \text{and}$$
$$P[X \le (1-\varepsilon)\mu] \le \exp\left(-\frac{\varepsilon^2}{2}\mu\right).$$

We will also use Theorem 3.5.2 that, under a mild assumption, states that the probability that a binomial random variable is larger than its expectation is larger than 1/4.

Theorem 3.5.2 ([61])**.** *Let $n \in \mathbb{N}$ and let $p \in (0,1)$ with $p > 1/n$. Let $X \sim Bin(n, p)$. Then $P[X \ge np] > 1/4$.*

Proof of Theorem 3.4.1. Let $\varepsilon > 0$ and define

$$c := (1/2 - \delta)\frac{\left(\frac{2\delta}{1-2\delta}\right)^2}{2 + \frac{2\delta}{1-2\delta}} \quad \text{and} \quad a = a(n, c, \varepsilon) := \frac{1}{c}\ln(n^{1+\varepsilon}) = \frac{1+\varepsilon}{c}\ln(n).$$

Set

$$\mathcal{N}_{low} = \mathcal{N}_{low}^{(n)} := \{v \in \mathcal{N}^{(n)} \mid |N(v) \cap \mathcal{E}| < a\} \quad \text{and} \quad \mathcal{N}_{high} = \mathcal{N}_{high}^{(n)} := \mathcal{N}^{(n)} \backslash \mathcal{N}_{low}^{(n)}$$

and let $n_l := |\mathcal{N}_{low}|$ and $n_h := |\mathcal{N}_{high}|$. We will prove the claim by showing the following two sub-claims.

(I) Whp all nodes from $\mathcal{N}_{high}$ become red.

(II) If all nodes from $\mathcal{N}_{high}$ become red, then, even if all nodes from $\mathcal{N}_{low}$ become blue, still whp[1] the majority of all nodes turns red.

We start with (I). We have

$$P\left[\bigcap_{v \in \mathcal{N}_{high}} R(v)\right] = 1 - P[\exists v \in \mathcal{N}_{high} : B(v)].$$

Write $\mathcal{N}_{high} = \{x_1, x_2, \ldots, x_{n_h}\}$. Let $i \in \{1, 2, \ldots, n_h\}$ and set $a_i := |N(x_i) \cap \mathcal{E}|$, in particular we have $a_i \geq a$. Let $X_i := |N(x_i) \cap \mathcal{E} \cap \mathcal{B}|$. It is $X_i \sim Bin(a_i, 1/2 - \delta)$ and hence $E[X_i] = (1/2 - \delta)a_i$. Using the union bound we obtain

$$P\left[\bigcap_{v \in \mathcal{N}_{high}} R(v)\right] \geq 1 - \sum_{i=1}^{n_h} P[X_i \geq a_i/2] = 1 - \sum_{i=1}^{n_h} P\left[X_i \geq \left(1 + \frac{2\delta}{1-2\delta}\right)(1/2 - \delta)a_i\right].$$

Hence Theorem 3.5.1 yields

$$P\left[\bigcap_{v \in \mathcal{N}_{high}} R(v)\right] \geq 1 - \sum_{i=1}^{n_h} \exp\left(-\frac{(\frac{2\delta}{1-2\delta})^2}{2 + \frac{2\delta}{1-2\delta}}(1/2 - \delta)a_i\right) \geq 1 - n\exp(-ca) = 1 - n^{-\varepsilon} \overset{n\to\infty}{\to} 1.$$

We continue with (II), i.e. we show that it suffices if all nodes from $\mathcal{N}_{high}$ turn red. In order to do this we will show that

$$n_l \leq \frac{1 - \mu + 2\delta\mu}{2}n - f(n) + \mathcal{O}(\sqrt{n}). \tag{3.5.1}$$

From Theorem 3.5.1, we get that for any $f_2 : \mathbb{N} \to \mathbb{R}^+$ with $f_2 = \omega(\sqrt{n})$ whp

$$|\mathcal{E} \cap \mathcal{B}| \leq (1/2 - \delta)\mu n + f_2(n). \tag{3.5.2}$$

[1]The remaining randomness is due to the fact that the experts take their opinions randomly.

Therefore, if we prove (3.5.1), the claim follows by adding the upper bounds for n_l and $|\mathcal{E}\cap\mathcal{B}|$ from (3.5.1) and (3.5.2) respectively. To prove (3.5.1) let

$$k := |\{(u,v) \in \mathcal{E} \times \mathcal{N} \mid v \notin N(u)\}|$$

denote the number of edges that are missing between $\mathcal{E}$ and $\mathcal{N}$. By definition, for every node $v \in \mathcal{N}_{low}$ there are at least $\mu n - a$ nodes $w \in \mathcal{E}$ such that $w \notin N(v)$. Thus we can infer

$$n_l(\mu n - a) \leq k \quad \text{ or equivalently } \quad n_l \leq \frac{k}{\mu n - a}.$$

Hence, to prove (3.5.1), it suffices to show

$$\frac{k}{\mu n - a} \leq \frac{1 - \mu + 2\delta\mu}{2} n - f(n) + \mathcal{O}(\sqrt{n}). \tag{3.5.3}$$

In order to do this we first bound k. It is

$$k = \mu n(1-\mu)n - e(\mathcal{E}, \mathcal{N}). \tag{3.5.4}$$

We have

$$e(\mathcal{E}, \mathcal{N}) \geq (m - \mu n)\mu n.$$

This, together with (3.5.4), yields

$$k \leq \mu n(1-\mu)n - (m - \mu n)\mu n = \mu n(n-m).$$

Using this, we can show (3.5.3); we have

$$\frac{k}{\mu n - a} \leq \frac{\mu n(n-m)}{\mu n - a} = \frac{n-m}{1 - a/(\mu n)} = n - m + \mathcal{O}(\ln(n)).$$

Thus, as $m \geq n(1 + \mu - 2\delta\mu)/2 + f(n)$, we arrive at

$$\frac{k}{\mu n - a} \leq n - \frac{1 + \mu - 2\delta\mu}{2} n - f(n) + \mathcal{O}(\ln(n)) = \frac{1 - \mu + 2\delta\mu}{2} n - f(n) + \mathcal{O}(\ln(n))$$

which shows (3.5.3) and thus the claim. □

Proof of Theorem 3.4.2. Fix $n \in \mathbb{N}$. We define a graph $G = (V, E)$ with $|V| = n$ as follows. Let V be partitioned into two disjoint subsets of sizes μn and $(1-\mu)n$ respectively. In slight abuse of notation we denote these subsets by $\mathcal{E}$ and $\mathcal{N}$; i.e. $V = \mathcal{E} \cup \mathcal{N}$ with $|\mathcal{E}| = \mu n$, $|\mathcal{N}| = (1-\mu)n$. Both, $\mathcal{E}$ and $\mathcal{N}$, form cliques. Fix $v^* \in \mathcal{E}$, set $\tilde{\mathcal{E}} := \mathcal{E} \backslash \{v^*\}$. Let $c := 1/2 - \delta\mu/(1-\mu)$. Fix a subset $\mathcal{N}_1 \subseteq \mathcal{N}$ with $|\mathcal{N}_1| = c(n - \mu n)$ and denote $N_2 = \mathcal{N} \backslash \mathcal{N}_1$. For all $u \in \tilde{\mathcal{E}}$, $v \in \mathcal{N}_1$ and $w \in \mathcal{N}_2$ we set

$$u \in N(v), \qquad v^* \in N(v), \qquad u \notin N(w) \quad \text{ and } \quad v^* \in N(w).$$

We claim that the minimum degree of G is $n(1 + \mu - 2\delta\mu)/2 - 1$; we also claim that the probability that the weak adversary wins if he chooses $\mathcal{E}$ to be the experts can be bounded

from below by a positive constant. In order to verify these claims, we first compute the degrees; let u, v, w be as above. We obtain

$$d(v^*) = n-1, \qquad d(u) = \mu n - 1 + c(1-\mu)n, \qquad d(v) = n-1 \quad \text{and} \quad d(w) = (1-\mu)n.$$

It is

$$\begin{aligned} d(u) &= \mu n - 1 + c(1-\mu)n \\ &\leq \mu n + n\left(\frac{1-\mu}{2} - \delta\mu\right) = n\left(\mu + \frac{1-\mu}{2} - \delta\mu\right) = n\left(\frac{1}{2} + \frac{1}{2}\mu - \delta\mu\right). \end{aligned}$$

Thus, as $\mu \leq 1/(3-2\delta)$, a direct calculation yields

$$d(u) \leq (1-\mu)n.$$

Therefore the minimum degree m of G is

$$m = \mu n + c(1-\mu)n - 1 = \frac{1+\mu-2\delta\mu}{2}n - 1.$$

Hence we are left to show that with positive n-independent probability more than $n/2$ nodes are blue at the end of the dissemination process. Let C denote the event that $|\tilde{\mathcal{E}} \cap \mathcal{B}| \geq (1/2-\delta)(\mu n - 1)$. Using Theorem 3.5.2, as $|\tilde{\mathcal{E}} \cap \mathcal{B}| \sim Bin(\mu n - 1, 1/2 - \delta)$, we obtain

$$P[B(v^*) \cap C] = P[B(v^*)]P[C] \geq (1/2 - \delta)/4.$$

Thus it suffices to show that conditioned on $B(v^*) \cap C$ more than $n/2$ nodes are blue at the end of the dissemination process. Note that $B(v^*)$ implies that for all $w \in \mathcal{N}_2$ we have $B(w)$. Therefore, conditioned on $B(v^*) \cap C$, we have

$$|\mathcal{B}| \geq |\mathcal{N}_2| + 1 + (1/2-\delta)(\mu n - 1) > (1-c)(n-\mu n) + (1/2-\delta)\mu n = n/2.$$

□

Proof of Theorem 3.4.3. We write $G_n = (V_n, E_n)$. By assumption, we know that $p = \omega(\ln(n)/n)$. Thus, using Theorem 3.5.1 and the union bound, as for all $v \in V_n$ it is $d(v) \sim Bin(n-1, p)$, we obtain that whp for all $v \in V_n$

$$d(v) = (1 + o(1))np. \tag{3.5.5}$$

Next we show that whp for all but $o(n)$ nodes $v \in \mathcal{N} = \mathcal{N}^{(n)}$

$$|N(v) \cap \mathcal{E} \cap \mathcal{B}| = (1 + o(1))(1/2 - \delta)\mu np. \tag{3.5.6}$$

First consider the case that the blue experts $\mathcal{B} = \mathcal{B}^{(n)}$ and the red experts $\mathcal{R} = \mathcal{R}^{(n)}$ are chosen uniformly at random; we will relate this to the adversarial case later. As we assume $\mathcal{B}$ and $\mathcal{R}$ to be chosen uniformly at random, we can assume that $\mathcal{B}$ and $\mathcal{R}$ are chosen before the edges are sampled; hence, for $v \in \mathcal{N}$, it is $|N(v) \cap \mathcal{E} \cap \mathcal{B}| \sim Bin((1/2-\delta)\mu n, p)$. As $p = \omega(\ln(n)/n)$, Theorem 3.5.1 yields that there is a function $g : \mathbb{N} \to \mathbb{R}^+$ with $g = \omega(1)$

such that for any $v \in \mathcal{N}$ with probability at least $1 - e^{-g(n)}$ Equation (3.5.6) holds.[2] Let Y denote the number of nodes $v \in \mathcal{N}$ for which (3.5.6) does not hold. Note that Y can be bounded from above by a binomial random variable $Bin((1-\mu)n, e^{-g(n)})$. Set

$$h := \frac{1}{\sqrt{g}} = o(1).$$

Recall that for $k, l \in \mathbb{N}, k \leq l$

$$\binom{l}{k} \leq \left(\frac{el}{k}\right)^k.$$

Therefore

$$\begin{aligned} P[Y \geq h(n)n] &\leq \binom{n}{h(n)n} \exp\Big(-g(n)h(n)n\Big) \leq \left(\frac{e}{h(n)}\right)^{h(n)n} \exp\Big(-\sqrt{g(n)}n\Big) \\ &= \exp\left(\left(h(n) - \ln(h(n))h(n) - \sqrt{g(n)}\right)n\right). \end{aligned}$$

As $\ln(h(n))h(n) = o(1)$ we can infer

$$P[Y \geq h(n)n] = e^{-\omega(n)}.$$

Thus we have shown that if $\mathcal{B}$ and $\mathcal{R}$ are chosen uniformly at random, then with probability $1 - e^{-\omega(n)}$ for all but $h(n)n = o(n)$ nodes $v \in \mathcal{N}$ Equation (3.5.6) holds. As every node is either a blue expert, a red expert or no expert, there are at most 3^n possibilities how $\mathcal{R}$ and $\mathcal{B}$ can be set. Therefore we obtain that even if $\mathcal{R}$ and $\mathcal{B}$ are chosen adversarially, with probability $1 - 3^n e^{-\omega(n)} = 1 + o(1)$ for all but $o(n)$ nodes $v \in \mathcal{N}$ Equation (3.5.6) holds. A similar approach is applied in [5] to prove that $G = G(n,p)$ is robust against the strong adversary for a suitable range of p. Analogously we can infer that whp for all but $o(n)$ nodes $v \in \mathcal{N}$

$$|N(v) \cap \mathcal{E} \cap \mathcal{R}| = (1 + o(1))(1/2 + \delta)\mu np. \tag{3.5.7}$$

Note that the strong adversary loses, i.e. the majority of all nodes turns red, if less than $n(1-\mu+2\delta\mu)/2$ non-experts become blue. Equations (3.5.6) and (3.5.7) imply that whp for all but $o(n)$ nodes $v \in \mathcal{N}$ the following holds: A necessary condition for v becoming blue is that the adversary deletes $(1 + o(1))2\delta\mu np$ edges between v and the red experts. Therefore, in total, whp, the adversary must delete

$$(1 + o(1))\frac{1 - \mu + 2\delta\mu}{2} n \cdot 2\delta\mu np = (1 + o(1))(1 - \mu + 2\delta\mu)\delta\mu n^2 p \tag{3.5.8}$$

edges between the red experts and the non-experts. In particular, if that many edges are deleted, as there are $(1/2 + \delta)\mu n$ red experts, there is a red expert v^* that loses at least

$$(1 + o(1))\frac{(1 - \mu + 2\delta\mu)\delta\mu n^2 p}{(1/2 + \delta)\mu n} = (1 + o(1))\frac{(1 - \mu + 2\delta\mu)\delta np}{1/2 + \delta}$$

[2] This is equivalent to the existence of a function $h : \mathbb{N} \to \mathbb{R}^+$ with $h = o(1)$ such that with probability $1 - h(n)$ Equation (3.5.6) holds; however, the used form will slightly simplify the calculation.

of its edges. This, together with (3.5.5), implies that then v^* whp loses a

$$(1+o(1))\frac{(1-\mu+2\delta\mu)\delta np}{(1/2+\delta)np} = (1+o(1))\frac{2(1-\mu+2\delta\mu)\delta}{1+2\delta}$$

fraction of its edges which shows the claim. □

The following proof could also have been merged with the proof of Theorem 3.4.3 without too many alterations, thereby saving some calculations; however, in order to obtain a clearer structure, we present it independently.

Proof of Theorem 3.4.4. We show that if the adversary uses the following randomised strategy, then he does not delete too many edges and, whp, wins.

1. Choose three disjoint sets of sizes

$$(1/2+\delta)\mu n, \qquad (1/2-\delta)\mu n \quad \text{and} \quad \frac{1-\mu+2\delta\mu}{2}n+1$$

uniformly at random, in slight abuse of notation we call these sets $\mathcal{R}\cap\mathcal{E}, \mathcal{B}\cap\mathcal{E}$ and $\mathcal{N}_B$ respectively. $\mathcal{R}\cap\mathcal{E}$ are the red experts, $\mathcal{B}\cap\mathcal{E}$ are the blue experts and $\mathcal{N}_B$ are the non-experts that the adversary wants to become blue. Note that if all nodes from $\mathcal{N}_B$ turn blue, then the adversary wins.

2. For a suitable $h = o(1)$ delete each edge e between $\mathcal{N}_B$ and $\mathcal{R}\cap\mathcal{E}$ independently with probability $(1+h(n))4\delta/(1+2\delta)$ unless e cannot be deleted because at the respective nodes already too many edges have been deleted; let us refer to the latter case as a failed deletion; we will show that whp there are no failed deletions. We will specify h in the remainder of the proof.

To prove that this strategy fulfils the requirements, we show that whp all nodes from $\mathcal{N}_B$ become blue. We write $G_n = (V_n, E_n)$. It is $p = \omega(\ln(n)/n)$, hence by Theorem 3.5.1 and by applying the union bound we obtain that whp for all $v \in V_n$

$$d(v) = (1+o(1))np \tag{3.5.9}$$

and that, before the edges are deleted,

$$|N(v)\cap\mathcal{R}\cap\mathcal{E}| - |N(v)\cap\mathcal{B}\cap\mathcal{E}| = (1+o(1))2\delta\mu np \tag{3.5.10}$$

and that whp for all $v \in \mathcal{N}_B$

$$(1+o(1))(1+h(n))\frac{4\delta}{1+2\delta}(1/2+\delta)\mu np = (1+o(1))(1+h(n))2\delta\mu np \tag{3.5.11}$$

edges between v and $\mathcal{R}\cap\mathcal{E}$ are deleted. Thus from (3.5.10) and (3.5.11) we can infer that $h = o(1)$ can be chosen such that after the deletions whp all nodes in $\mathcal{N}_B$ have more blue than red expert neighbours and therefore become blue themselves. Hence we are left to show that whp the adversary did not try to delete too many edges, i.e. that whp there are no

failed edge deletions. Using Theorem 3.5.1 and the union bound we obtain that whp each red expert has lost

$$(1+o(1))\frac{4\delta}{1+2\delta}\frac{1-\mu+2\delta\mu}{2}np = (1+o(1))\frac{2\delta(1-\mu+2\delta\mu)}{1+2\delta}np$$

edges. Therefore, using (3.5.9), whp each red expert has lost a

$$(1+o(1))\frac{2\delta(1-\mu+2\delta\mu)}{1+2\delta}$$

fraction of its edges. Because of (3.5.9) and (3.5.11) whp each node from $\mathcal{N}_B$ has lost a $(1+o(1))2\delta\mu$ fraction of its edges. As

$$2\delta\mu < \frac{2\delta(1-\mu+2\delta\mu)}{1+2\delta},$$

this shows that whp there have been no failed edge deletions.

