Average-case complexity of shortest-paths problems

Volker Priebe

Dissertation
zur Erlangung des Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)
der Naturwissenschaftlich-Technischen Fakultät I
der Universität des Saarlandes

Saarbrücken
2001

Tag des Kolloquiums:	1. Juni 2001
Dekan:	Prof. Dr. Rainer Schulze-Pillot-Ziemen
Gutachter:	Prof. Dr. Kurt Mehlhorn
	Prof. Alan Frieze, Ph. D.

Abstract. We study both upper and lower bounds on the average-case complexity of shortest-paths algorithms. It is proved that the all-pairs shortest-paths problem on n-vertex networks can be solved in time $O(n^2 \log n)$ with high probability with respect to various probability distributions on the set of inputs. Our results include the first theoretical analysis of the average behavior of shortest-paths algorithms with respect to the *vertex-potential model*, a family of probability distributions on complete networks with arbitrary real arc costs but without negative cycles. We also generalize earlier work with respect to the common *uniform model*, and we correct the analysis of an algorithm with respect to the *endpoint-independent model*. For the algorithm that solves the all-pairs shortest-paths problem on networks generated according to the vertex-potential model, a key ingredient is an algorithm that solves the single-source shortest-paths problem on such networks in time $O(n^2)$ with high probability. All algorithms mentioned exploit that with high probability, the single-source shortest-paths problem can be solved correctly by considering only a rather sparse subset of the arc set. We prove a lower bound indicating the limitations of this approach. In a fairly general probabilistic model, any algorithm solving the single-source shortest-paths problem has to inspect $\Omega(n \log n)$ arcs with high probability.

Kurzzusammenfassung. In dieser Arbeit werden sowohl obere als auch untere Schranken für die *average-case*-Komplexität von Kürzeste-Wege-Algorithmen untersucht. Wir beweisen für verschiedene Wahrscheinlichkeitsverteilungen auf Netzwerken mit n Knoten, dass das *all-pairs shortest-paths problem* mit hoher Wahrscheinlichkeit in Zeit $O(n^2 \log n)$ gelöst werden kann. Insbesondere können wir dieses Laufzeit für einen Algorithmus beweisen, dessen Eingaben gemäß des *vertex-potential model* erzeugt werden, einer Familie von Wahrscheinlichkeitsverteilungen auf vollständigen Netzwerke mit reellen Kantenkosten, die jedoch keine negative Kreise besitzen. Theoretische Ergebnisse für dieses Eingabemodell waren bislang nicht bekannt. Wir verallgemeinern außerdem frühere Arbeit bezüglich des *uniform model* und korrigieren die Laufzeit-Analyse eines Algorithmus bezüglich des *endpoint-independent model*. Der Algorithmus, der das *all-pairs shortest-paths problem* auf Netzwerken löst, die gemäß des *vertex-potential model* erzeugt werden, baut entscheidend darauf auf, dass wir auch einen Algorithmus entwickeln, der das *single-source shortest-paths problem* auf solchen Netzwerken mit hoher Wahrscheinlichkeit in Zeit $O(n^2)$ löst. Alle bislang erwähnten Algorithmen nutzen aus, dass das *single-source shortest-paths problem* auch dann mit hoher Wahrscheinlichkeit korrekt gelöst werden kann, wenn wir nur einen Teil der Kantenmenge betrachten. Wir beweisen eine untere Schranke, die die Grenzen dieses Reduktionsansatzes belegt. Auf einer Klasse von Netzwerken mit ganzzahligen Kantenkosten muss jeder Algorithmus mit hoher Wahrscheinlichkeit $\Omega(n \log n)$ Kanten inspizieren, um das *single-source shortest-paths problem* zu lösen.

Acknowledgments

First and foremost, I am indebted to my *Doktorvater*, Kurt Mehlhorn, for supporting my work with unshakeable faith, patience, and generosity. Work on this thesis would have been impossible without the financial support through a Graduiertenkolleg graduate fellowship, granted by the Deutsche Forschungsgemeinschaft, and through a Ph. D. position at the Max-Planck-Institut für Informatik at Saarbrücken. It has certainly been a privilege to be a member of this institute.