For the complete graph on n nodes the strategy from above can be slightly modified such that the adversary wins deterministically: Instead of deleting edges randomly, from each node in $\mathcal{N}_B$ he deletes $2\delta\mu n+1$ edges to nodes in $\mathcal{R}\cap\mathcal{E}$ such that each node in $\mathcal{R}\cap\mathcal{E}$ loses the same number of edges (up to a difference of at most one edge). Then each node from $\mathcal{N}_B$ becomes blue deterministically and the adversary has not deleted too many edges. The same approach can be used to obtain a deterministic strategy for the adversary in the setting with Erdős-Rényi random graphs; however, there, the adversary still wins only whp and not deterministically due to the randomness inherent to the graph sampling. □

Proof of Theorem 3.4.5. For better readability, in this proof we will write "almost all" instead of "all but $o(n)$". We start with the lower bound for the local resilience with deletions and insertions. From (3.5.6) and (3.5.7) we know that whp almost all non-experts $v\in\mathcal{N}=\mathcal{N}^{(n)}$ have $(1+o(1))2\delta\mu pn$ more red than blue expert neighbours. We say that an edge is changed at a node v if it is either deleted or inserted at v. Thus, whp, for almost all non-experts $v\in\mathcal{N}$ a necessary condition for v to become blue is that $(1+o(1))2\delta\mu pn$ edges are changed at v. According to (3.5.5), whp, every node has degree $(1+o(1))np$. Hence, in the best case (from the adversary's perspective) the changes are distributed on the experts as uniformly as possible; i.e. ideally at each expert the same number of changes is made. However, this may not be possible: While the respective deletions can always be made, it might be the case that the adversary cannot insert so many edges as already too many edges are present; as according to (3.5.6) whp almost all non-experts $v\in\mathcal{N}$ have $(1+o(1))(1/2-\delta)\mu pn$ blue expert neighbours and because there are $(1/2-\delta)\mu n$ blue experts in total, at most $(1/2-\delta)\mu(1-p+o(p))n$ edges can be inserted between v and the blue experts. It is optimal for the adversary to distribute the changes among all experts as uniformly as possible; in particular, whp, for almost all non-experts $v\in\mathcal{N}$ that are supposed to become blue, the adversary inserts

$$\begin{aligned}&\min\left\{(1+o(1))\left(\frac{1}{2}-\delta\right)2\delta\mu pn, \left(\frac{1}{2}-\delta\right)\mu(1-p+o(p))n\right\}\\ &\quad=(1+o(1))\left(\frac{1}{2}-\delta\right)\mu n\min\{2\delta p, 1-p+o(p)\}\end{aligned}$$

edges between v and the blue experts and consequently deletes

$$\begin{aligned}&(1+o(1))2\delta\mu pn-(1+o(1))\left(\frac{1}{2}-\delta\right)\mu n\min\{2\delta p,1-p+o(p)\}\\&=(1+o(1))\left(2\delta\mu pn-\left(\frac{1}{2}-\delta\right)\mu n\min\{2\delta p,1-p\}\right)\end{aligned}$$

edges between v and the red experts. Note that the adversary loses if less than $n(1-\mu+2\delta\mu)/2$ non-experts become blue. Thus, whp, a necessary condition for the adversary to win is that there is a red expert $v^*\in\mathcal{E}\cap\mathcal{R}$ that loses at least

$$\begin{aligned}&(1+o(1))\frac{1-\mu+2\delta\mu}{2}n\frac{2\delta\mu pn-(1/2-\delta)\mu n\min\{2\delta p,1-p\}}{(1/2+\delta)\mu n}\\&=(1+o(1))\frac{1-\mu+2\delta\mu}{2}n\frac{(1/2+\delta)2\delta\mu pn+(1/2-\delta)2\delta\mu pn-(1/2-\delta)\mu n\min\{2\delta p,1-p\}}{(1/2+\delta)\mu n}\\&=(1+o(1))\frac{1-\mu+2\delta\mu}{2}n\left(2\delta p+\frac{1-2\delta}{1+2\delta}(2\delta p-\min\{2\delta p,1-p\})\right)\\&=(1+o(1))\frac{1-\mu+2\delta\mu}{2}n\left(2\delta p+\frac{1-2\delta}{1+2\delta}\max\{0,2\delta p-(1-p)\}\right)\end{aligned}$$

edges. Therefore, as whp every node has degree $(1+o(1))np$, whp v^* loses at least a

$$\begin{aligned}&(1+o(1))\frac{1-\mu+2\delta\mu}{2}\left(2\delta+\frac{1-2\delta}{1+2\delta}\max\left\{0,2\delta-\frac{1-p}{p}\right\}\right)\\&\quad=(1+o(1))\left((1-\mu+2\delta\mu)\delta+\frac{(1-\mu+2\delta\mu)(1-2\delta)}{2+4\delta}\max\left\{0,2\delta-\frac{1-p}{p}\right\}\right)\end{aligned}$$

fraction of its edges. Thus the local resilience with deletions and insertions is at least

$$(1-\mu+2\delta\mu)\left(\delta+\frac{1-2\delta}{2+4\delta}\liminf_{n\to\infty}\left(\max\left\{0,2\delta-\frac{1-p(n)}{p(n)}\right\}\right)\right)\tag{3.5.12}$$

as claimed. By adapting the adversary's strategies from the proof of Theorem 3.4.4 canonically, i.e. by distributing the changes as uniformly as possible to all experts, it is straightforward that (3.5.12) is also an upper bound for the local resilience with deletions and insertions. Note that unlike in the setting of Theorem 3.4.4, in the present setting, with the adapted strategies, we cannot infer that whp more than $n/2$ nodes of G_n turn blue, but we can infer that this holds for a suitable subsequence of $\mathcal{G}$. □

Proof of Theorem 3.4.6. Consider the subset sum problem (SUBS) that is NP-hard; it is defined as follows.
Input: A finite set $A=\{m_1,m_2\ldots,m_n\}\subseteq\mathbb{N}$ and a natural number $K\in\mathbb{N}$ with $K\leq\sum_{1\leq i\leq n}m_i$.
Output: A subset $B\subseteq A$ such that

$$\sum_{m\in B}m=\min_{C\subseteq A}\left\{\sum_{m\in C}m\mid\sum_{m\in C}m\geq K\right\},$$

i.e. a subset of A that sums up to at least K but such that the sum is as close to K as possible. We will reduce (SUBS) to (SAP). To do this, let $\{m_1, m_2, \ldots, m_n\} \subseteq \mathbb{N}$, $K \in \mathbb{N}$ be an instance of (SUBS). We construct a (SAP) instance as follows: We define the respective graph as the union of disjoint cliques; in particular, there are $K/2$ cliques of size $2(m_1 + m_2 + \cdots + m_n)$ and n cliques $H_1, \ldots, H_n$ of sizes $m_1, \ldots, m_n$ respectively. Further we specify that the strong adversary has to choose K nodes as red experts and $K/2$ nodes as blue experts. By construction, each optimal solution for this instance of (SAP) has the following two properties.

- In each of the $K/2$ large cliques there is exactly one blue and no red expert. In particular, all red experts are in the smaller cliques.
- The red experts are distributed on the n smaller cliques such that the sum of the sizes of those smaller cliques which contain at least one red node is minimal.

Thus, if we assume that we have such an optimal solution for the constructed instance of (SAP), then the following solution for the original (SUBS) instance is optimal: Choose $B \subseteq A$ such that for $i \in \{1, \ldots, n\}$ we have $m_i \in B$ if and only if H_i contains at least one red expert in the optimal (SAP) solution. □

3.6 Outlook

One direction for future research is to prove local resilience results also for further graph classes like, e.g., graphs with sufficiently good expansion properties. Another possible direction is to investigate models where the dissemination has not finished after one round but proceeds iteratively over several rounds. Such an iterative model is already briefly considered in [5]. Also many other generalisations are conceivable. For example, it would be interesting to consider a setting where some experts are more influential than others; this could be realised by assigning weights to the experts (e.g. according to some probability distribution) and then, when counting the expert neighbours, each expert contributes proportional to its weight. Moreover, the concept of "robustness against a certain adversary" could be generalised to "α-robustness against a certain adversary", where α-robustness means that, whp, an α fraction of all nodes will be red at the end of the dissemination process.

Chapter 4

Vehicle Routing with Drones

4.1 Introduction

Drones for deliveries in the logistics sector — what seemed like science fiction ten years ago now starts to become reality. Deutsche Post's Paketkopter[1], Amazon's project PrimeAir[2] and Mercedes' Vision Van[3] are only few examples of global companies doing research on how to use drones for package delivery. For example, Mercedes' Vision Van combines the use of a truck with the use of drones where the drones are supposed to be utilised for the so-called last mile delivery. This seems promising as the last mile often is the bottleneck that avoids faster delivery when using a truck only.

This motivates the following optimisation problem: Suppose that you have a fleet consisting of two types of vehicles, namely trucks and drones, to deliver packages to given destinations. Drones and trucks may operate according to different metrics. Drones can deliver one package at a time and afterwards have to return to a truck and charge before they can deliver another package. Now the task is to assign a tour to each truck and drone through suitable package destinations such that the packages are delivered as fast as possible. There are several possibilities to make "as fast as possible" precise. For example, one possible objective is to minimise the completion time; another objective is to minimise the average delivery time, i.e. the average time a customer has to wait for his delivery. An interesting aspect of this problem is that it combines different core problems of combinatorial optimisation. On the one hand, there are tours that tendentiously should be short which relates to the Travelling Salesman Problem (TSP); indeed, for the special case with one truck and without drones, it is the TSP. On the other hand, scheduling aspects are relevant since often a vehicle has to wait for another vehicle if they are supposed to travel together.

The remainder of this chapter is organised as follows. We provide a summary of the existing literature in Section 4.2. Afterwards, in Section 4.3, we first contour the vehicle routing with drones (VRD) model less formally to illustrate the core ideas. Then, in Section 4.4, we provide a precise and formal definition of the model; due to the synchronicity constraints inherent to the problem, this contains some subtleties; moreover, we prove an

[1] Cf. `dpdhl.com/de/presse/specials/paketkopter` (Retrieval date: 12th April 2018).
[2] Cf. `amazon.com/Amazon-Prime-Air/b?node=8037720011` (Retrieval date: 12th April 2018).
[3] Cf. `daimler.com/innovation/specials/vision-van/` (Retrieval date: 12th April 2018).

equivalent characterisation of the feasibility of a solution. In Section 4.5, we provide a short introduction to local search algorithms in general. Afterwards, in Subsection 4.6.1, we introduce a local search algorithm to solve the VRD problem and, in Subsection 4.6.2, we look at computational results.

I worked on this subject together with Elisabeth Kraus. She will publish different but thematically related results in her doctoral thesis which were also obtained during this joint work; portions of this chapter are part of our article ([24]).

Contribution The VRD problem (up to some rather small differences) was already introduced in [107]. While the description in [107] is relatively clear and intuitive, it is not very formal. We close this gap and provide a precise and formal definition. Due to the synchronicity constraints inherent to the problem this contains some subtleties and is slightly laborious. This is all the more the case as in contrast to the setting in [107], we allow drones to travel on different trucks during their tours which complicates a formal definition of the synchronicity constraints. Afterwards we prove an alternative characterisation of the feasibility of a solution that relates the feasibility to the absence of cycles in a suitably defined multidigraph that fulfil a certain property. We also provide a simple local search algorithm to solve the VRD problem; empirical results show that it runs very fast and outperforms a canonical Greedy algorithm. Its capabilities and limitations are discussed in the paragraph "Discussion of VRD-LOC" in Subsection 4.6.1.

4.2 Related Literature

The problem we are dealing with is located in the field of vehicle routing problems. The common idea of vehicle routing problems is that there is a fleet of vehicles that has to deliver a set of packages to certain positions as fast as possible. Usually there are also some kind of constraints for the vehicles, e.g. each vehicle cannot deliver more than a fixed number of packages; also innumerable other variants of vehicle routing problems are considered in the literature. For literature in the context of vehicle routing problems see, e.g., [102, 72, 73, 101, 92, 59].

While the consideration of the symbiosis of trucks and drones has started only very recently, some related models have already been investigated earlier. The key innovation of the new models with trucks and drones is that they have strong synchronicity requirements: Not only tours have to be assigned but these tours do interact and hence have to be synchronised. Now we will present some work that shares some characteristics with models with trucks and drones. Besides the summary that we provide here, we also recommend the literature reviews in the article by Murray and Chu ([82]) and the article by Agatz et al. ([4]). Many of the sources discussed here are treated there as well.

One model to mention is the Covering Salesman Problem which was introduced by Current and Schilling in [22] and which has been generalised by Golden et al. in [60]. In the Covering Salesman Problem a weighted graph is given and each node in the graph has assigned a (possibly empty) subset of the other nodes that it "covers". The objective is to find a tour of minimum weight through a subset of the nodes of the graph such that every node is either part of the tour or covered by a node that is part of the tour. This problem, in

some aspects, resembles the scenario where a drone supports a truck: The truck does not need to visit all nodes, it suffices if it visits a subset of the nodes from where the drone can do the remaining work. However, variants considering the use of a drone to support the truck involve some subtle points which make them substantially more complicated: In the Covering Salesman Problem, there are no synchronicity constraints since there is only one vehicle. Even if the Covering Salesman Problem was canonically generalised to a version with several trucks, then these trucks still would not interact and hence there still would not be any synchronicity constraints. In contrast, when a drone and a truck interact, sometimes one vehicle has to wait for another vehicle. As a consequence, the objective value (several choices are possible, e.g. the completion time) of a solution cannot be computed by looking at all tours isolated. Hence the Covering Salesman Problem has some similarity with the version with trucks and drones, but the latter seems more subtle.

Another model that is closely related to the Covering Salesman Problem is the Close Enough Travelling Salesman Problem (CETSP), see e.g. [98, 36, 62, 9, 109]. In the CETSP, points in the Euclidean plane which have to be visited by a truck (target points) and one starting point (i.e. the depot) where the truck tour starts and ends are given. Each target point has assigned a connected compact subset of the Euclidean plane containing the target point. Now one has to find a tour in the Euclidean plane such that the tour has at least one common point with each of the compact subsets assigned to the target points; i.e. visiting the compact subset of a target point is "close enough" to consider the point as visited. Like the Covering Salesman Problem, the CETSP resembles the variant with truck and drone as in both problems the truck does not need to visit all nodes. However, like the Covering Salesman Problem, also the CETSP lacks the presence of synchronicity constraints.

While the models we considered so far did not incorporate aspects of synchronicity, vehicle routing problems that deal with synchronicity have been investigated. A summary of such variants of vehicle routing problems that require synchronisation is provided in the survey article by Drexl ([39]).

One of the most extensively investigated vehicle routing problems with synchronicity constraints is the Truck and Trailer Routing Problem (TTRP) introduced by Chao in [15]; for further work on the TTRP see also e.g. [75, 96, 38, 86, 32, 104, 105] and for a closely related problem see [40]. In the TTRP there are trucks and trailers that have to serve customers. A truck can move by itself and pull a trailer, in contrast, a trailer can only be pulled by a truck but not move by itself. There are two kinds of customers: Customers of the first kind can be served by a truck alone or by a truck pulling a trailer. Customers of the second kind can only be served by trucks without trailer (e.g. because the streets are too small for trucks pulling trailers). Also each customer has a certain demand of the good which is delivered (we do not distinguish between different goods) and each truck and each trailer has a certain capacity. Note that if the trucks had an unlimited capacity, then the trailers would be useless. Trailers can be parked at those customers that can be served by trucks pulling trailers. One can also allow that trailers are shared by different trucks, i.e. one truck parks a trailer at some customer where it is picked up by another truck. Now one has to find tours such that each customer obtains his demand of the good and the mentioned constraints are satisfied. The objective value of a solution can be defined in various ways, e.g. as the total distance or as the completion time or as some more complicated expression. In the TTRP, synchronisation is necessary; but there are also some differences compared to the

setting with trucks and drones: The trailers, which have similarities with the drones, cannot move on their own; hence, as already mentioned earlier, they are only useful when there are capacity constraints for the trucks; in contrast, drones affect the solution also without any capacity constraints of the trucks. Indeed, in this work, we will not consider capacity constraints for the trucks. So, while the TTRP shares the characteristic of synchronicity constraints, it differs in other important aspects from the variant with trucks and drones.

Another model with synchronicity constraints is investigated by Lin ([74]). In that model there are heavy and light vehicles. The heavy vehicles can carry light vehicles. Also both types of vehicles can move on their own and both types of vehicles can pick up (or deliver) packages at the customers. Differences compared to the model with drones are that a light vehicle may depart from a heavy vehicle only one time; in contrast, a drone may depart arbitrarily often from a truck. Furthermore, unlike the light vehicles considered in [74], drones can only carry one package at a time which makes it even more important that they are allowed to depart from a truck more than one time.

Also the vehicle routing problem with time windows and multiple service workers (cf. the articles of de Grancy and Reimann ([27, 28])) shares some characteristics with the setting with trucks and drones. There the truck drives to a position and from that position the service workers in that truck depart, deliver packages and return to the truck. Then the truck drives to another position, the service workers depart and so on. The service workers in this model share some characteristics with the drones; however a major difference is that the truck cannot move while the service workers are delivering packages. Hence, the aspect of synchronicity is not strongly present. Moreover, there the truck cannot deliver packages by itself.

The consideration of the collaboration of drones and trucks for parcel delivery has started only very recently. Murray and Chu introduced a respective variant of the TSP with drone in [82]. They call the problem the Flying Sidekick Travelling Salesman Problem (FSTSP). Other work related to the topic is [4, 107, 88, 63, 91, 44, 16, 37, 78, 77, 11]. In the following subsections, we briefly summarise four of these articles (namely [82, 4, 107, 88]), afterwards we shortly highlight which topics some of the other work deals with.

4.2.1 "The Flying Sidekick Travelling Salesman Problem: Optimization of Drone-assisted Parcel Delivery" by Murray and Chu ([82])

Murray and Chu investigated the synergy of drones and one truck ([82]). Two different models are introduced; for both models, heuristics and mixed integer linear programs are provided. The Flying Sidekick Travelling Salesman Problem (FSTSP) is one of these two models; in the FSTSP, one truck and one drone (that has a restricted flight endurance) are supposed to deliver packages. Some packages cannot be carried by a drone (e.g. because they are too heavy). The objective is to minimise the time until the latest return of the truck or drone after the delivery of all packages, i.e. the completion time. They provide an integer linear programming formulation which they try to solve for modest-sized instances (10 packages) with the solver Gurobi. However, even for these comparatively small instances, Gurobi did not find provably optimal solutions within 30 minutes. This motivates the use of heuristics.

They provide a heuristic that starts with a solution where the truck delivers all packages on its own. Then in each iteration several changes (using also the drones) are investigated and the change with the highest saving is applied. Computational results for this local search heuristic are provided. The second model they propose contains several drones (and still one truck). However, the drones do not interact with the truck any more, but can only start from the depot, deliver a package within their reach and then return to the depot. Here, unlike in the FSTSP, drones may leave the depot several times. One has to find a truck tour that covers at least the packages which cannot be served by the drones (e.g. because of their weight or because they are too far away from the depot). The packages that are not delivered by the truck have to be served by the drones from the depot directly. In order to do this, a schedule for the drones has to be found. The objective is the same as for the FSTSP. For this model a heuristic is proposed as well. It starts with a solution in which all packages that can be delivered by drones in fact are delivered by drones; the remaining packages are delivered by the truck. Then a local search approach is applied to improve this solution. Considering multiple trucks and drones is suggested as a topic for future research (which is what we do in this thesis).

4.2.2 "Optimization Approaches for the Travelling Salesman Problem with Drone" by Agatz, Bouman and Schmidt ([4])

In [4], Agatz et al. investigate the Travelling Salesman Problem with Drone (TSP-D) which is very similar to the FSTSP from [82]. One difference is that in [4] the truck and the drone follow the same metric (up to a constant multiplicative factor); in contrast, in [82] they can travel according to different metrics, e.g. the drone flies according to the Euclidean metric while the truck moves according to the Manhattan metric. After introducing the problem, some theoretical aspects are considered, e.g. it is shown that if the truck has speed 1 and the drone has speed α, then the objective value of an optimal solution for the truck only version (i.e. the TSP) is at most $1+\alpha$ times the optimal objective value for the TSP-D; an example where this approximation factor occurs is provided which shows that the factor of $1+\alpha$ is tight. They note that a consequence of their bound is that the TSP-D is constant-factor approximable in polynomial time: By using the Christofides algorithm an approximation factor of $1.5+1.5\alpha$ is obtained; they further improved this factor. Afterwards a mixed integer programming formulation is provided. Then heuristic approaches are developed, based on so called route first cluster second procedures (cf. Beasley's article ([8])). Furthermore, a computational study is performed. Moreover, it is suggested to investigate the setting with multiple trucks and drones.

4.2.3 Vehilce Routing with Drones — Work by Wang, Poikonen and Golden ([107, 88])

In [107] the Vehicle Routing Problem with Drones (VRPD) is introduced and several worst-case bounds are obtained. A fleet of trucks is considered where each truck is able to carry several drones. In contrast to the model that we consider in this thesis, in their model a drone never changes the truck during its tour; a respective generalisation is suggested in

the outlook of [88]. Among other insights, the bounds obtained in [107] provide information about the maximum gain from using drones additionally to trucks compared to truck only variants. The model introduced in [107] is very similar to the model that we will consider; for a summary of the new contributions in this thesis (also compared to the model formulation in [107]) see the paragraph "Contribution" in Section 4.1.

In their follow-up article [88] the results from [107] are generalised; for example, limited battery life of the drones is taken into consideration, settings when trucks and drones follow different metrics are covered and a more general approach to define the objective function is introduced. Also relations to other problems are investigated.

4.2.4 Further Related Work

In the past few years the collaboration of trucks and drones for parcel delivery has been actively investigated. Hence a lot of further work on related topics exists. For several of these contributions we provide a very brief summary of aspects that were covered.

In [11], bin Othman et al. introduce a model where a truck and a drone deliver packages to customers. There are certain rendezvous points where the drone and the truck can meet and there are customer positions at which parcels have to be delivered. They introduce NP-hard problems in this setting and provide polynomial time approximation algorithms for these problems.

In [78], Mathew et al. consider a model where a truck collaborates with a drone to deliver packages. As in [11], the nodes where the truck and the drone can interact are different from the positions at which packages have to be delivered. However, here the truck itself does not deliver packages at all but the drone delivers all packages. The problem is shown to be NP-hard and several solution approaches are suggested; one such solution approach reduces the problem to a generalised TSP for which advanced solving procedures exist.

In [63], Ha et al. investigate a model similar to the model proposed in [82] but different objectives are taken into consideration. Two approaches to solve the problem are developed and tested computationally. The approaches start with a TSP tour and convert it into a solution for the respective problem with drone.

In [91], Ponza investigates a model similar to the model from [82] and proposes a simulated annealing approach to solve it.

In [44], Ferrandez et al. study the cooperation of drones with a truck for parcel delivery. The drones have to return to the truck after each delivery. A solution approach is proposed which is based on k-means clustering; when the truck stops, drones deliver packages to nearby positions that were assigned to this stop using k-means clustering. Computational results are provided.

4.3 Informal Description of the Model

Before we introduce the Vehicle Routing with Drones (VRD) model formally we informally sketch the main characteristics of the model. Note that a very similar model has already been introduced in [107]; for a summary of the new contribution in this thesis, see the paragraph "Contribution" in Section 4.1 and for a brief summary of [107] see Subsection

4.2.3. An instance of the VRD problem consists of the number of trucks, the number of drones and, for each two packages P_1, P_2 (or a package and the depot), the travel time from the destination of P_1 to the destination of P_2 for both, trucks and drones.

Given an instance, we want to construct a feasible solution. A (not necessarily feasible) solution is given by assigning each vehicle (vehicle refers to truck or drone) a tour that starts at the depot, visits some package destinations and returns to the depot; additionally, when a drone travels between two positions (we call this an edge), we have to define whether it travels on a truck or whether it flies on its own. Not every solution is feasible, for feasibility the following constraints have to be satisfied.

- Each package has to be delivered.
- A drone can carry at most one package; a truck can carry an arbitrary number of packages and drones. Hence the delivery time of a package is the arrival time at the respective destination of the first vehicle that is not a drone that has already delivered a package on its current flight.
- If a drone is assumed to travel an edge on a truck then there must be a truck tour that contains the edge between the two respective package destinations as well.
- Drones can only fly along two consecutive edges without recharging. They can recharge by travelling one edge on a truck.
- As drones may travel on trucks, sometimes a vehicle has to wait for another vehicle. There must not be situations where none of the vehicles that have not already returned to the depot can continue because each of these vehicles is waiting for another vehicle (cf. Figure 4.1 for two examples).
- Each node is only visited by vehicles that either deliver a package at that node or interact with a vehicle that delivers a package at that node (e.g. a drone lands on a truck or a truck picks up a drone).
- No vehicle may visit a package destination more than once. All vehicles start and end their tours at the depot; if a vehicle arrives at the depot, then it cannot leave again.

Two aspects in which our model differs from the model in [107] are that in our model, drones are allowed to travel on different trucks during their tours and that they have to travel an edge on a truck to recharge. The former aspect makes it more complicated to define the feasibility of a solution, as situations may occur where a vehicle has to wait for another vehicle forever, because this vehicle for its part has to wait for another vehicle and so on.

There are several possibilities for an objective function. One is the completion time, i.e. the time it takes until all vehicles have returned to the depot. Another choice is the average time until a package has been delivered. In either case it is crucial to also consider the waiting times of the vehicles instead of simply adding up the pure travel times.

4.4 Formal Definition of the Model

Now we introduce the VRD model formally. In order to do this, we need the following graph theoretic definitions.

Definition 4.4.1 (Multidigraph)**.** *A multidigraph G is a triple (V, E, ψ) where V and E are finite sets and ψ is a function $\psi : E \to V \times V$. The sets V and E are called node set of G and edge set of G respectively. If ψ is injective, then G is a digraph. We call $|V|$ the size of G. For an edge $e \in E$ with $\psi(e) = (v_1, v_2) \in V \times V$. We refer to the first or second entry of the tuple $\psi(e)$ by $(\psi(e))[1]$ or $(\psi(e))[2]$ respectively, i.e. $(\psi(e))[1] = v_1$ and $(\psi(e))[2] = v_2$.*

Definition 4.4.2 (Subgraph of a multidigraph)**.** *Let $G_1 = (V_1, E_1, \psi_1)$ and $G_2 = (V_2, E_2, \psi_2)$ be multidigraphs. We say that G_1 is a subgraph of G_2 and write $G_1 \subseteq G_2$ if $V_1 \subseteq V_2$, $E_1 \subseteq E_2$ and for all $e \in E_1$ it holds $\psi_2(e) = \psi_1(e)$.*

Definition 4.4.3 (Cycle / circuit in a multidigraph)**.** *Let $G = (V, E, \psi)$ be a multidigraph. Let $e_1, e_2 \in E$. Set $(v, w) := \psi(e_1)$ and $(x, y) := \psi(e_2)$. If $w = x$, then we say e_2 follows e_1. Let $k \in \mathbb{N}$. A subgraph $C \subseteq G$, $C = (V_C, E_C, \psi_C)$, is called a circuit of size k in G if the edges in E_C can be indexed as $E_C = \{e_1, \dots, e_k\}$ such that for $1 \leq i \leq k-1$ the edge e_{i+1} follows the edge e_i and also e_1 follows e_k. If additionally for each $z \in V_C$ there is exactly one edge $e \in E_C$ with $(\psi_C(e))[1] = z$, then C is a cycle of size k in G.*

Remark 4.4.4. *Note that Definition 4.4.3 in particular defines cycles and circuits in digraphs. In this case, circuits (and in particular cycles) can be represented particularly easily: Let $G = (V, E, \psi)$ be a digraph, in particular ψ is injective. Let $C = (V_C, E_C, \psi_C)$ be a circuit of size k in G. Then we can write C as a vector with entries $u_1, \dots, u_k \in V_C$, i.e. $C = (u_1, \dots, u_k)$. While the choice of the respective vector is not unique, the converse is true: Given such a vector, the respective circuit is uniquely determined. Hence, to consider a circuit in a digraph, sometimes we will use this vector representation. Note that we did not require that the nodes $u_1, \dots, u_k$ are pairwise disjoint. Analogously, we can describe a circuit as a vector with the respective edges as entries.*

Definition 4.4.5 (Complete digraph)**.** *Let $n \in \mathbb{N}$ and let $V = \{v_1, v_2, \dots, v_n\}$. Let $E = V \times V$ and let $\psi : E \to V \times V$ be the identity function. Then $K_n = (V, E, \psi)$ is called the complete digraph on n nodes. We write E_{K_n} for its edge set.*

Next we formally define what an instance of the VRD model is (Definition 4.4.6), what a solution is (Definition 4.4.7) and the feasibility of a solution (Definitions 4.4.9, 4.4.10, 4.4.13, 4.4.14, 4.4.15).

Definition 4.4.6 (Instance)**.** *Let $n \in \mathbb{N}$. An instance of size n of the VRD problem is a quadruple $(n_t, n_d, D_t, D_d) \in \mathbb{N} \times \mathbb{N} \times (\mathbb{R}_{\geq 0}^{n+1} \times \mathbb{R}_{\geq 0}^{n+1})^2$.*

Intuitively, n_t and n_d denote the numbers of trucks and drones respectively, the entry in the ith row and jth column of D_t, i.e. $D_t[i, j]$, denotes the time a truck needs to drive from the ith to the jth destination; the $(n+1)$st row and column correspond to the depot. D_d analogously describes the flying times of a drone.

Definition 4.4.7 (Solution)**.** *Let $n \in \mathbb{N}$ and let (n_t, n_d, D_t, D_d) be an instance of size n. A solution for this instance is an $(n_t + 2n_d)$-tuple $(T_1^{(t)}, \dots, T_{n_t}^{(t)}, T_1^{(d)}, \dots, T_{n_d}^{(d)}, b_1, \dots, b_{n_d})$ where, for $1 \leq j \leq n_t$ and $1 \leq k \leq n_d$, $T_j^{(t)} = (V_j^{(t)}, E_j^{(t)})$ and $T_k^{(d)} = (V_k^{(d)}, E_k^{(d)})$ are cycles in the complete digraph with node set $\{v_1, \dots, v_{n+1}\}$, such that v_{n+1} is contained in each cycle; b_k is a binary function $b_k : E_k^{(d)} \to \{0, 1\}$.*

Notation 4.4.8. *If we consider an instance I of size n, then this implicitly defines n_t, n_d, D_t, D_d and similarly, if we consider a solution S for an instance I, then this implicitly defines $T_1^{(t)}, \dots, T_{n_t}^{(t)}, T_1^{(d)}, \dots, T_{n_d}^{(d)}, b_1, \dots, b_{n_d}$ and also implicitly defines, for $1 \leq j \leq n_t$ and $1 \leq k \leq n_d$, the node and the edge sets $V_j^{(t)}, E_j^{(t)}$ and $V_k^{(d)}, E_k^{(d)}$ such that $T_j^{(t)} = (V_j^{(t)}, E_j^{(t)})$ and $T_k^{(d)} = (V_k^{(d)}, E_k^{(d)})$. Moreover, we write $\mathcal{T}^{(t)} = \{T_1^{(t)}, \dots, T_{n_t}^{(t)}\}$, $\mathcal{T}^{(d)} = \{T_1^{(d)}, \dots, T_{n_d}^{(d)}\}$ and $\mathcal{T} = \mathcal{T}^{(t)} \cup \mathcal{T}^{(d)}$. When we consider an instance of size n, then we always denote the nodes of a complete digraph of size $n+1$ with $\{v_1, \dots, v_{n+1}\}$. Vice versa, if we denote a node with v_i then we imply that we are considering a node of the complete digraph of size $n+1$ where n denotes the size of the instance that we are considering. Intuitively, $v_1, \dots, v_n$ correspond to the package destinations and v_{n+1} corresponds to the depot. To refer to the driving time from a node x to a node y, in slight abuse of notation, we sometimes write $D_t[x, y]$ (and analogously for D_d).*

The cycles in $\mathcal{T}^{(t)}$ represent the truck tours, the cycles in $\mathcal{T}^{(d)}$ represent the drone tours and the binary functions indicate whether the respective drone is carried by a truck on the respective edge. Therefore sometimes we will refer to the cycles as truck (drone) cycles or truck (drone) tours.

One necessary condition for the feasibility of a solution is that it satisfies the visiting constraint specified in Definition 4.4.9.

Definition 4.4.9 (Visiting constraint)**.** *Let $n \in \mathbb{N}$, let I be an instance of size n and let S be a solution for I. S satisfies the visiting constraint, if all following conditions are fulfilled.*

- *It is*
$$\bigcup_{j=1}^{n_t} V_j^{(t)} \cup \bigcup_{k=1}^{n_d} V_k^{(d)} = \{v_1, \dots, v_{n+1}\}.$$
- *Assume that there are $1 \leq i \leq n$ and $1 \leq k \leq n_d$ such that there are $u, w \in \{v_1, \dots, v_n, v_{n+1}\}$ such that $(u, v_i), (v_i, w) \in E_k^{(d)}$. Further assume that $b_k(u, v_i) = b(v_i, w) = 0$. Then*
$$v_i \notin \bigcup_{j=1}^{n_t} V_j^{(t)} \cup \bigcup_{l=1, l \neq k}^{n_d} V_l^{(d)}.$$
- *Assume that there are $1 \leq i \leq n$ and $1 \leq j \leq n_t$ such that $v_i \in V_j^{(t)}$. Then*
$$v_i \notin \bigcup_{l=1, l \neq j}^{n_t} V_l^{(t)}.$$
- *Let $k, l \in \{1, \dots, n_d\}$. Consider $T_k^{(d)} = (w_1, \dots, w_s)$ and $T_l^{(d)} = (x_1, \dots, x_t)$ where $w_1 = x_1 = v_{n+1}$. Assume that there are $2 \leq i \leq s-1$ and $2 \leq j \leq t-2$ such that $w_i = x_j$ and $w_{i+1} = x_{j+1}$; further assume that $b_k(w_i, w_{i+1}) = b_l(x_j, x_{j+1}) = 1$. Then $\{x_{j+2}, \dots, x_t\} \cap \{w_1, \dots, w_{i-1}\} = \emptyset$.*

The first point of the visiting constraint assures that every package is delivered. The second point assures that if a drone flies to a package destination and also flies away from there, then no other vehicle visits this destination. The third point assures that each package destination is visited by at most one truck. The second and the third point together assure that each node is only visited by vehicles that either deliver a package at that node or interact with a vehicle that delivers a package at that node. This is motivated by the fact that residents probably do not want to have vehicles waiting in front of their homes and thus occupying parking space if these vehicles do not even deliver a package to them. The fourth condition excludes some pathological solutions; in particular, it rules out solutions of the following kind: Truck 1 arrives at node v carrying drone 1. Then drone 1 flies away while truck 1 keeps waiting at v. Note that drone 1 can never return to node v. Then, later during its tour, drone 1 travels an edge on truck 2 together with drone 2. Afterwards, drone 2 flies to node v and continues its tour carried by truck 1. Since drones 1 and 2 have travelled an edge on truck 2 together, after this edge, they are interchangeable. Hence, as drone 1 is not allowed to return to node v, the same should hold for drone 2; the fourth condition assures this.

The following constraint assures that drones can fly at most along two consecutive edges before they have to recharge.

Definition 4.4.10 (Charging constraint)**.** *Let $n \in \mathbb{N}$ and let I be an instance of size n. Let S be a solution for I. S satisfies the charging constraint if the following holds. For any $1 \leq k \leq n_d$ consider $T_k^{(d)} = (V_k^{(d)}, E_k^{(d)})$. Let $u, v, w, x \in V_k^{(d)}$ and $v, w \neq v_{n+1}$; assume that $(u, v), (v, w), (w, x) \in E_k^{(d)}$. Then $b_k(u, v) + b_k(v, w) + b_k(w, x) > 0$.*

In Definition 4.4.11, we define functions C_t and C_d that have the following intuitive meaning. For a given solution, for an edge e of the complete digraph of size $n+1$ (when we consider an instance of size n), $C_t(e)$ states which (if any) truck drives along this edge and $C_d(e)$ gives the drones that are carried by a truck on edge e (if any).

Definition 4.4.11 (Edge functions)**.** *Let $n \in \mathbb{N}$ and let I be an instance of size n. Let S be a solution for I that satisfies the visiting constraint. Define $C_t : E_{K_{n+1}} \to \{\emptyset, \{1\}, \{2\}, \ldots, \{n_t\}\}$ and $C_d : E_{K_{n+1}} \to \mathcal{P}(\{1, 2, \ldots, n_d\})$ as follows.*

- *For an edge $e \in E_{K_{n+1}}$, if there is an $1 \leq j \leq n_t$ such that $e \in E_j^{(t)}$, set $C_t(e) := \{j\}$, otherwise set $C_t(e) = \emptyset$. Note that if such a j exists, it is unique as the visiting constraint is satisfied.*

- *For an edge $e \in E_{K_{n+1}}$, for $1 \leq k \leq n_d$, it is $k \in C_d(e)$ if and only if $e \in E_k^{(d)}$ and $b_k(e) = 1$.*

We call C_t and C_d the edge functions for the solution S.

Given a solution for an instance, we consider an edge e. If there are drones that travel on a truck along this edge, then we say that the respective vehicle tours are interconnected on e; Definition 4.4.12 formalises this.

Definition 4.4.12 (Interconnected cycles)**.** *Let $n \in \mathbb{N}$ and let I be an instance of size n. Let S be a solution for I. Let $e \in E_{K_{n+1}}$ and let C_t and C_d denote the edge functions for the solution S. We define the set*

$$IC(e,S) := \bigcup_{k \in C_d(e)} \{T_k^{(d)}\} \cup \bigcup_{j \in C_t(e)} \{T_j^{(t)}\}$$

and call it the set of the interconnected cycles of the edge e; note that $IC(e,S)$ can be the empty set.

Definition 4.4.13 assures that if a drone is carried by a truck along an edge according to its binary function, then there is indeed a truck that drives along this edge.

Definition 4.4.13 (Drone carrying constraint)**.** *Let $n \in \mathbb{N}$ and let I be an instance of size n. Let S be a solution for I. Let C_t and C_d be the edge functions of S. We say that S satisfies the drone carrying constraint if for any $e \in E_{K_{n+1}}$ the following holds. If $C_d(e) \neq \emptyset$ then $C_t(e) \neq \emptyset$.*

The following constraint assures that no situations occur where one vehicle has to wait forever for another vehicle as this vehicle also cannot continue, as it also waits for another vehicle. This definition still contains an iterative process; hence it is desirable to also provide a non-iterative characterisation of consistency. This is done in Theorem 4.4.24.

Definition 4.4.14 (Consistency constraint)**.** *Let $n \in \mathbb{N}$ and let I be an instance of size n. Let S be a solution for I. Consider the following round based process. In the first round in each of the cycles in $\mathcal{T}$ we mark the edge leaving v_{n+1}. Consider an edge $e \in \bigcup_{1 \leq j \leq n_t} E_j^{(t)} \cup \bigcup_{1 \leq k \leq n_d} E_k^{(d)}$. Let $T \in \mathcal{T}$ and assume that e is an edge of T. We say that e is ready within T if it follows an edge in T that is already marked. If $IC(e,S) \neq \emptyset$, then we say that e is ready within $IC(e,S)$ if e is ready within every cycle in $IC(e,S)$. In each round for all $T \in \mathcal{T}$, $T = (V,E)$, and all $e \in E$ we do the following: If $T \notin IC(e,S)$ and e is ready within T, then we mark e in T; if $T \in IC(e,S)$ and e is ready within $IC(e,S)$ then we mark e within T. Markings of this round are only applied at the end of the round, i.e. new markings of this round only have an effect for subsequent rounds. S satisfies the consistency constraint if each edge of each cycle $T \in \mathcal{T}$ becomes marked by this process.*

Definition 4.4.15 specifies under which conditions a solution is feasible or almost feasible.

Definition 4.4.15 ((Almost) feasible solution)**.** *Let $n \in \mathbb{N}$ and let I be an instance of size n. A solution for the instance I is called almost feasible if it satisfies the visiting, the charging and the drone carrying constraints. If it additionally satisfies the consistency constraint, then it is called feasible. Note that every feasible solution is also almost feasible.*

Figure 4.1 shows the relevant parts of two solutions that are almost feasible but do not satisfy the consistency constraint. The second example provided in Figure 4.1 cannot be repaired as easily as the first.

Next we define an equivalence relation on the almost feasible solutions of an instance; intuitively, this is done because if two drones travelled together on a truck for an edge, then the remainders of their tours are interchangeable; also the order of the truck and drone tours in the solution is not important. Hence it suffices to consider the equivalence classes that we introduce in Definition 4.4.16.

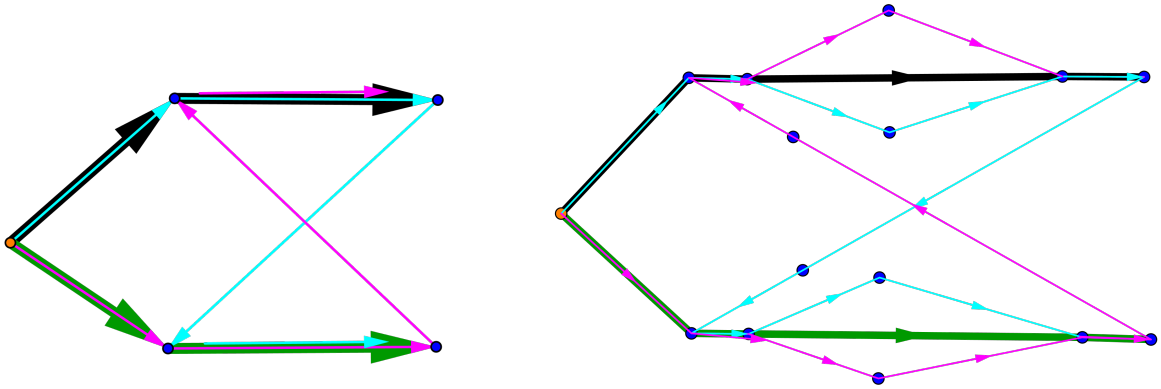


Figure 4.1: Parts of solutions that violate the consistency constraint; trucks 1 and 2 are indicated in black and green respectively, drones 1 and 2 are indicated in cyan and pink respectively, the depot is indicated in orange.

Definition 4.4.16 (Equivalent solutions)**.** *Let $n \in \mathbb{N}$ and let I be an instance of size n. Let S be a solution for I that is almost feasible. Let $k, l \in \{1, \dots, n_d\}$. Consider $T_k^{(d)} = (w_1, \dots, w_s)$ and $T_l^{(d)} = (x_1, \dots, x_t)$ where $w_1 = x_1 = v_{n+1}$. Assume that there are $1 \leq i \leq s-1$ and $1 \leq j \leq t-1$ such that $w_i = x_j$ and $w_{i+1} = x_{j+1}$ and assume that $b_k(w_i, w_{i+1}) = b_l(x_j, x_{j+1}) = 1$. Define the two cycles $\tilde{T}_k^{(d)} = (w_1, \dots, w_i, x_{j+1}, \dots, x_t)$ and $\tilde{T}_l^{(d)} = (x_1, \dots, x_j, w_{i+1} \dots, w_s)$. Also define $\tilde{b}_k$ by*

$$\begin{aligned}
&\tilde{b}_k(w_r, w_{r+1}) = b_k(w_r, w_{r+1}) \text{ for } r \in \{1, \dots, i-1\},\\
&\tilde{b}_k(w_i, x_{j+1}) = 1 \text{ and}\\
&\tilde{b}_k(x_r, x_{r+1}) = b_l(x_r, x_{r+1}) \text{ for } r \in \{j+1, \dots, t-1\}.
\end{aligned}$$

Similarly define $\tilde{b}_l$ by

$$\begin{aligned}
&\tilde{b}_l(x_r, x_{r+1}) = b_l(x_r, x_{r+1}) \text{ for } r \in \{1, \dots, j-1\},\\
&\tilde{b}_l(x_j, w_{i+1}) = 1 \text{ and}\\
&\tilde{b}_l(w_r, w_{r+1}) = b_k(w_r, w_{r+1}) \text{ for } r \in \{i+1, \dots, s-1\}.
\end{aligned}$$

Define the solution $\tilde{S}$ for I by replacing $T_k^{(d)}$ with $\tilde{T}_k^{(d)}$, $T_l^{(d)}$ with $\tilde{T}_l^{(d)}$, b_k with $\tilde{b}_k$ and b_l with $\tilde{b}_l$; since S is almost feasible (in particular, also the fourth condition of the visiting constraint is fulfilled), we obtain that $\tilde{S}$ is almost feasible, too; moreover $\tilde{S}$ is feasible if and only if S is feasible. We say S and $\tilde{S}$ are equivalent solutions. Additionally we consider solutions as equivalent that just differ in the order of the tours, i.e. S is equivalent to $\bar{S}$ where $\bar{S} = (T_{\sigma(1)}^{(t)}, \dots, T_{\sigma(n_t)}^{(t)}, T_{\varphi(1)}^{(d)}, \dots, T_{\varphi(n_d)}^{(d)}, b_{\varphi(1)}, \dots, b_{\varphi(n_d)})$ where σ is a permutation on $\{1, 2, \dots, n_t\}$ and φ is a permutation on $\{1, 2, \dots, n_d\}$. This canonically induces an equivalence relation on the almost feasible solutions for a given instance I such that an almost feasible solution is feasible if and only if all its equivalent solutions are feasible, too.

In Definition 4.4.19, we will define the multidigraph of a solution; in order to do this we need Definitions 4.4.17 and 4.4.18.

Definition 4.4.17 (Disjoint union of sets)**.** *Let I be any index set. For each $i \in I$, let A_i be a set. Then the disjoint union of the A_i over I is defined as*

$$\bigsqcup_{i \in I} A_i := \bigcup_{i \in I} \{(x, i) \mid x \in A_i\}.$$

For the disjoint union of two sets A and B we simply write $A \sqcup B$.

Definition 4.4.18 (Union of multidigraphs)**.** *Let $G_1 = (V_1, E_1, \psi_1)$ and $G_2 = (V_2, E_2, \psi_2)$ be two multidigraphs. We define their union $G_1 \cup G_2$ to be the triple $(V_1 \cup V_2, E_1 \sqcup E_2, \psi)$ where*

$$\begin{aligned} \psi : E_1 \sqcup E_2 &\to (V_1 \cup V_2)^2 \\ (e, i) &\mapsto \psi_i(e). \end{aligned}$$

To emphasise that the disjoint union of the edges is considered, we often write $G_1 \sqcup G_2$ instead of $G_1 \cup G_2$.

Note that in Definition 4.4.19, the expressions "corresponds to a flying drone" etc. could easily be formalised, i.e. there are no hidden subtleties; however, for better readability at this point we use the informal terms.

Definition 4.4.19 (Multidigraph of a solution)**.** *Let $n \in \mathbb{N}$ and let I be an instance of size n. Let S be a solution for I. Let $M = (V, E, \psi)$ denote the multidigraph that is defined as the union of the individual tours of S, i.e. $M := T_1^{(t)} \sqcup \ldots \sqcup T_{n_t}^{(t)} \sqcup T_1^{(d)} \sqcup \ldots \sqcup T_{n_d}^{(d)}$. Define $c : E \to \{0, 1, 2\}$ by*

$$c(e) = \begin{cases} 0 & \text{if } e \text{ corresponds to a flying drone} \\ 1 & \text{if } e \text{ corresponds to a drone carried by a truck.} \\ 2 & \text{if } e \text{ corresponds to a truck} \end{cases}$$

In slight abuse of notation (as only M is a multidigraph), we call the tuple (M, c) the multidigraph of S.

Lemma 4.4.20 states that the multidigraph of an almost feasible solution captures all important information of that solution.

Lemma 4.4.20. *Let $n \in \mathbb{N}$ and let I be an instance of size n. Let S be a solution for I that is almost feasible. Given the multidigraph (M, c) of S, one can reconstruct a solution $\tilde{S}$ for I which is equivalent to S.*

Proof. Let $1 \leq i < j \leq n_t$. It is $V_i^{(t)} \cap V_j^{(t)} = \{v_{n+1}\}$. Hence $T_1^{(t)}, \ldots, T_{n_t}^{(t)}$ clearly can be reconstructed (up to a permutation of these cycles, but this suffices to reconstruct the solution up to equivalence). Now we provide an algorithm that reconstructs cycles $\tilde{T}_1^{(d)}, \ldots, \tilde{T}_{n_d}^{(d)}$ and respective binary functions $\tilde{b}_1, \ldots, \tilde{b}_{n_d}$ such that $\tilde{S} = (T_1^{(t)}, \ldots, T_{n_t}^{(t)}, \tilde{T}_1^{(d)}, \ldots, \tilde{T}_{n_d}^{(d)}, \tilde{b}_1, \ldots, \tilde{b}_{n_d})$ is a solution that is equivalent to S. In the beginning, we call all edges of M that are not contained in any of the cycles in $\mathcal{T}^{(t)}$ available. While the algorithm proceeds, edges will become unavailable and the algorithm terminates when all edges are unavailable. It is clear that in the beginning n_d available edges leave node v_{n+1} in the multidigraph. Let us denote these edges by $e_1^*, \ldots, e_{n_d}^*$. For each $1 \leq i \leq n_d$, do the following:

1. Set $e = e_i^*$. Initialise $\tilde{T}_i^{(d)}$ as $\tilde{T}_i^{(d)} = (e)$ and set $\tilde{b}_i(e) = c(e)$.[4] Mark e as unavailable. Set $s = c(e)$ and set $x = (\psi(e))[2]$.

2. As we assume that S is almost feasible, there is an $l \in \{1, \dots, n_d\}$ such that there are l available edges $e_1, \dots, e_l$ such that for each $1 \leq i \leq l$ it holds $(\psi(e_i))[1] = x$. Set

$$E_0 = \{e_i \mid 1 \leq i \leq l \text{ and } c(e_i) = 0\}$$

and

$$E_1 = \{e_i \mid 1 \leq i \leq l \text{ and } c(e_i) = 1\}.$$

If $s = 0$: If $E_1 \neq \emptyset$, choose $e \in E_1$ and set $\tilde{b}_i(e) = 1$, else choose $e \in E_0$ and set $\tilde{b}_i(e) = 0$.

If $s = 1$: If $E_0 \neq \emptyset$, choose $e \in E_0$ and set $\tilde{b}_i(e) = 0$, else choose $e \in E_1$ and set $\tilde{b}_i(e) = 1$.

Add e to $\tilde{T}_i^{(d)}$ (in the canonical way, e.g. if before we had (e_1, e_2, e_3) then now we have (e_1, e_2, e_3, e)). Set $x = (\psi(e))[2]$ and set $s = c(e)$.

If $x \neq v_{n+1}$ then go to 2. Else $\tilde{T}_i^{(d)}$ is complete: If $i < n_d$, increase i by 1 and go to 1; if $i = n_d$ we are already finished.

As S is almost feasible, this procedure yields a solution $\tilde{S}$ that is equivalent to S. □

Remark 4.4.21. *The intuition behind the algorithm given in the proof of Lemma 4.4.20 is the following. If a drone arrived flying at a node then it is either the only vehicle visiting that node and hence also flies away, or if it is not the only vehicle visiting that node, then it must leave carried by a truck. Similarly, if a node is visited by more than one vehicle and there are edges corresponding to flying drones leaving that node, then all drones corresponding to these edges must have arrived carried by a truck. Hence, if whenever possible we let drones that arrived flying leave carried by a truck and whenever possible we let drones that were carried to that node fly away, then we reconstruct the solution up to equivalence.*

Remark 4.4.22. *Let $n \in \mathbb{N}$ and let I be an instance of size n. Let S_1 and S_2 be almost feasible solutions for I. If S_1 and S_2 are equivalent, then they have the same multidigraph.*

Theorem 4.4.24 provides an alternative characterisation of the consistency (and thus of the feasibility) of a solution. It relates the consistency to the absence of cycles in the multidigraph of the solution that have a certain property. In particular, it enables us to characterise consistency without using an iterative, round-based process. In order to state Theorem 4.4.24, we need Definition 4.4.23.

[4] This is a slight abuse of notation: To be precise, we should say that if e is an edge in the multidigraph from node u to node w with $u, w \in \{v_1, \dots, v_{n+1}\}$, i.e. $(u, w) = \psi(e)$, then we initialise $\tilde{T}_i^{(d)}$ as $\tilde{T}_i^{(d)} = ((u, w))$ and set $\tilde{b}_i((u, w)) = c(e)$. However, to lighten the notation, in the following we ignore this subtlety, as the precise meaning is clear from the context.

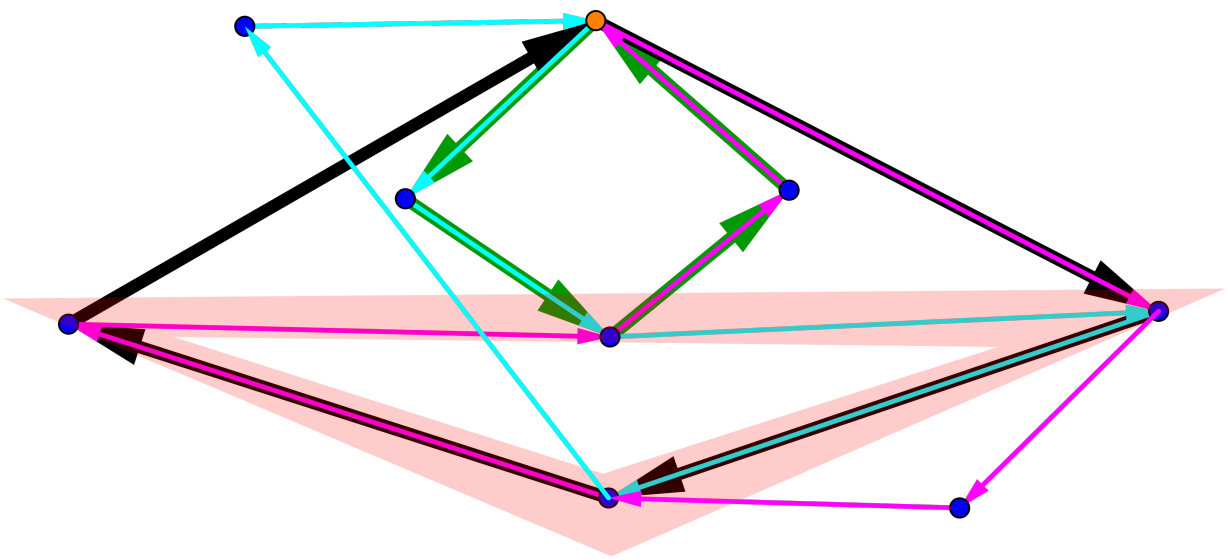


Figure 4.2: Necessity to allow flip-cycles. Trucks 1 and 2 are indicated in black and green respectively, drones 1 and 2 are indicated in cyan and pink respectively, the depot is indicated in orange. The flip cycle is indicated in light red.

Definition 4.4.23 (Flip circuit, flip cycle)**.** *Let (M, c), $M = (E, V, \psi)$, be a multidigraph of an almost feasible solution S for an instance I. Let $\tilde{S} = (T_1^{(t)}, \dots, T_{n_t}^{(t)}, T_1^{(d)}, \dots, T_{n_d}^{(d)}, b_1, \dots, b_{n_d})$ be a solution that is equivalent to S. For $1 \leq k \leq n_d$, we write $T_k^{(d)} = (V_k^{(d)}, E_k^{(d)})$. Let $C \subseteq M$ be a cycle. We say that $C = (V_C, E_C, \psi_C)$ is a flip cycle w.r.t. $\tilde{S}$, if there are edges $e_1, e_2 \in E_C$ such that e_2 follows e_1 and such that there are $i, j \in \{1, \dots, n_d\}$ with $i \neq j$ such that $e_1 \in E_i^{(d)}$ and $e_2 \in E_j^{(d)}$. If C is only a circuit instead of a cycle, then under the described conditions it is called a flip circuit w.r.t. $\tilde{S}$.*

Theorem 4.4.24 (Alternative characterisation of consistency)**.** *Let I be an instance and S be a solution for I that is almost feasible. Let (M, c) be the multidigraph for S and let $\tilde{S}$ be a solution that is reconstructed from (M, c) according to Lemma 4.4.20. Then $\tilde{S}$ satisfies the consistency constraint and thus is feasible if and only if each cycle $C \subseteq M$ contains v_{n+1} or is a flip cycle w.r.t. $\tilde{S}$. Note that S is feasible if and only if $\tilde{S}$ is feasible.*

Figure 4.2 illustrates the necessity to allow flip cycles in Theorem 4.4.24.

Proof of Theorem 4.4.24. For the one direction we will assume that $\tilde{S} = (T_1^{(t)}, \dots, T_{n_t}^{(t)}, T_1^{(d)}, \dots, T_{n_d}^{(d)}, b_1, \dots, b_{n_d})$ is not consistent and prove that there is a cycle C in the multidigraph (M, c) of $\tilde{S}$, $M = (V, E, \psi)$, that is neither a flip cycle w.r.t. $\tilde{S}$ nor contains the depot. Let $\tilde{\mathcal{T}} = \{T_1^{(t)}, \dots, T_{n_t}^{(t)}, T_1^{(d)}, \dots, T_{n_d}^{(d)}\}$. By assumption, we know that the round based process defined in Definition 4.4.14 will not mark every edge of every cycle $T \in \tilde{\mathcal{T}}$; in particular, when the marking process has stopped, there is a cycle $T_1 \in \tilde{\mathcal{T}}$, $T_1 = (V_1, E_1)$, such that T_1 has the form $T_1 = (v_{n+1}, \dots, w_1, u_1, \dots)$ and exactly the edges on the path from v_{n+1} to w_1 are marked. This implies that there is another tour $T_2 \in \tilde{\mathcal{T}}$, $T_2 = (V_2, E_2)$, $T_2 \neq T_1$, such that there is $w_2 \in V_2$ such that $T_2 = (v_{n+1}, \dots, w_2, \dots, w_1, u_1, \dots)$ and exactly the edges

on the path from v_{n+1} to w_2 are marked. By iterating we can infer that there are tours $T_i, T_{i+1}, \dots, T_j \in \tilde{\mathcal{T}}$ such that

$$\begin{aligned}
T_i &= (v_{n+1}, \dots, w_i, u_i, \dots), \\
T_{i+1} &= (v_{n+1}, \dots, \underbrace{w_{i+1}, u_{i+1}, \dots,}_{S_{j-i+1}} w_i, u_i, \dots), \\
T_{i+2} &= (v_{n+1}, \dots, \underbrace{w_{i+2}, u_{i+2}, \dots, w_{i+1}, u_{i+1}}_{S_{j-i}}, \dots), \\
&\vdots \\
T_j &= (v_{n+1}, \dots, \underbrace{w_j, u_j, \dots, w_{j-1}, u_{j-1}}_{S_2}, \dots) \quad \text{and} \\
T_i &= (v_{n+1}, \dots, \underbrace{w_i, u_i, \dots, w_j, u_j}_{S_1}, \dots)
\end{aligned}$$

where for $i \leq k \leq j$ in T_k exactly the edges on the path from v_{n+1} to w_k are marked. Now we construct a circuit $C_1 \subseteq M$; to do this we concatenate $S_1, S_2, \dots, S_{j-i+1}$ in this order such that each edge corresponds to the cycle from which the corresponding S_i was taken from, except for the overlapping edges: The overlapping edges are included only once and in these cases the respective edges are chosen such that they correspond to truck cycles. This is always possible (even if, e.g., all cycles $T_i, \dots, T_j$ are drone cycles) because whenever a vehicle waits for another vehicle, it travels the next edge on a truck or is a truck itself. By construction, C_1 neither contains the depot nor is a flip circuit w.r.t. $\tilde{S}$. Next we modify $C_1 = (V_1, E_1, \psi_1)$ as follows, thereby obtaining $C_2 = (V_2, E_2, \psi_2)$. For all $e \in E_1$, do the following. If there is a $q \in \{1, \dots, n_t\}$ such that for $T_q^{(t)} = (V_q^{(t)}, E_q^{(t)})$ there is an $\tilde{e} \in E_q^{(t)}$ with $\psi(\tilde{e}) = \psi(e)$, then replace e by $\tilde{e}$. This yields C_2; clearly C_2 neither contains the depot nor is a flip circuit w.r.t. $\tilde{S}$.

Next we modify the circuit C_2 to obtain a cycle C that has the desired properties. Assume that there is a node $y \in V_2$ that occurs $k \geq 2$ times, i.e. $C_2 = (\dots, x_1, y, z_1, \dots, x_2, y, z_2, \dots, x_k, y, z_k, \dots)$.[5] First, note that for $1 \leq l \leq k$ it holds $c(x_l, y) + c(y, z_l) \geq 1$, because otherwise (i.e. if $c(x_l, y) + c(y, z_l) = 0$), since C_2 is no flip circuit w.r.t. $\tilde{S}$, (x_l, y) and (y, z_l) would correspond to the same flying drone and then y would be part of exactly one cycle from $\{T_1^{(d)}, \dots, T_{n_d}^{(d)}\}$. Hence it suffices to consider the following two cases. The first case is that $c(x_1, y) = 2$. In this case replace $(\dots x_1, y, z_1, \dots x_2, y, z_2, \dots, x_k, y, z_k, \dots)$ by $(\dots x_1, y, z_k, \dots)$. The second case is that $c(y, z_1) = 2$; in this case, replace $(\dots x_1, y, z_1, \dots x_2, y, z_2, \dots, x_k, y, z_k, \dots)$ by $(y, z_1, \dots x_2)$. Note that cases 1 and 2 already cover all cases as, by replacing C_1 with C_2, we have replaced all edges on which c takes the value 1 by edges on which c takes the value 2. The changes in both cases clearly preserve the properties that the circuit is neither a flip circuit w.r.t. $\tilde{S}$ nor contains the depot. Iterating this yields a cycle with the desired properties.

[5] This is a slight abuse of notation, as C_2 is a cycle in a multidigraph and not just in a digraph. Hence not all information is captured by a sequence of nodes because between two nodes can be several edges. To lighten the notation, we nevertheless describe the cycle as a sequence of nodes since from the context it is clear which edges are meant. Expressions like, e.g., $c(x_1, y)$ are interpreted in the same sense.

We continue with the other direction, i.e. we assume that $\tilde{S}$ is almost feasible and further assume that in the multidigraph of $\tilde{S}$ there is a cycle C that is neither a flip cycle w.r.t. $\tilde{S}$ nor contains the depot. We have to show that $\tilde{S}$ is not consistent. Let $k \in \mathbb{N}$ be the size of C. Let us write $C = (w_1, \dots, w_k)$. To prove the inconsistency of $\tilde{S}$ we show that in the marking process defined in Definition 4.4.14, each edge in C cannot be marked before its predecessor edge in C is marked. As in the beginning no edges are marked, this implies that never all edges of C will be marked; indeed it implies that never any edge of C will be marked. Note that after the first round of the marking process no edges of C are marked (as C does not contain the depot v_{n+1}). Consider two consecutive edges $e_1 = (x, y)$ and $e_2 = (y, z)$ of C. Let $T_1, T_2 \in \{T_1^{(t)}, \dots, T_{n_t}^{(t)}, T_1^{(d)}, \dots, T_{n_d}^{(d)}\}$ be the cycles corresponding to e_1 and e_2. Then, since C is not a flip cycle, we are in at least one of the following three cases.

- It is $T_1 = T_2$.
- The cycles T_1 and T_2 are interconnected on e_1 (according to Definition 4.4.12).
- The cycles T_1 and T_2 are interconnected on e_2.

In either case, e_2 cannot be marked before e_1 is marked. □

To define an objective function, Definitions 4.4.25 and 4.4.26 will be useful. Definition 4.4.25 formalises the arriving, the waiting and the departure times. Intuitively, as the name suggests, the arriving time of a vehicle at a package destination is the time needed until that vehicle arrives at the respective package destination; analogously, the respective departure time is the time until the vehicle leaves that package destination; the respective waiting time is defined as the difference of the departure and the arriving time.

Definition 4.4.25 (Arriving / waiting / departure time)**.** *Let $n \in \mathbb{N}$ and let I be an instance of size n. Let S be a solution for I. For each $T \in \mathcal{T}$, $T = (V, E)$, for each $v \in V$, we define the arrival time $ar(v, T) \in \mathbb{R}_0^+$, the departure time $dp(v, T) \in \mathbb{R}_0^+$ and the waiting time $w(v, T) := dp(v, T) - ar(v, T) \in \mathbb{R}_0^+$. Write $T = (v_{n+1}, u_1, \dots, u_{k-1}, u_k)$. We set $dp(v_{n+1}, T) = 0$. If $T \in \mathcal{T}^{(t)}$ or if there is a $1 \leq k \leq n_d$ such that $T = T_k^{(d)}$ and such that $b_k(v_{n+1}, u_1) = 1$, then set $ar(u_1, T) = D_t[v_{n+1}, u_1]$. Else set $ar(u_1, T) = D_d[v_{n+1}, u_1]$. We do this for all T. Now we inductively define the remaining arrival and departure times simultaneously for all $T \in \mathcal{T}$. To do this, recall the round based process from Definition 4.4.14. Consider the marking after round $r \geq 1$. We assume that for any $T \in \mathcal{T}$, $T = (V, E)$, for any $u \in V$ with $u \neq v_{n+1}$ and such that u is adjacent to a marked edge $e \in E$, we already know $ar(u, T)$. Let $e = (x, y)$ be an edge of a cycle $T \in \mathcal{T}$ such that e will become marked in round $r + 1$. By induction we already know $ar(x, T)$. We define $dp(x, T)$ and $ar(y, T)$ as follows. Recall the definition of interconnected cycles from 4.4.12. Like in Definition 4.4.14 we distinguish between the following two cases.*

- *Case 1, $T \notin IC(e, S)$: Set $dp(x, T) = ar(x, T)$ and set $ar(y, T) = dp(x, T) + D_d[x, y]$.*
- *Case 2, $T \in IC(e, S)$: Set $dp(x, T) = \max\{ar(x, \tilde{T}) \mid \tilde{T} \in IC(e, S)\}$ and $ar(y, T) = dp(x, T) + D_t[x, y]$.*

Afterwards we perform the next round of the marking step; thereby, inductively, we obtain all arrival, departure and waiting times.

Intuitively, case 1 in Definition 4.4.25 refers to the case that a drone flies along the respective edge. In contrast, in case 2, a truck, possibly carrying several drones, drives along the respective edge; it can only start when the last vehicle involved has arrived.

Definition 4.4.26 specifies the (average) delivery time; intuitively, as the name suggests, the delivery time of a package is the time until that package is delivered; note that this is not necessarily the time until the first vehicle arrives at the respective destination, as drones can carry at most one package at once. The average delivery time is defined as the average of the delivery times of all packages.

Definition 4.4.26 ((Average) delivery time)**.** *Let $n \in \mathbb{N}$ and let I be an instance of size n. Let S be a feasible solution for I. For any $1 \leq i \leq n$ we will define the delivery time of v_i. In order to do this, we consider a certain subset $\mathcal{T}(i) \subseteq \mathcal{T}$: A cycle $T \in \mathcal{T}, T = (V, E)$, is contained in $\mathcal{T}(i)$ if and only if $v_i \in V$ and additionally at least one of the following three conditions is fulfilled.*

(1) There is $1 \leq j \leq n_t$ such that $T = T_j^{(t)}$.

(2) v_i follows v_{n+1} in T.

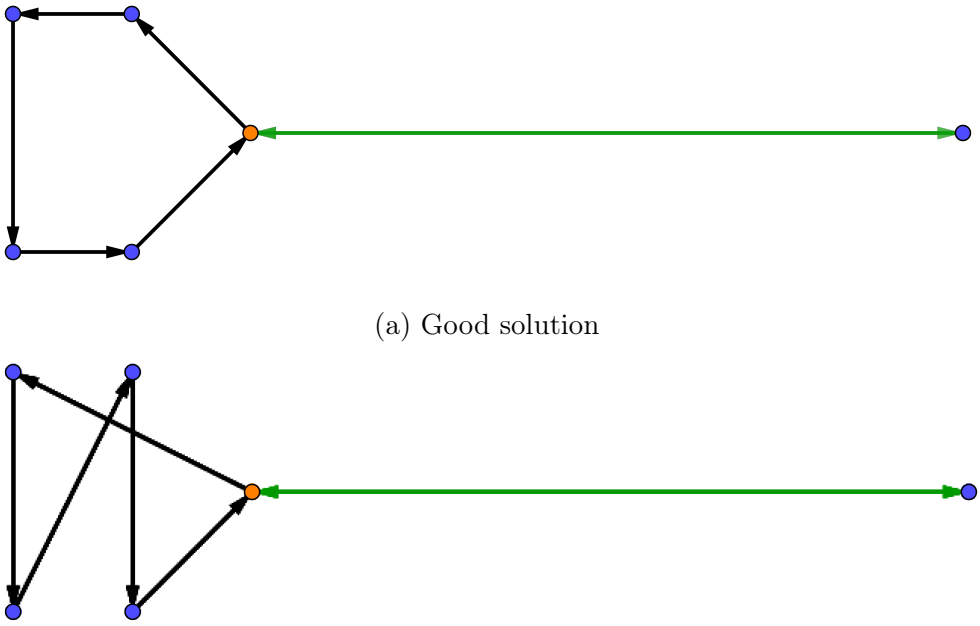
(3) There is $1 \leq k \leq n_d$ such that $T = T_k^{(d)}$. Let y be the first and x be the second predecessor node of v_i in T, i.e. $T = (\dots, x, y, v_i, \dots)$. Then $b_k(x, y) + b_k(y, v_i) \geq 1$.

As S is feasible, $\mathcal{T}(i)$ is non-empty. The delivery time of v_i is defined as $del(v_i) := \min\{ar(v_i, \tilde{T}) \mid \tilde{T} \in \mathcal{T}(i)\}$. The average delivery time of S is defined as

$$av(S) := \frac{1}{n} \sum_{i=1}^{n} del(v_i).$$

Intuitively, conditions 1, 2 and 3 cover all cases except for the case that the cycle T corresponds to a drone that already has delivered a package on its flight and hence, when arriving at v_i, cannot deliver the package at v_i; this reflects that drones can carry at most one package at once.

Different choices for an objective function than the average delivery time are possible. A canonical choice is the completion time, i.e. the time until all packages are delivered and all vehicles have returned to the depot. However, choosing the average delivery time, which is the average time that a customer has to wait for a delivery, as the objective function has some advantages: The most important advantage is that it also mirrors improvements that have no influence on the completion time because the arrival time of the last vehicle is not affected; compare Figure 4.3. There an example with two trucks and without drones is provided; both solutions have the same completion time (we assume Euclidean distances in this example) but solution (a) is clearly preferable. Another advantage is that the average delivery time is sensitive to whether many or few customers get delivered late; i.e. while the completion time may be the same, solutions where most customers are served soon have a better average delivery time, cf. Figure 4.4 where an example with one truck and without drones is provided.



(a) Good solution

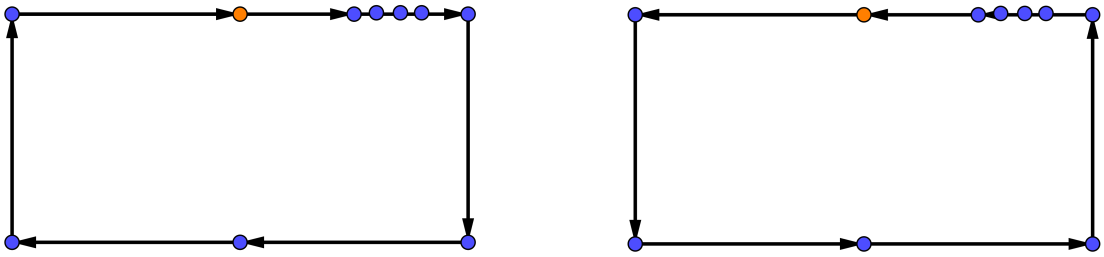
(b) Poor solution

Figure 4.3: Solutions with the same completion time, one truck in green, one truck in black, depot in orange.

Again both solutions have the same completion time (assuming symmetric distances), but solution (a) is preferable. This motivates to use the average delivery time. Thus, from now on, with "objective function" we refer to the average delivery time.

4.5 Local Search Algorithms

Local search algorithms are powerful approaches to deal with optimisation problems. In this section, we will provide a brief general introduction to local search algorithms and sketch some important examples. A core idea of local search algorithms is the following. For the



(a) Good solution

(b) Poor solution

Figure 4.4: Solution and reversed solution, same completion time, depot in orange.

solutions in the search space, define a suitable neighbourhood, i.e. for a given solution we want to be able to create a list of its neighbour solutions. Defining well-suited neighbourhoods is crucial when defining a local search algorithm. Next, for a suitable neighbourhood definition, we start at some point in the search space, i.e. at an initial solution, and look at its neighboured solutions. Then, we continue with one of its neighbours. This neighbour is chosen in a way that tends to improve the objective function, i.e. it is more likely to go to a good neighbour than to go to a bad neighbour. However, this does not mean that always the best neighbour is chosen; on the contrary, it is important to also allow a worsening of the objective function as otherwise one gets trapped in local optima. To illustrate the idea, suppose that we want to maximise a two-dimensional, real-valued function. The graph of the function can be imagined as a mountain region with several peaks but many of these peaks are only local maxima. Now a local search algorithm can be thought of as a strategy for a hiker that starts somewhere in this mountain region and wants to climb on the highest peak. However, it is foggy, hence the hiker can only see up to ten meters around him. Now he tends to go in a direction where the path goes upwards, however, if he has reached a peak, i.e. there is no more path upwards, he might go several hundred meters downwards and then try a different direction. This visualises that while tending to go upwards is often useful, it is also necessary to accept worse solutions in the short term in exchange for the possibility to escape local optima that are not globally optimal. For a precise treatment of the field of stochastic local search, we refer to the book by Hoos and Stützle ([65]). Looking at the basic idea of local search algorithms, it is apparent that a crucial part of defining a local search algorithm is to define the neighbourhoods for the solutions in the search space. In Section 4.6, we will define such neighbourhoods in the context of the VRD model, but first we will present some fundamental local search algorithms to illustrate the core idea described above and to demonstrate the diversity in the field of local searches. A summary of the following subsections can be found in Table 4.1.

4.5.1 Steepest Descent — Random Descent

Steepest descent is a very basic local search algorithm. From a current solution, the algorithm investigates the complete neighbourhood and moves on to the best neighboured solution if it is better than the current solution. If no neighboured solution is better than the current solution, then it terminates. Steepest descent has the major drawback that it has no mechanism to avoid getting stuck in local optima. This can be mitigated by rerunning the algorithm with different initial solutions; however, this often will not suffice, especially when there are many local optima.

Random descent is very similar, but instead of searching for the best solution in the whole neighbourhood, in each step a random neighbour is chosen and, if it is better, it replaces the current solution, otherwise a new random neighbour is considered. One has to define a suitable stopping criterion, e.g. that there are no improvements for a certain number of steps. Just as steepest descent, random descent might get stuck in local optima. There is a canonical combination of random descent and steepest descent: Consider k random neighbours and take the best if it is better than the current solution; iterate this.

Local Search	Key Idea
Steepest Descent	For any solution, define its neighbourhood. Take an initial solution. Look at its entire neighbourhood. If a solution in this neighbourhood is better, replace the current solution with the best solution of the neighbourhood, otherwise terminate. Iterate this.
Random Descent	Basically like steepest descent, but instead of looking at all neighbours, only look at one randomly sampled neighbour and accept it if it is better. This process is iterated. Random descent and steepest descent can be combined canonically.
Tabu Search	Tabu search can be considered as a generalisation of steepest descent. However, it has a mechanism that makes it possible to escape from local optima, i.e. a tabu list of, e.g., solutions or moves depending on recently visited solutions or recently applied moves (short-term memory). Solutions (or moves) on this list are tabu, i.e. they may currently not be visited (or applied). It also has mechanisms for diversification and intensification.
Metropolis Search and Simulated Annealing	For Metropolis search fix a temperature. Choose a solution. Look at a neighbour. Accept it if it is better. Accept it also with a certain probability if it is worse, depending on how much worse it is and depending on the temperature (the higher the more probable is acceptance). For simulated annealing do the same but decrease the temperature over time.
Parallel Tempering	Look at several copies of a system where each system has a different temperature and each system has assigned an initial solution. Perform Metropolis search on each copy. After some time, two neighboured systems (with respect to their temperatures) switch their solutions such that already good solutions can move to replicas with low temperatures where they can converge and poor solutions can move to replicas with higher temperatures where they can be substantially changed (and thereby ideally improved) before they can cool down again.

Table 4.1: Summary of several fundamental local search algorithms

4.5.2 Tabu Search

Tabu search can be thought of as an attempt of taking the positive aspects of steepest descent while avoiding its main disadvantage, i.e. the disability to escape from local optima. Tabu search was introduced by Glover ([54, 55, 56]). Compare also, e.g., [58, 57, 71]. Here we will only provide a rough sketch of tabu search, omitting many details and variants.

Short-Term Memory for Escaping Local Extrema Here the current solution is not necessarily the best solution. In the end, the best solution found so far is returned. The basic principle of the simplest variant of tabu search is the same as for steepest descent except for one important difference: In order to be able to escape from local optima, one keeps track of a number (can be always the same fixed number, but can also vary over time) of solutions that one has visited last. It is forbidden (tabu) to go there, even if it is the best solution in the neighbourhood of the current solution. Now, in each step, one replaces the current solution by the best solution within its neighbourhood that is not tabu. This enables the search to escape from local optima. Note that tabu search in its general form is not restricted to forbidding certain solutions. More general, moves can be forbidden, which for example for the TSP problem could be to forbid swapping cities i and j. In these cases so called aspiration criteria can improve the results. If such an aspiration criterion is fulfilled (e.g. the respective move yields a solution that is better than the best solution found so far) it is accepted as the new current solution even if it is tabu. The tabu list in which the algorithm keeps track of the forbidden solutions or moves respectively is often referred to as the short-term memory.

Intermediate-Term Memory for Intensification The intermediate-term memory tries to push the current solution into good regions. Therefore, loosely speaking, it looks at good solutions that are collected so far and extracts some features of them. E.g. in good TSP solutions, only a small subset of the edges of the graph might be used. Thus, solutions using these edges might be rewarded and hence may even be preferred over solutions that have a better objective value. Therefore this procedure causes an intensification of the search. Compare again [55].

Long-Term Memory for Diversification The long-term memory has a complementary function compared to the intermediate-term memory. Its purpose is to diversify the solution by penalising features that occurred often. For example for edges that occurred in many TSP solutions found so far (not only in the good ones like above) there may be a penalty for solutions containing them. This diversifies the search, as new parts of the search space are visited.

4.5.3 The Metropolis Algorithm — Simulated Annealing

Metropolis search is based on the more general works by Metropolis et al. ([80, 79]). We have a fixed temperature and a solution. Then we look at a random neighbour of the solution and accept it if it is better or accept it with a certain probability if it is worse. This probability depends on how much worse it is (much worse solutions are unlikely to be accepted) and on how high the temperature is (higher temperatures correspond to higher probabilities).

Metropolis search can be considered as a simplification of simulated annealing, but there the temperature is decreasing over time; compare [69, 1, 103]. Or, as Ingo Wegener states in [108], the Metropolis algorithm is equivalent to simulated annealing without temperature changes.

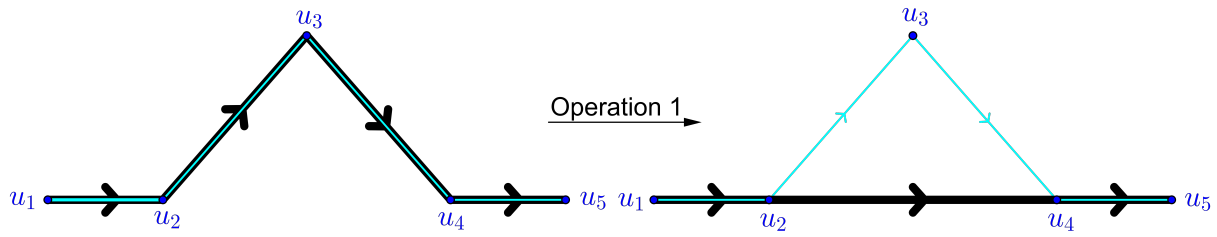
4.5.4 Parallel Tempering for Combinatorial Optimisation

This method was introduced by Swendsen and Wang [100] and is based on Metropolis search. See also [49, 66, 41]. However, the previous mentioned literature does cover parallel tempering with no special focus on combinatorial optimisation. For its application to combinatorial optimisation we refer to, e.g., [106] where the application of parallel tempering to the TSP is illustrated. We roughly summarise the principle of parallel tempering for combinatorial optimisation staying close to [106]: The core idea is to look at several copies of a system, where each sytem has assigned a temperature and an initial solution. In each system, the corresponding solution is modified according to the Metropolis search described in Subsection 4.5.3. After some iterations, two solutions of neighboured systems (with respect to their temperatures) are swapped; for details, we again refer to [106]. Thereby poor solutions can rise in temperature and hence be substantially changed (and ideally improved), thus overcoming local optima. In contrast, high-quality solutions can move to colder replicas to converge to good local (and ideally even global) optima.

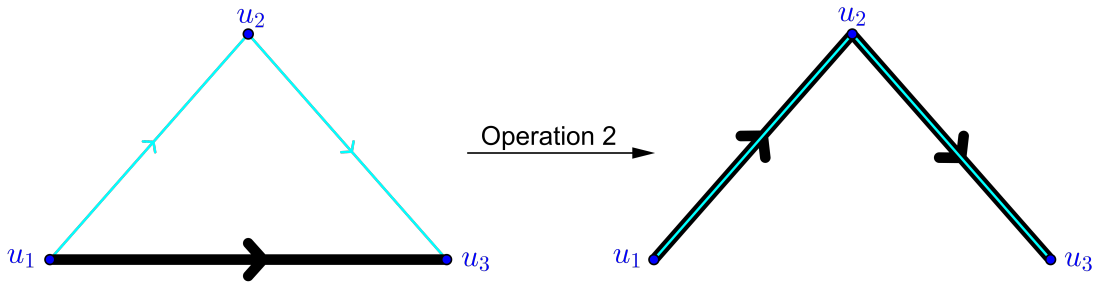
4.6 A Local Search Algorithm for Vehicle Routing with Drones

4.6.1 Definition of VRD-LOC

In this section we introduce the local search algorithm VRD-LOC (short for VRD Local Search) to solve the VRD problem; we will provide computational results in Subsection 4.6.2. VRD-LOC is fast, simple and can be used as a starting point to create more elaborate local search algorithms. In fact, in our article ([24]) we introduced an extended version of VRD-LOC which has seven additional operations to create a neighbour; these additional operations are not part of this thesis but will be part of Elisabeth Kraus' doctoral thesis; as we will see in Subsection 4.6.2, by such additional operations the solution quality can be improved significantly. We will discuss the scope and limitations of VRD-LOC in the paragraph "Discussion of VRD-LOC" below. Now we introduce VRD-LOC. For a given instance we start with an initial solution that ignores the drones and uses only the trucks, i.e. we start with a solution for the travelling salesmen problem with multiple salesmen (mTSP). This is a well-studied problem (cf. for example [10]) that is NP-hard. We treat the respective part of the algorithm as a blackbox that solves the mTSP, i.e. developing an algorithm for the well-studied mTSP is not part of this work. Next we want to put the drones into use. In order to do this, we first define two operations that take a feasible solution and modify it to obtain another (possibly not feasible) solution; we will use these operations in the algorithm formulation. Operation 2 is essentially the inverse of Operation 1. Both operations are illustrated in Figure 4.5.



(a) Operation 1



(b) Operation 2

Figure 4.5: Operations 1 and 2, the truck is indicated in black and the drone in cyan.

Operation 1 *Let $n \in \mathbb{N}$ and let I be an instance of size n. Let S be a feasible solution for I. Let $T_0 \in \mathcal{T}^{(t)}$ and $T_1 \in \mathcal{T}^{(d)}$ and let $b_{(1)}$ denote the respective binary function. Assume that there are nodes u_1, u_2, u_3, u_4, u_5 such that both, T_0 and T_1, have the form $(v_{n+1} \dots u_1, u_2, u_3, u_4, u_5 \dots)$ and such that $b_{(1)}(u_1, u_2) = b_{(1)}(u_2, u_3) = b_{(1)}(u_3, u_4) = b_{(1)}(u_4, u_5) = 1$. We allow $u_1 = v_{n+1}$, i.e. in this case the part "$v_{n+1} \dots u_1$" simply stands for "v_{n+1}"; analogously we allow $u_5 = v_{n+1}$. Delete u_3 in T_0 and set $b_{(1)}(u_2, u_3) = b_{(1)}(u_3, u_4) = 0$. For each $T \in \mathcal{T}^{(d)}$ with respective binary function b, $T \neq T_1$, that has the form $T = (v_{n+1}, \dots, u_2, u_3, u_4, \dots)$ delete u_3 from T and set $b(u_2, u_4) = 1$.*

Operation 1 can also be applied (canonically modified) if T_0 and T_1 have the form $(v_{n+1}, u_3, u_4, u_5, \dots)$ or $(v_{n+1}, \dots, u_1, u_2, u_3)$; the intuitive reason for this is that when starting from the depot, all drones are charged and when a drone returns to the depot it is not allowed to leave again anyway. Note that the constructed solution is not necessarily feasible, as it might be necessary that T_0 contains u_3 because, for example, the respective truck might have to collect another drone there.

Operation 2 *Let $n \in \mathbb{N}$ and let I be an instance of size n. Let S be a feasible solution for I. Let $T_0 \in \mathcal{T}^{(t)}$ and let $T_1 \in \mathcal{T}^{(d)}$ with respective binary function $b_{(1)}$. Assume that there are nodes u_1, u_2, u_3 such that $T_0 = (v_{n+1}, \dots, u_1, u_3, \dots)$, $T_1 = (v_{n+1}, \dots, u_1, u_2, u_3, \dots)$ and such that $b_{(1)}(u_1, u_2) = b_{(1)}(u_2, u_3) = 0$; analogously to Operation 1 we allow $u_1 = v_{n+1}$ or $u_3 = v_{n+1}$. In T_0, insert u_2 between u_1 and u_3. Set $b_{(1)}(u_1, u_2) = b_{(1)}(u_2, u_3) = 1$. For all $T \in \mathcal{T}^{(t)}$ with respective binary function b, $T \neq T_1$, such that $T = (v_{n+1}, \dots, u_1, u_3, \dots)$ with $b(u_1, u_3) = 1$ insert u_2 between u_1 and u_3 and set $b(u_1, u_2) = b(u_2, u_3) = 1$.*

Note that a solution that is obtained by applying Operation 2 is always feasible. Now we define VRD-LOC.

VRD-LOC Let $n \in \mathbb{N}$ and let I be an instance of size n. We obtain an initial solution

$S_{ini} = (T_1^{(t)}, \dots, T_{n_t}^{(t)}, T_1^{(d)}, \dots, T_{n_d}^{(d)}, b_1, \dots, b_{n_d})$ as follows: The cycles $T_1^{(t)}, \dots, T_{n_t}^{(t)}$ are given by an mTSP solution (we consider an mTSP solver as a blackbox, i.e. providing a new algorithm to solve the mTSP is not part of VRD-LOC). The drone cycles are obtained as follows. For $1 \leq k \leq n_d$ set $T_k^{(d)} := T^{(t)}_{(k-1 \mod n_t)+1}$. The functions b_j are all set to be constant functions with value 1. Now for $1 \leq k \leq n_d$ we modify $T_k^{(d)}$ one after the other. So let k be fixed. We start with the cycle $T_k^{(d)}$ from S_{ini} and modify it by applying parallel tempering (cf. Subsection 4.5.4). In order to do this, we have to define how a neighboured solution for the current solution is obtained: Let $T_k^{(d)} = (u_1, \dots, u_r)$ where $u_1 = v_{n+1}$ and set $u_{r+1} = v_{n+1}$. Consider $T^{(t)}_{(k-1 \mod n_t)+1} = (u'_1, \dots, u'_{r'})$ where $u'_1 = v_{n+1}$; set $u'_{r'+1} = v_{n+1}$. Choose $i' \in \{1, \dots, r'\}$ uniformly at random. If $i' = 1$, let $i^* = 1$. Else set

$$A := \{j \mid 1 \leq j \leq r \wedge \exists j' : (i' \leq j' \leq r' \wedge u_j = u_{j'} \wedge b(u_{j-1}, u_j) = 1)\}.$$

If $A = \emptyset$, then retry and sample i' again, else set $i^* := \min A$. Set $u = u_{i^*}$. Remark 4.6.1 provides an intuitive explanation for the choice of A and u.

Check whether Operation 1 is applicable where u has the role of u_2 in the definition of Operation 1. If this is the case, then check whether the resulting neighboured solution is feasible. Also check whether Operation 2 is applicable where u has the role of u_1 in the definition of Operation 2. Recall that if Operation 2 is applicable, the resulting neighboured solution is always feasible. If neither Operation 1 nor Operation 2 is applicable such that a feasible solution is obtained, then we restart by sampling i' again. Otherwise exactly one of the Operations 1 and 2 is applicable. Applying this operation then yields a feasible solution, i.e. we obtain a neighboured solution.

Now, as we have defined how to obtain an initial solution and how to construct a neighbour for a given solution, we can apply parallel tempering (recall that the objective function is the average delivery time). After some time (i.e. when a stopping criterion, e.g. a runtime bound, is fulfilled, cf. Subsection 4.6.2) we terminate parallel tempering for $T_k^{(d)}$ and continue with modifying $T_{k+1}^{(d)}$ and so on. After having modified all drone cycles using parallel tempering with the neighbour generation as described, we have obtained our final solution.

Remark 4.6.1. *While the definition of the auxiliary set A is somewhat technical, the respective part of VRD-LOC has a simple intuition. We consider the drone tour $T_k^{(d)}$ of drone k and the corresponding truck tour $T^{(t)}_{(k-1 \mod n_t)+1}$. We choose a node i' of the truck tour uniformly at random and, starting from this node, we continue along the truck tour until we arrive for the first time at a node u where drone k is charged. If there is no such node, we sample i' again and restart.*

Discussion of VRD-LOC Before we provide some empirical results, we want to discuss what VRD-LOC should accomplish and what its restrictions are. Its main purpose is to function as a proof of concept that parallel tempering with neighbourhood definitions like the one provided using Operations 1 and 2 is a suitable approach to solve the VRD problem. The two provided operations to create new neighbours only serve as a starter kit and have to be extended by additional operations in order to further improve the solution quality; in fact, in our article ([24]) we have considered seven additional operations and thereby obtained

significantly improved results (we will quantify this below when discussing the empirical results); the additional seven operations used in [24] are not part of this thesis but will be part of Elisabeth Kraus' doctoral thesis. Moreover, VRD-LOC might be further improved by adding operations that allow a drone to change trucks during its tour and operations that allow to switch packages between trucks during the local search; also compare the paragraph "Summary and Suggestions for Improvements" in Subsection 4.6.2.

A restriction of the provided algorithm is that it will not perform well on instances with more drones than trucks. This can be seen as follows: Loosely speaking, Operation 1 lets the truck skip one node in its tour which then is served by the drone under consideration and the drone meets again with the truck at the next node in the truck tour. Thus, after the first drone tour has been modified, at the majority of nodes that are still visited by the corresponding truck, the truck either collects or sends away the drone; also recall that each drone has to charge on one edge between two flights. Therefore, the truck cannot simply skip nodes on its tour which makes it difficult to appropriately put a second drone that is assigned to this truck into use.

To overcome this restriction, some possibilities are conceivable: By using additional operations like e.g. letting the truck skip one node in its tour (like in Operation 1) but meeting with the drone after the truck has delivered more than one package (cf. Operation 9 in [24]), the problem can be mitigated; the reason for this is that after a drone has been put into use, the truck tour will still contain nodes that can be simply skipped by the truck in order to put the next drone into use; this is done in our article ([24]) and there it is shown empirically that using the additional operations, also instances with more drones than trucks can be handled appropriately. While the added operations improved the solution quality significantly, for further improvements of the algorithm on instances with more drones than trucks it might be necessary to put the drones into use simultaneously instead of sequentially. Because of the discussed reasons, in this work we will focus on instances with at most one drone per truck.

Another important aspect besides the solution quality is the runtime; in the paragraph "Runtime" in Subsection 4.6.2 we will see empirically that VRD-LOC performs well in this respect.

A Greedy Algorithm In the following, we introduce a Greedy algorithm to which we will compare VRD-LOC. Intuitively, it works as follows. Each truck has assigned the same number of drones (up to at most one drone difference). We start with the same initial solution as for VRD-LOC. For each truck tour, starting from the depot, all l drones assigned to that truck are sent away to deliver the next l packages of the initial truck tour and then the truck collects all l drones at the destination of package $l+1$, travels one edge to the destination of package $l+2$ such that the drones are recharged and then, again, all l drones are sent away to deliver the next l packages and so on. If in the last step only $r < l$ packages are left to deliver, then the remaining packages are delivered by r drones which afterwards fly to the depot and the remaining $l - r$ drones travel to the depot directly, carried by the truck. Now we define the Greedy algorithm formally. Let $n \in \mathbb{N}$ and let I be an instance of size n. Let S_{ini} be the same initial solution as used by VRD-LOC. Next we modify the initial solution: For each $T \in \mathcal{T}^{(t)}$ we do the following. Write $T = (v_{n+1}, u_1, \ldots, u_k)$. Let $T_1, \ldots, T_l \in \mathcal{T}^{(d)}$ be those

drone cycles that, for $1 \leq i \leq l$, satisfy $T_i = T$. Let $b_{(1)}, \ldots, b_{(l)} \in \{b_1, \ldots, b_{n_d}\}$ denote the respective binary functions. To obtain the Greedy solution S_G, we do the following. Replace T by

$$(v_{n+1}, u_{(l+2)-1}, u_{l+2}, u_{2(l+2)-1}, u_{2(l+2)}, u_{3(l+2)-1}, \ldots, u_{\lfloor \frac{k}{l+2} \rfloor (l+2)-1}, u_{\lfloor \frac{k}{l+2} \rfloor (l+2)})$$

and for $1 \leq j \leq l$ replace T_j by

$$(v_{n+1}, u_j, u_{(l+2)-1}, u_{l+2}, u_{l+2+j}, u_{2(l+2)-1}, u_{2(l+2)}, u_{2(l+2)+j}, \ldots, u_{\lfloor \frac{k}{l+2} \rfloor (l+2)}, u_{\lfloor \frac{k}{l+2} \rfloor (l+2)+j})$$

where the last node, $u_{\lfloor k/(l+2) \rfloor (l+2)+j}$, is only present if $\lfloor k/(l+2) \rfloor (l+2) + j \leq k$; for all $1 \leq m \leq \lfloor k/(l+2) \rfloor$, we set $b_{(j)}(u_{m(l+2)-1}, u_{m(l+2)}) = 1$ and on all other edges e we set $b_{(j)}(e) = 0$. By applying the described procedure for each $T \in \mathcal{T}^{(t)}$, we obtain our final solution S_G of the Greedy algorithm. Clearly the runtime of Greedy is negligible.

4.6.2 Computational Results

We empirically evaluate VRD-LOC and compare the results to the results obtained by the introduced Greedy algorithm and to the mTSP results; as an outlook we also compare it to the results from our article ([24]) where an extended version of VRD-LOC with seven additional operations for the neighbour generation is introduced; these additional operations are not part of this thesis but will be part of Elisabeth Kraus' doctoral thesis. Before presenting the results, we again want to emphasise that VRD-LOC mainly serves as a proof of concept and starter kit that can be extended to more elaborate local search algorithms (for a detailed discussion about its capabilities and limitations see the paragraph "Discussion of VRD-LOC" in Subsection 4.6.1). In the following we will see that, while VRD-LOC already outperforms Greedy, by using the generalised version with seven additional moves, the algorithm can be further improved considerably. This underlines that VRD-LOC is a suitable starting point but should be extended by additional operations to further improve the solution quality.

The VRD problem formalised in this thesis was implemented in Java by Elisabeth Kraus; in particular, her implementation provides solution and instance classes and functions that check whether a given solution is feasible and that compute the average delivery time of a given solution. The implementation by Elisabeth Kraus is based on the Java metaheuristic search framework James ([26]) developed by De Beukelaer which allows to implement local search algorithms by specifying how to create an initial solution, by providing the objective function and by defining how a neighboured solution for a given solution can be constructed. Her implementation also provides an mTSP solver that is an adapted version of a local search algorithm to solve the TSP which is provided as one of the examples of James ([26]). The implementation of VRD-LOC is based on the model implementation by Elisabeth Kraus.[6]

[6]The Java implementation of both, VRD-LOC and the model (i.e. solution and instance classes etc., cf. above), can be accessed under the link `https://github.com/RabbitCodes/VRDLight`. There, also an implementation of a generalised version of VRD-LOC with seven additional operations for the neighbour generation is available; the implementation of these additional moves is made by Elisabeth Kraus and the generalised algorithm with these additional moves will be part of her doctoral thesis (cf. also the paragraph "Discussion of VRD-LOC" in Subsection 4.6.1). The model implementation was also made by Elisabeth Kraus; in particular, due to its modular structure, in order to implement VRD-LOC in Java, I only needed to implement Operations 1 and 2.

varying	fixed
# packages (25 / 50 / 75 / 100 / 125 / 150 / 200 / 250 / 300)	2 trucks, 2 drones
# trucks = # drones (1 / 2 / 3 / 4 / 5)	200 packages

Table 4.2: Test settings for VRD-LOC

In the following, we will present computational results to evaluate VRD-LOC empirically. To obtain test instances we sampled the package destinations uniformly at random on a 401×401 integer grid (x and y coordinates range from -200 to 200) excluding $(0, 0)$ which is the depot. The distances for the trucks are according to the Manhattan metric while the distances for the drones are according to the Euclidean metric. This is motivated by the observation that while trucks are restricted to the street network, drones may not be affected by such restrictions. In order to solve the mTSP to obtain the initial solution, we used the mTSP solver provided by Elisabeth Kraus' implementation. We ran computations[7] according to the settings described in Table 4.2.

Besides investigating the solution quality after a fixed period of time, we also consider how fast the algorithm converges, i.e. we consider the objective value after different periods of time. In the following, if not explicitly stated differently, with the objective value of VRD-LOC we refer to the objective value after one minute per drone tour (this does not include the time needed to solve the mTSP which is ten minutes per instance); we will not further consider the time needed to solve the mTSP to obtain an initial solution, as the mTSP is a well-studied problem (cf. for example the survey article by Bektas [10]) and we want to focus on the other aspects of VRD-LOC. Therefore we consider the mTSP solver as a blackbox that provides us with an mTSP solution. Note that also the Greedy algorithm is based on that mTSP solution. We will see that even within much shorter runtimes than one minute per drone tour, solutions of almost the same quality are obtained.

Before we consider the solution quality and the runtime in detail, we provide the internal parameters that we used for the local searches; as described above, the implementation of VRD-LOC is based on the Java implementation by Elisabeth Kraus which itself is based on the Java metaheuristic search framework James ([26]). Using these implementations, we applied parallel tempering as part of VRD-LOC (cf. Subsection 4.6.1). In order to obtain initial solutions, we used the mTSP solver from Elisabeth Kraus' Java implementation that is based on a TSP solver provided as an example in James using parallel tempering. We used a runtime of ten minutes for the mTSP solution for each instance that we considered; for the reasons mentioned above, in all further runtime considerations we will not include these ten minutes in the runtime.

We set the internal James settings to apply parallel tempering as part of the mTSP solver mentioned above as follows: We used three replicas, the temperature of the coldest replica is $t_{\min} = 0.0001$, the temperature of the warmest replica is $t_{\max} = 1.4$ and the temperature of the third replica is $(t_{\min} + t_{\max})/2$. This specifies the settings to obtain the initial mTSP solution.

[7]We ran the computations on a Microsoft Windows 10 Home machine with an Intel Core i7-7700 CPU (3.6 GHz, 4 cores, 8 threads), 16 GB RAM and an AMD Radeon R7 450 graphic card; the Java version we used is Java 8 Update 151 (Oracle); we used Eclipse (Oxygen, Release 4.7.1a) as our IDE.

Next we specify the parameters used in the local searches in the main part of VRD-LOC. Per each drone tour a local search (namely parallel tempering) was performed. In the following, we provide the internal James parameters that we used for these local searches. The runtime for each of these local searches was set to one minute (i.e. one minute per drone tour); however, we considered intermediate results at several time steps and it turns out that drastically shortened runtimes suffice to obtain solutions of almost the same quality, see the paragraph "Runtime" below for details. Again we used three replicas, this time with minimum temperature $\tilde{t}_{\min} = 0.001$, maximum temperature $\tilde{t}_{\max} = 1.0$ and where the temperature of the third replica is $(\tilde{t}_{\min} + \tilde{t}_{\max})/2$.

We always used 30 instances per setting that are sampled independently as described above. In all settings, for VRD-LOC the same mTSP solution was used as for Greedy, i.e. both algorithms started with the same initial solution.

Solution Quality We start with discussing the setting described in the first row of Table 4.2; i.e. we consider instances with two drones, two trucks and a varying number of packages. Per number of packages, we consider 30 instances and take the average of the objective values. The results can be found in Table 4.3. Now we summarise the main characteristics of the results. In the following, with "objective value" we refer to the average objective value over the respective 30 instances. For all numbers of packages the objective value of the mTSP solution without drones was between 10.0% and 11.0% bigger than the objective value of the VRD-LOC solution and the objective value of the Greedy solution was between 3.4% and 4.4 % bigger than the objective value of the VRD-LOC solution.

While the solution quality of VRD-LOC is not eminently better than the solution quality of Greedy, it is still noteworthy that despite the very simple and straight-forward neighbour generation, VRD-LOC outperforms Greedy. It also motivates to further extend the neighbour generation by additional operations which is done in [24] (the additional seven operations will be part of Elisabeth Kraus' doctoral thesis); the results in [24] show that considering the generalised version with the additional operations yields considerably better results: In the setting of the first row of Table 4.2, the objective values of the mTSP solutions were between 13.6% and 16.2% bigger than the objective values of the generalised version of VRD-LOC and the Greedy algorithm's objective values were between 7.2% and 9% bigger than the objective values of the generalised version of VRD-LOC. Note that there are some differences between the setting from [24] and the present setting (e.g. the mTSP was solved slightly differently and the runtime differed; also while the instances were sampled in the same way, in [24] different samples were used and the average was taken over 10 instances); nevertheless this illustrates that VRD-LOC is a suitable starter kit that can serve as a basis for improved algorithms that are obtained by adding further operations for the neighbour generation. It is noteworthy that the relative improvement of VRD-LOC compared to Greedy and compared to the mTSP solution is largely unaffected by the number of packages.

If we only use Operation 1 instead of both operations, we obtain results that have roughly the same (most of the times slightly better, rarely slightly worse) quality than Greedy. This is not surprising as almost always when Operation 1 is applicable, it will improve the previous solution; however, even if it improves the previous solution, as it cannot be "undone" as the inverse operation (Operation 2) is not available, it may prevent from finding even better

pckg.	ops.	mTSP	Grd.	50	250	1000	2000	3000	4000	5000	7500	10000	30000	60000	only 1 drone
25	1	494.1	460.8	456.2	456.2	456.2	456.2	456.2	456.2	456.2	456.2	456.2	456.2	456.2	474.0
	1&2			446.9	445.6	445.4	445.4	445.4	445.4	445.4	445.0	445.0	445.0	445.0	469.0
50	1	676.9	632.4	631.0	630.8	630.8	630.8	630.8	630.8	630.8	630.8	630.8	630.8	630.8	654.0
	1&2			614.5	612.6	612.0	611.9	611.7	611.7	611.7	611.7	611.6	611.6	611.6	643.9
75	1	803.5	758.0	753.8	753.8	753.8	753.8	753.8	753.8	753.8	753.8	753.8	753.8	753.8	778.4
	1&2			732.7	729.8	728.3	728.0	727.9	727.9	727.8	727.7	727.7	727.7	727.6	763.8
100	1	904.4	851.8	849.8	849.8	849.8	849.8	849.8	849.8	849.8	849.8	849.8	849.8	849.8	877.4
	1&2			826.5	822.9	820.5	819.5	819.3	818.9	818.8	818.8	818.8	818.6	818.1	860.6
125	1	1015.4	956.1	955.5	955.3	955.3	955.3	955.3	955.3	955.3	955.3	955.3	955.3	955.3	986.3
	1&2			933.2	927.8	924.3	922.4	922.1	921.7	921.5	921.2	920.8	920.2	920.1	968.8
150	1	1115.8	1057.4	1054.7	1054.6	1054.6	1054.6	1054.6	1054.6	1054.6	1054.6	1054.6	1054.6	1054.6	1084.3
	1&2			1028.9	1022.0	1018.3	1017.2	1016.6	1016.2	1016.1	1015.8	1015.5	1014.9	1014.4	1063.1
200	1	1288.2	1216.1	1217.3	1213.3	1213.3	1213.3	1213.3	1213.3	1213.3	1213.3	1213.3	1213.3	1213.3	1250.7
	1&2			1192.9	1181.5	1173.4	1171.0	1170.5	1169.8	1169.6	1168.1	1167.2	1165.9	1165.0	1226.2
250	1	1433.5	1352.0	1362.7	1354.1	1354.1	1354.1	1354.1	1354.1	1354.1	1354.1	1354.1	1354.1	1354.1	1394.6
	1&2			1335.1	1320.1	1311.3	1306.8	1305.4	1304.7	1304.2	1303.5	1302.5	1300.6	1299.7	1365.5
300	1	1572.1	1483.4	1502.5	1487.9	1487.6	1487.6	1487.6	1487.6	1487.6	1487.6	1487.6	1487.6	1487.6	1529.7
	1&2			1471.2	1452.6	1443.1	1439.3	1437.0	1435.9	1435.2	1433.4	1432.0	1429.0	1427.4	1498.9

Table 4.3: Two trucks, two drones and a varying number of packages (pckg.) are considered. Objective values for the setting without drones (mTSP), for Greedy (Grd.), for Operation 1 alone and for VRD-LOC (i.e. Operations (ops.) 1 and 2 together) are provided. The column "only 1 drone" refers to the solution with two trucks and one drone after one minute runtime to optimise that drone tour, starting with the mTSP solution. For Operation 1 alone and VRD-LOC the results are provided after various periods of time. In particular, the columns "50", "250" etc. refer to the runtime (in ms) that is used to put the second drone into use, starting with the "only 1 drone" solution; e.g. for 200 packages, considering the results for VRD-LOC, the objective value has improved from 1226.2 to 1192.9 within the first 50 ms during which the second drone has been put into use. This form of presentation is used because the drone tours are optimised one after the other. All results are rounded to one decimal. For each number of packages the average over 30 instances is taken.

solutions. In particular, most operations will be applied greedily and, as this cannot be undone, after a short period of time almost no operations can be applied any more and hence the solution quality does not improve any more. Hence it is not surprising that the solution quality is comparable to the solution quality of Greedy. Note that using only Operation 2 would have no effect as it could never be applied; in particular, the initial mTSP solution would never be modified.

We continue with discussing the setting described in the second row of Table 4.2; i.e. we investigate the impact of simultaneously increasing the number of trucks and drones used. For the second row of Table 4.2 we consider 30 instances in total and as before we consider the average of the objective values. The results can be found in Table 4.4.

We will observe that the objective value using two trucks and drones is less than half of the objective value using only one truck and drone; if our objective function was the completion time, this would be disconcerting: For simplicity consider the setting with two trucks and without drones; given a solution for this setting, by concatenating both tours (and shortening it canonically such that the truck is at the depot only at the beginning and at the end of its tour) one obtains a solution which has at most twice the completion time as the solution for two trucks. However, it is easy to come up with instances for which the optimal average delivery time with one truck is more than twice the optimal average delivery time with two trucks. Hence it is not disconcerting that the respective factor is bigger than

# trucks = # drones	ops.	mTSP	Grd.	60000
1	1	2655.4	2513.0	2521.6
	1&2			2435.7
2	1	1271.3	1202.0	1199.6
	1&2			1151.8
3	1	840.2	792.4	792.4
	1&2			761.6
4	1	648.5	613.1	610.8
	1&2			586.7
5	1	528.2	498.9	495.6
	1&2			479.1

Table 4.4: 200 packages and a varying number of trucks and drones are considered. Objective values for the setting without drones (mTSP), for Greedy (Grd.), for Operation 1 alone and for VRD-LOC (i.e. Operations (ops.) 1 and 2 together) are provided. The column "60000" refers to VRD-LOC and to the variant using only Operation 1; per drone tour one minute runtime is used, starting with the mTSP solution.[8] The results are rounded to one decimal, the average over 30 instances is taken.

two. To be specific, with only one truck the mTSP objective value is 2.09 times as large as with two trucks, with two trucks it is 1.51 times as large as with three trucks, with three trucks it is 1.30 times as large as with four trucks and with four trucks it is 1.23 times as large as with five trucks. For Greedy and VRD-LOC (always with the same number of drones as trucks) these factors are the same (up to ± 0.02).

If there is one truck and one drone we have the following results: The objective value of the mTSP solution without drones was 9.0% bigger than the objective value of the VRD-LOC solution and the Greedy solution was 3.2% bigger than the objective value of the VRD-LOC solution. For comparison (recall that the settings slightly differ, cf. above), we again consider the generalised version of VRD-LOC from [24] that uses seven additional operations for the neighbour generation (which will be part of Elisabeth Kraus' doctoral thesis). The objective value of the mTSP solution was 12.9% bigger than for the generalised version of VRD-LOC and the objective value of Greedy was 6.7% bigger than for the generalised version of VRD-LOC.

For the settings with two to five trucks and drones, the objective value of the mTSP solution without drones was between 10.2% and 10.5% bigger than the objective value of the VRD-LOC solution and the objective value of the Greedy solution was between 4.0% and 4.5% bigger than the objective value of the VRD-LOC solution. As above, for comparison we consider the generalised version of VRD-LOC from [24] that uses seven additional operations. The objective value of the mTSP solution was between 14.1% and 14.8% bigger than for the generalised version of VRD-LOC and the objective value of Greedy was between 7.1% and 7.7% bigger than for the generalised version of VRD-LOC. This underlines that VRD-LOC

[8] In particular, the total runtime for the respective entries in the rows further down is larger than the runtime for the respective entries at the top; however, as we have seen in Table 4.3 (and will see in more detail in Table 4.5 and in the paragraph "Runtime" in Subsection 4.6.2) VRD-LOC converges very fast, hence this has only marginal impact on the comparability of the different rows.

is a suitable starting point but additional operations for the neighbour generation should be added in order to improve the solution quality. Also note that in all settings of the second row of Table 4.2 the solution quality using Operation 1 only has been largely the same as the solution quality of Greedy, which according to the explanation given above is as expected.

Runtime Finally, we consider how fast VRD-LOC converges. Note that in general, e.g. in order to deduce suitable stopping times for a local search algorithm, it often does not suffice to only consider the average over several runs of the algorithm; in particular, even if we rerun a randomised algorithm on the same instance, runtime and objective value may differ. However, in the present setting the results for the individual instances (recall that for each setting we averaged over 30 instances) mostly yielded results rather close to the respective average results in Table 4.5 which contains the *relative* improvements after different time steps and is described in detail below; thus, here for approximative runtime considerations we use the average values. The issue that in many settings considering only the average runtime may not be sufficient was addressed in [64]; however, note that the setting discussed there in some aspects differs from the present setting; in particular, there Las Vegas algorithms were considered which are characterised by the property that whenever they return a solution, it is correct and their runtime is nondeterministic. In contrast, we can extract the currently best solution at any time step, i.e. VRD-LOC is a so-called anytime algorithm; also for many instances VRD-LOC may be unable (even theoretically) to find an optimal solution, because of its restricted neighbourhood generation. Nevertheless, the reasoning from [64] to some extent also applies here and if VRD-LOC should be applied in situations where runtime is highly critical, the behaviour of the respective random variables should be investigated in greater detail; then also different stopping criteria might be better suited, e.g. to stop if there is no significant improvement over a certain number of steps.

Now we start considering the runtime; in order to do this, recall that the algorithm puts the drones into use sequentially, i.e. parallel tempering is applied for each drone tour one after the other. Therefore we investigate the speed of convergence of VRD-LOC by considering the speed of convergence while parallel tempering is applied to one individual drone tour. Note that as we consider only settings with at most one drone per truck the optimisation of the first drone tour is largely independent of the optimisation of the second drone tour, hence it does not make a substantial difference whether we consider the improvement during the optimisation of the first or of the second drone tour. Thus, in order to investigate the speed of convergence, we consider the results in Table 4.5 which measure the progress of the local search while the second drone is put into use; the data can be derived from the absolute results from Table 4.3: We start with the solution where the first drone is already put into use; e.g. for 200 packages, considering VRD-LOC, this corresponds to the value "1226.2" from Table 4.3. Then we consider the objective value after the second drone has been put into use, i.e. after one additional minute runtime, in the example with 200 packages this corresponds to the value "1165.0". So in total there is an improvement of $1226.2 - 1165.0 = 61.2$ which corresponds to 100%. Now we investigate, after different time steps, which proportion of the improvement is still lacking; in the example with 200 packages, after 50 ms there is 45.6% lacking or in other words already 54.4% of the improvement has been achieved within the first 50 ms; note that the quantity 45.6% is obtained from the absolute values in Table 4.3

by the calculation $(1192.9 - 1165.0)/61.2 \approx 0.4559$.

pckg.	ops.	50	250	1000	2000	3000	4000	5000	7500	10000	30000
25	1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	1&2	7.9	2.5	1.7	1.7	1.7	1.7	1.7	0.0	0.0	0.0
50	1	0.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	1&2	9.0	3.1	1.2	0.9	0.3	0.3	0.3	0.3	0.0	0.0
75	1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	1&2	14.1	6.1	1.9	1.1	0.8	0.8	0.6	0.3	0.3	0.3
100	1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	1&2	19.8	11.3	5.6	3.3	2.8	1.9	1.6	1.6	1.6	1.2
125	1	0.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	1&2	26.9	15.8	8.6	4.7	4.1	3.3	2.9	2.3	1.4	0.2
150	1	0.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	1&2	29.8	15.6	8.0	5.7	4.5	3.7	3.5	2.9	2.3	1.0
200	1	10.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	1&2	45.6	27.0	13.7	9.8	9.0	7.8	7.5	5.1	3.6	1.5
250	1	21.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	1&2	53.8	31.0	17.6	10.8	8.7	7.6	6.8	5.8	4.3	1.4
300	1	35.4	0.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	1&2	61.3	35.2	22.0	16.6	13.4	11.9	10.9	8.4	6.4	2.2

Table 4.5: Two trucks, two drones and a varying number of packages (pckg.) are considered. The relative progress in percent after various periods of time (in ms) for VRD-LOC (i.e. Operations (ops.) 1 and 2) and for the variant with Operation 1 alone is provided (cf. the description below). The results are rounded to one decimal. As described in the paragraph "Runtime" in Subsection 4.6.2, this table can be deduced from Table 4.3 which contains the respective absolute results. For better readability, here we repeat the explanation of how this table can be deduced from Table 4.3: We start with the solution where the first drone is already put into use; e.g. for 200 packages, considering VRD-LOC, this corresponds to the value "1226.2" from Table 4.3. Then we consider the objective value after the second drone has been put into use, i.e. after one additional minute runtime, in the example with 200 packages this corresponds to the value "1165.0". So in total there is an improvement of $1226.2 - 1165.0 = 61.2$ which corresponds to 100%. Now we investigate, after different time steps, which proportion of the improvement is still lacking; in the example with 200 packages, after 50 ms there is 45.6% lacking or in other words already 54.4% of the improvement has been achieved within the first 50 ms; the quantity 45.6% is obtained from the absolute values in Table 4.3 by the calculation $(1192.9 - 1165.0)/61.2 \approx 0.4559$.

Now let us analyse the results from Table 4.5. First we observe that the restricted version using only Operation 1 is extremely fast, i.e. in all settings but the setting with 300 packages, after 250 ms the entire improvement was achieved already and in the setting with 300 packages one second sufficed. This is not surprising, because as there is no "inverse" operation for Operation 1, the algorithm behaves rather greedily as we already observed above; hence after a short period of time there is no additional improvement. Now let us consider VRD-LOC, i.e. both operations are used. First we notice that the needed runtime increases with the number of packages; this is as expected as this causes longer tours and hence, loosely speaking, there are more parts of the tour where the operations can be applied and hence

there are more possible solutions that are tried by VRD-LOC. For all runtime considerations keep in mind that the number of packages is split on the different vehicles, e.g. for instances with 200 packages on average each of both pairs of one drone and one truck respectively has to deliver 100 packages. We observe that even for large numbers of packages the major share of the total improvement is achieved after short runtimes; for example for 250 packages within the first three seconds more than 91.3% of the improvement that is made within one minute is achieved and after ten seconds it is already 95.7%. This shows that VRD-LOC can handle large instances within short runtimes. Small instances even have a considerably smaller runtime: For up to 150 packages, within the first second already more than 90% of the total improvement that is made within one minute is achieved and after three seconds more than 95% of the total improvement that is made within one minute is achieved. For the detailed results see Table 4.5.

Summarising, we have observed that VRD-LOC is very fast on small instances and still fast on large instances. As expected, the needed runtime increases with the number of packages. Possible reasons for the fast speed of convergence are the following. First, VRD-LOC possesses some "greedy" characteristics: Operation 1 almost always yields an improved solution, so applying Operation 1 can be considered as greedy behaviour; clearly VRD-LOC does not behave entirely greedily as applying Operation 2 almost always worsens the current solution; however, as it only inverts the impact of an operation of type 1, overall VRD-LOC still maintains some greedy characteristics which can sustain a good runtime performance. An overlapping aspect, considered from a different perspective, is that VRD-LOC does not search within the entire search space, i.e. for many instances it may be impossible to obtain an optimal solution by only applying Operations 1 and 2; thus VRD-LOC does only search in a restricted search space which can contribute to the fast runtime; of course this comes at the price that potentially better solutions may not be contained in this restricted search space. As discussed earlier, by adding additional operations for the neighbour generation, a higher solution quality can be achieved; this can be interpreted as enlarging the restricted search space.

Summary and Suggestions for Improvements In summary, we have seen that VRD-LOC outperforms Greedy and that substantial savings compared to the solutions without drones are achieved (the latter point of course is not surprising). As discussed in the paragraph "Discussion of VRD-LOC" in Subsection 4.6.1, the main purpose of VRD-LOC is to provide an initial structure that can be extended by adding further neighbour generating operations to obtain a more elaborate local search algorithm in order to improve the solution quality. In fact, in our article ([24]) we have used seven additional operations to generate neighbours thereby improving the solution quality considerably; this shows that VRD-LOC can be used as a suitable starting point; the seven additional moves are not part of this thesis but will be part of Elisabeth Kraus' doctoral thesis. We also have seen that VRD-LOC runs very fast on small instances and runs still fast on large instances. As an outlook, we want to provide several possibilities to generalise and improve VRD-LOC; some of these ideas are already mentioned in the paragraph "Discussion of VRD-LOC" in Subsection 4.6.1.

One direction that we already mentioned is to create additional operations for neighbour generation (cf. [24]). Another direction is to modify the approach such that the drones are

put into use simultaneously and not sequentially. Both previous suggestions can sustain the algorithm to also deal appropriately with instances where more drones than trucks are available. A third approach can be to extent VRD-LOC by adding "rather global" operations, i.e. by adding operations that affect more than just one truck tour and the drones that interact with that truck; for example operations that let a drone change the truck on which it travels during its tour and operations that enable to switch packages that are assigned to one truck from this truck to another truck could be considered. Note that in [24] we started to consider such rather global operations, but the approach used there needs to be refined. As a fourth approach, it might be beneficial to use more elaborate probability distributions. Until now the random decisions that are made in VRD-LOC are made in a rather naïve way; choosing more sophisticated probability distributions might improve the algorithm.

4.7 Outlook

Here we suggest some generalisations of the VRD model which constitute interesting topics for future research; for improvement suggestions for VRD-LOC we refer to the paragraph "Discussion of VRD-LOC" in Subsection 4.6.1 and to the paragraph "Summary and Suggestions for Improvements" in Subsection 4.6.2.

One interesting extension of the VRD model would be to consider settings where interaction (arrival, departure) between drones and trucks is not only possible at package destinations but also at different positions. This could be realised in different ways. One possibility is to add a finite number of additional positions at which no package has to be delivered but interactions between trucks and drones are allowed; note that in related settings such "rendezvous points" where drones and trucks can interact were already taken under consideration, cf. for example [11]. Another, though less realistic, possibility is to allow such interactions everywhere (e.g. in the Euclidean plane).

Furthermore, we assumed that a drone is uncharged after two flying edges and then needs one edge on a truck to be fully charged again; considering a more realistic charging process would be interesting. For example, the battery could last for a certain distance and be charged continuously while the drone is carried by a truck. Then, e.g., a drone could depart partially charged if it has enough energy to arrive at the next truck. Note that a restricted battery life is also investigated in [88]. There it is also suggested that packages could have assigned weights and drones cannot carry packages that are too heavy, i.e. some packages must be delivered by trucks. Numerous further extensions are possible; for example, also in a setting where the packages have assigned weights, the maximum flight distance of a drone could be considered as a function of the weight that is carried by the respective drone.

More generally, basically all variants and restrictions of various vehicle routing problems could be investigated in the context of drones supporting the trucks.

Chapter 5

Conclusion and Outlook

In Chapter 2, we have analysed the runtime and robustness of randomised algorithms for information spreading. For *Push* on the complete graph, we have determined the probability distribution of the runtime considerably more accurately than all previous work. Afterwards we have considered the robustness of *Push* against adversarial edge deletions. *Push* is often referred to as a very robust algorithm. Thus perhaps somewhat surprisingly, we have proven that on expander graphs, *Push* is not robust against *adversarial* edge deletions; however, on the positive side we have shown that adversarial edge deletions cannot prevent *Push* from informing *almost* all nodes as fast as without the deletions. We also have investigated information spreading on *random evolving graphs*. In particular, for *Pull* and *Push&Pull* (for *Push* this was already done in [34]) we have determined their expected runtimes up to constant additive terms and obtained large deviation bounds where in each round the underlying graph is a newly sampled Erdős-Rényi random graph $G(n, a/n)$ where $a > 0$ is a constant. Future research should also determine the respective runtimes in the edge-Markovian model (cf. the paragraph "Evolving Graphs" in Subsection 2.2.2).

While in Chapter 2 we investigated how one piece of information spreads, Chapter 3 addresses the question of how contradictory opinions spread. In particular, we have investigated a model introduced by Alon et al. where two competing opinions spread in a graph. One opinion is considered to be true (red) and the other to be false (blue). The opinion spreading is affected by an adversary (either the weak or the strong adversary) with certain powers that wants to promote the falsehood (see Section 3.2 for an accurate definition of the model and of the different types of adversaries). One central question is, which properties of the underlying graph assure that the majority of the nodes become red. We have shown that a lower bound on the minimum degree assures that the truth prevails in spite of the weak adversary's efforts. We also have shown that Erdős-Rényi random graphs are very robust against the strong adversary in the sense that even if we allow a large number of adversarial edge deletions in an Erdős-Rényi random graph, the majority of nodes will still become red. Furthermore, we have obtained a respective result where not only edge deletions but also edge insertions are considered. We also have changed perspectives and have proven that finding an optimal strategy from the strong adversary's perspective is NP-hard. Many possibilities for future research in this direction exist, for a respective discussion we refer to Section 3.6.

In Chapter 4, we have considered a vehicle routing problem that models the collaboration

of trucks and drones for parcel delivery. An intriguing facet of this vehicle routing variant are its inherent scheduling aspects, as often trucks have to wait for drones to carry them or vice versa. We have provided an accurate and formal definition of the problem and we have proven an equivalent characterisation of the feasibility of a solution. Moreover, we have introduced a simple local search algorithm and evaluated it empirically. The algorithm runs very fast and outperforms a canonical Greedy algorithm. For a discussion of limitations and improvement suggestions for the algorithm we refer to the paragraphs "Discussion of VRD-LOC" in Subsection 4.6.1 and "Summary and Suggestions for Improvements" in Subsection 4.6.2; for a discussion of directions for future research in the context of the collaboration of trucks and drones for parcel delivery we refer to Section 4.7.

In summary, we think that the topics covered in this thesis lie in exciting and active fields of research where still many intriguing questions can be investigated.

Bibliography

[1] E. H. L. Aarts and J. H. M. Korst. *Simulated annealing and Boltzmann machines - a stochastic approach to combinatorial optimization and neural computing*. Wiley-Interscience series in discrete mathematics and optimization. Wiley, 1990.

[2] H. Acan, A. Collevecchio, A. Mehrabian, and N. Wormald. On the push&pull protocol for rumour spreading. *ArXiv e-prints: 1411.0948v2*, 2015.

[3] L. A. Adamic, R. M. Lukose, A. R. Puniyani, and B. A. Huberman. Search in power-law networks. *Physical review E*, 64(4):046135, 2001.

[4] N. Agatz, P. Bouman, and M. Schmidt. Optimization approaches for the traveling salesman problem with drone. *ERIM Report Series Reference No. ERS-2015-011-LIS*, 2015.

[5] N. Alon, M. Feldman, O. Lev, and M. Tennenholtz. How Robust Is the Wisdom of the Crowds? In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 2055–2061, 2015.

[6] O. Angel, A. Mehrabian, and Y. Peres. The string of diamonds is tight for rumor spreading. *ArXiv e-prints: 1704.00874v2*, 2017.

[7] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[8] J. E. Beasley. Route first cluster second methods for vehicle routing. *Omega*, 11(4):403–408, 1983.

[9] B. Behdani and J. C. Smith. An Integer-Programming-Based Approach to the Close-Enough Traveling Salesman Problem. *INFORMS Journal on Computing*, 26(3):415–432, 2014.

[10] T. Bektas. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34(3):209 – 219, 2006.

[11] M. S. bin Othman, A. Shurbevski, Y. Karuno, and H. Nagamochi. Routing of carrier-vehicle systems with dedicated last-stretch delivery vehicle and fixed carrier route. *Journal of Information Processing*, 25:655–666, 2017.

[12] B. Bollobás. *Random Graphs.* Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2001.

[13] S. Boucheron, G. Lugosi, and P. Massart. A sharp concentration inequality with applications. *Random Struct. Algorithms*, 16(3):277–292, 2000.

[14] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Randomized gossip algorithms. *IEEE Transactions on Information Theory*, 52(6):2508–2530, 2006.

[15] I.-M. Chao. A tabu search method for the truck and trailer routing problem. *Computers & OR*, 29(1):33–51, 2002.

[16] M. Chen and J. Macdonald. Optimal routing algorithm in swarm robotic systems. *Available at: Department of Computer Sciences, California Institute of Technology*, 2014.

[17] F. Chierichetti, S. Lattanzi, and A. Panconesi. Almost tight bounds for rumour spreading with conductance. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 399–408. ACM, 2010.

[18] F. Chierichetti, S. Lattanzi, and A. Panconesi. Rumour spreading and graph conductance. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 1657–1663. SIAM, 2010.

[19] F. Chierichetti, S. Lattanzi, and A. Panconesi. Rumor spreading in social networks. *Theoretical Computer Science*, 412(24):2602–2610, 2011.

[20] F. Chung and L. Lu. Connected components in random graphs with given expected degree sequences. *Annals of combinatorics*, 6(2):125–145, 2002.

[21] A. Clementi, P. Crescenzi, C. Doerr, P. Fraigniaud, F. Pasquale, and R. Silvestri. Rumor spreading in random evolving graphs. *Random Structures & Algorithms*, 48(2):290–312, 2016.

[22] J. R. Current and D. A. Schilling. The Covering Salesman Problem. *Transportation Science*, 23(3):208–213, 1989.

[23] R. Daknama. Pull and Push&Pull in Random Evolving Graphs. *ArXiv e-prints: 1801.00316v2*, 2018.

[24] R. Daknama and E. Kraus. Vehicle Routing with Drones. *ArXiv e-prints: 1705.06431*, 2017.

[25] S. Daum, F. Kuhn, and Y. Maus. Rumor Spreading with Bounded In-Degree. In *Structural Information and Communication Complexity - 23rd International Colloquium, SIROCCO 2016, Helsinki, Finland, July 19-21, 2016, Revised Selected Papers*, pages 323–339, 2016.

[26] H. De Beukelaer, G. F. Davenport, G. De Meyer, and V. Fack. JAMES: A modern object-oriented Java framework for discrete optimization using local search metaheuristics. In *4th International symposium and 26th National conference on Operational Research*, pages 134–138. Hellenic Operational Research Society, 2015.

[27] G. S. de Grancy and M. Reimann. Evaluating two new heuristics for constructing customer clusters in a VRPTW with multiple service workers. *CEJOR*, 23(2):479–500, 2015.

[28] G. S. de Grancy and M. Reimann. Vehicle routing problems with time windows and multiple service workers: a systematic comparison between ACO and GRASP. *Central European Journal of Operations Research*, 24(1):29–48, 2016.

[29] D. Dellamonica, Y. Kohayakawa, M. Marciniszyn, and A. Steger. On the Resilience of Long Cycles in Random Graphs. *Electr. J. Comb.*, 15(1), 2008.

[30] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, pages 1–12. ACM, 1987.

[31] S. Dereich, C. Mönch, and P. Mörters. Typical distances in ultrasmall random networks. *Advances in Applied Probability*, 44(2):583–601, 2012.

[32] U. Derigs, M. Pullmann, and U. Vogel. Truck and trailer routing - Problems, heuristics and computational experience. *Computers & OR*, 40(2):536–546, 2013.

[33] B. Doerr, M. Fouz, and T. Friedrich. Social networks spread rumors in sublogarithmic time. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 21–30. ACM, 2011.

[34] B. Doerr and A. Kostrygin. Randomized Rumor Spreading Revisited. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 80. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[35] B. Doerr and M. Künnemann. Tight analysis of randomized rumor spreading in complete graphs. In *2014 Proceedings of the Eleventh Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*, pages 82–91. SIAM, 2014.

[36] J. Dong, N. Yang, and M. Chen. Heuristic approaches for a tsp variant: The automatic meter reading shortest tour problem. *Extending the Horizons: Advances in Computing, Optimization, and Decision Technologies*, 37:145–163, 2007.

[37] K. Dorling, J. Heinrichs, G. G. Messier, and S. Magierowski. Vehicle Routing Problems for Drone Delivery. *IEEE Trans. Systems, Man, and Cybernetics: Systems*, 47(1):70–85, 2017.

[38] M. Drexl. Branch-and-price and heuristic column generation for the generalized truck-and-trailer routing problem. *Revista de Métodos Cuantitativos para la Economía y la Empresa*, 12, 2011.

[39] M. Drexl. Synchronization in Vehicle Routing - A Survey of VRPs with Multiple Synchronization Constraints. *Transportation Science*, 46(3):297–316, 2012.

[40] M. Drexl. Branch-and-cut algorithms for the vehicle routing problem with trailers and transshipments. *Networks*, 63(1):119–133, 2014.

[41] D. J. Earl and M. W. Deem. Parallel tempering: Theory, applications, and new perspectives. *Physical Chemistry Chemical Physics*, 7(23):3910–3916, 2005.

[42] P. Erdős and A. Rényi. On a classical problem of probability theory. *Magyar Tud. Akad. Mat. Kutató Int. Közl. 6*, pages 215–220, 1961.

[43] U. Feige, D. Peleg, P. Raghavan, and E. Upfal. Randomized broadcast in networks. *Random Structures & Algorithms*, 1(4):447–460, 1990.

[44] S. M. Ferrandez, T. Harbison, T. Weber, R. Sturges, and R. Rich. Optimization of a truck-drone in tandem delivery network using k-means and genetic algorithm. *Journal of Industrial Engineering and Management*, 9(2):374–388, 2016.

[45] N. Fountoulakis, A. Huber, and K. Panagiotou. Reliable Broadcasting in Random Networks and the Effect of Density. In *INFOCOM 2010. 29th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 15-19 March 2010, San Diego, CA, USA*, pages 2552–2560, 2010.

[46] N. Fountoulakis and K. Panagiotou. Rumor spreading on random regular graphs and expanders. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 560–573, 2010.

[47] N. Fountoulakis, K. Panagiotou, and T. Sauerwald. Ultra-fast rumor spreading in social networks. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1642–1660. Society for Industrial and Applied Mathematics, 2012.

[48] A. M. Frieze and G. R. Grimmett. The shortest-path problem for graphs with random arc-lengths. *Discrete Applied Mathematics*, 10(1):57–77, 1985.

[49] C. J. Geyer. Markov chain monte carlo maximum likelihood. In *Computing Science and Statistics: Proc. of the 23rd Symposium on the Interface*, 1991.

[50] G. Giakkoupis. Tight bounds for rumor spreading in graphs of a given conductance. In *28th International Symposium on Theoretical Aspects of Computer Science, STACS 2011, March 10-12, 2011, Dortmund, Germany*, pages 57–68, 2011.

[51] G. Giakkoupis. Tight Bounds for Rumor Spreading with Vertex Expansion. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 801–815, 2014.

[52] G. Giakkoupis, Y. Nazari, and P. Woelfel. How Asynchrony Affects Rumor Spreading Time. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*, PODC '16, pages 185–194, New York, NY, USA, 2016.

[53] G. Giakkoupis and T. Sauerwald. Rumor spreading and vertex expansion. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 1623–1641, 2012.

[54] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers & OR*, 13(5):533–549, 1986.

[55] F. Glover. Tabu search — part I. *ORSA Journal on computing*, 1(3):190–206, 1989.

[56] F. Glover. Tabu search — part II. *ORSA Journal on computing*, 2(1):4–32, 1990.

[57] F. Glover. Tabu search: A tutorial. *Interfaces*, 20(4):74–94, 1990.

[58] F. Glover and M. Laguna. *Tabu Search*. Springer, 2013.

[59] B. Golden, S. Raghavan, and E. Wasil. *The Vehicle Routing Problem: Latest Advances and New Challenges*. Operations Research/Computer Science Interfaces Series. Springer US, 2008.

[60] B. L. Golden, Z. N. Azimi, S. Raghavan, M. Salari, and P. Toth. The Generalized Covering Salesman Problem. *INFORMS Journal on Computing*, 24(4):534–553, 2012.

[61] S. Greenberg and M. Mohri. Tight lower bound on the probability of a binomial exceeding its expectation. *ArXiv e-prints: 1306.1433v3*, 2013.

[62] D. J. Gulczynski, J. W. Heath, and C. C. Price. The close enough traveling salesman problem: A discussion of several heuristics. *Perspectives in Operations Research*, pages 271–283, 2006.

[63] Q. M. Ha, Y. Deville, Q. Pham, and M. H. Hà. Heuristic methods for the Traveling Salesman Problem with Drone. *ArXiv e-prints: 1509.08764v3*, 2017.

[64] H. H. Hoos and T. Stützle. Evaluating Las Vegas Algorithms: Pitfalls and Remedies. In *UAI '98: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, University of Wisconsin Business School, Madison, Wisconsin, USA, July 24-26, 1998*, pages 238–245, 1998.

[65] H. H. Hoos and T. Stützle. *Stochastic Local Search: Foundations & Applications*. Elsevier / Morgan Kaufmann, 2004.

[66] K. Hukushima and K. Nemoto. Exchange Monte Carlo method and application to spin glass simulations. *Journal of the Physical Society of Japan*, 65(6):1604–1608, 1996.

[67] S. Janson, T. Łuczak, and A. Ruciński. *Random graphs*. Wiley-Interscience series in discrete mathematics and optimization. John Wiley, 2000.

[68] R. M. Karp, C. Schindelhauer, S. Shenker, and B. Vöcking. Randomized Rumor Spreading. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 565–574, 2000.

[69] S. Kirkpatrick. Optimization by simulated annealing: Quantitative studies. *Journal of statistical physics*, 34(5-6):975–986, 1984.

[70] A. Kostrygin. *Precise Analysis of Epidemic Algorithms*. PhD thesis, Université Paris-Saclay, 2017.

[71] M. Laguna. A guide to implementing tabu search. *Investigación Operativa*, 4(1):5–25, 1994.

[72] G. Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(3):345–358, 1992.

[73] G. Laporte, M. Gendreau, J.-Y. Potvin, and F. Semet. Classical and modern heuristics for the vehicle routing problem. *International transactions in operational research*, 7(4-5):285–300, 2000.

[74] C. K. Y. Lin. A vehicle routing problem with pickup and delivery time windows, and coordination of transportable resources. *Computers & OR*, 38(11):1596–1609, 2011.

[75] S. Lin, V. F. Yu, and S. Chou. Solving the truck and trailer routing problem based on a simulated annealing heuristic. *Computers & OR*, 36(5):1683–1692, 2009.

[76] G. Lugosi. Concentration-of-measure inequalities. Lecture notes. 2004.

[77] Z. Luo, Z. Liu, and J. Shi. A Two-Echelon Cooperated Routing Problem for a Ground Vehicle and Its Carried Unmanned Aerial Vehicle. *Sensors*, 17(5):1144, 2017.

[78] N. Mathew, S. L. Smith, and S. L. Waslander. Planning Paths for Package Delivery in Heterogeneous Multirobot Teams. *IEEE Trans. Automation Science and Engineering*, 12(4):1298–1308, 2015.

[79] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

[80] N. Metropolis and S. Ulam. The monte carlo method. *Journal of the American statistical association*, 44(247):335–341, 1949.

[81] D. Mosk-Aoyama and D. Shah. Fast Distributed Algorithms for Computing Separable Functions. *IEEE Trans. Information Theory*, 54(7):2997–3007, 2008.

[82] C. C. Murray and A. G. Chu. The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Research Part C: Emerging Technologies*, 54:86–109, 2015.

[83] K. Panagiotou, X. Pérez-Giménez, T. Sauerwald, and H. Sun. Randomized Rumour Spreading: The Effect of the Network Topology. *Combinatorics, Probability & Computing*, 24(2):457–479, 2015.

[84] K. Panagiotou, A. Pourmiri, and T. Sauerwald. Faster Rumor Spreading With Multiple Calls. *Electr. J. Comb.*, 22(1):P1.23, 2015.

[85] K. Panagiotou and L. Speidel. Asynchronous Rumor Spreading on Random Graphs. *Algorithmica*, 78(3):968–989, 2017.

[86] I. T. Pérez, C. C. Corona, and J. L. Verdegay. Solving the Truck and Trailer Routing Problem with Fuzzy Constraints. *Int. J. Computational Intelligence Systems*, 8(4):713–724, 2015.

[87] B. Pittel. On Spreading a Rumor. *SIAM J. Appl. Math.*, 47(1):213–223, 1987.

[88] S. Poikonen, X. Wang, and B. L. Golden. The vehicle routing problem with drones: Extended models and connections. *Networks*, 70(1):34–43, 2017.

[89] G. Pólya. Über den zentralen Grenzwertsatz der Wahrscheinlichkeitsrechnung und das Momentenproblem. *Mathematische Zeitschrift*, 8(3):171–181, 1920.

[90] G. Polya and G. Szegö. *Problems and Theorems in Analysis I: Series. Integral Calculus. Theory of Functions.* Classics in Mathematics. Springer Berlin Heidelberg, 1997.

[91] A. Ponza. Optimization of drone-assisted parcel delivery. Master's thesis, University of Padova, 2016.

[92] T. K. Ralphs, L. Kopman, W. R. Pulleyblank, and L. E. Trotter. On the capacitated vehicle routing problem. *Mathematical programming*, 94(2-3):343–359, 2003.

[93] S. Reisser. Rumor spreading algorithms on expander graphs. Master's thesis, Ludwig-Maximilians-Universität München, 2016.

[94] E. M. Rogers. *Diffusion of innovations (5. ed.).* Free Press, 2003.

[95] T. Sauerwald and A. Stauffer. Rumor spreading and vertex expansion on regular graphs. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 462–475, 2011.

[96] S. Scheuerer. A tabu search heuristic for the truck and trailer routing problem. *Computers & OR*, 33:894–909, 2006.

[97] R. Serfling. *Approximation Theorems of Mathematical Statistics.* Wiley Series in Probability and Statistics. Wiley, 2009.

[98] R. Shuttleworth, B. L. Golden, S. Smith, and E. Wasil. Advances in meter reading: Heuristic solution of the close enough traveling salesman problem over a street network. In *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 487–501. Springer, 2008.

[99] B. Sudakov and V. H. Vu. Local resilience of graphs. *Random Struct. Algorithms*, 33(4):409–433, 2008.

[100] R. H. Swendsen and J.-S. Wang. Replica Monte Carlo simulation of spin-glasses. *Physical Review Letters*, 57(21):2607, 1986.

[101] P. Toth and D. Vigo. Models, relaxations and exact approaches for the capacitated vehicle routing problem. *Discrete Applied Mathematics*, 123(1):487–512, 2002.

[102] P. Toth and D. Vigo. *Vehicle Routing: Problems, Methods, and Applications, Second Edition*. MOS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics, 2014.

[103] V. Černỳ. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of optimization theory and applications*, 45(1):41–51, 1985.

[104] J. G. Villegas, C. Prins, C. Prodhon, A. L. Medaglia, and N. Velasco. Heuristic column generation for the truck and trailer routing problem. *International Conference on Industrial Engineering and Systems Management IESM*, pages 25–27, 2011.

[105] J. G. Villegas, C. Prins, C. Prodhon, A. L. Medaglia, and N. Velasco. A matheuristic for the truck and trailer routing problem. *European Journal of Operational Research*, 230(2):231–244, 2013.

[106] C. Wang, J. D. Hyman, A. Percus, and R. Caflisch. Parallel tempering for the traveling salesman problem. *International Journal of Modern Physics C*, 20(04):539–556, 2009.

[107] X. Wang, S. Poikonen, and B. Golden. The vehicle routing problem with drones: Several worst-case results. *Optimization Letters*, 11(4):679–697, 2017.

[108] I. Wegener. Simulated annealing beats metropolis in combinatorial optimization. *Electronic Colloquium on Computational Complexity (ECCC)*, (89), 2004.

[109] B. Yuan, M. Orlowska, and S. Sadiq. On the optimal robot routing problem in wireless sensor networks. *IEEE Transactions on Knowledge and Data Engineering*, 19(9):1252–1261, 2007.

Eidesstattliche Versicherung

(Siehe Promotionsordnung vom 12.07.2011, §8, Abs. 2 Pkt. 5.)

Hiermit erkläre ich an Eidesstatt, dass die Dissertation von mir selbstständig, ohne unerlaubte Beihilfe angefertigt ist.

Daknama, Rami

München, den 15. Oktober 2018 _____