Some of the material collected in this thesis has grown out of joint work with Kurt Mehlhorn, Colin Cooper, Devdatt Dubhashi, Alan Frieze, and Desh Ranjan. I would like to thank them all for sharing their insights and their enthusiasm with me. I gratefully acknowledge numerous discussions with Torben Hagerup, which helped to clarify my presentation of some of the results.

I would also like to thank both Alan Frieze and Manfred Jaeger for being willing to serve on my thesis committee.

I have never been by myself during my thesis project, since I could rely on the support of many advisors, most of them friends indeed. Of those whom I met at Saarbrücken, I would like to mention Thomas Berger, Shiva Chaudhuri, Fritz Eisenbrand, Jordan Gergov, Susan E. Hert, Uli Meyer, Peter Müller, Edgar A. Ramos, Ortwin Scheja, Jop F. Sibeyn, Gerhard Spelz, Georg Struth, Günther Torner, Jesper Larsson Träff, and Désirée Wittkowski. During all the past years, I could also feel assured of the loving understanding of my parents and my sister Dagmar, and I thank them for contributing more than their due share to the completion of this thesis.

It is not easy to put into words what I owe to Hannah Bast. I have learned a lot from her enlightening comments on the shortcomings of earlier versions of this thesis. I would like to express my heartfelt thanks to her for setting me straight whenever I had gone astray. Without her guidance, I might have struggled on forever.

Contents

Chapter 1

Introduction

A large variety of combinatorial-optimization problems can be modeled by *networks*, that is, by directed graphs in which arcs are assigned real numbers. We refer to these numbers as *arc costs*; the cost of an arc represents, for example, the amount of time or money that is consumed whenever this arc is traversed. *Shortest-paths problems* interpret arc costs as "lengths" of the arcs and ask for *distances* between pairs of vertices. For any two vertices v and w, the distance of vertex w from vertex v is defined as the infimum of the costs of all directed paths from v to w, where the cost of a path is the sum of the costs of its arcs. All distances are finite real numbers if and only if the network is strongly connected and does not contain any *negative cycles*, that is, directed cycles of negative cost.[1] We will ensure that input networks can safely be assumed to be strongly connected. The presence of a negative cycle, however, is an issue of investigation, since we allow any shortest-paths algorithm to stop its computations immediately whenever it encounters a negative cycle in the input network. We concentrate on two types of shortest-paths problems. In the *single-source shortest-paths problem*, we are interested in the distances of all vertices from a given source vertex; in the *all-pairs shortest-paths problem*, we want to compute the distances between all pairs of vertices.

Shortest-paths problems not only model problems from transportation industries or telecommunication, which suggest themselves as possible areas of application, but also model problems from areas as diverse as production planning, DNA sequence alignment, and robotics; see the references in [1, Chapter 4]. Moreover, shortest-paths problems often arise as subproblems in algorithms solving other combinatorial-optimization problems such as minimum-cost flow problems. There is

[1] Put differently, some distances are $-\infty$ if the network contains a negative cycle, since this cycle can be arbitrarily often traversed by paths. If we had required in addition that shortest paths may traverse negative cycles at most once, the problem would become at least as hard as the traveling-salesperson problem, as it was first observed by Dantzig [15]. This would mean a drastic change in the complexity of the problem.

also an intimate relation between shortest-paths problems and the theory of (discrete) dynamic programming. Not only can dynamic programming be viewed as solving a special shortest-paths problem, but many algorithms for solving shortest-paths problems can in turn be explained nicely in the framework of dynamic programming. For all these reasons, research on the design of efficient algorithms for solving shortest-paths problems dates back to the advent of computer science in the late fifties, and has been an active line of research ever since.

1.1 Worst-case analysis of shortest-paths algorithms

As a method for measuring the efficiency of an algorithm, *worst-case analysis* of its running time is well-established. Worst-case analysis asks for an upper bound on the running time that is valid on any instance of the problem to be solved by the algorithm; the bound is usually parameterized by the size of the respective problem instances. If the input instances are networks, as in shortest-paths problems, the size of the input network is commonly measured by the cardinality n of its vertex set and by the number m of its arcs. We can sometimes obtain sharper worst-case bounds on the running time of an algorithm if we use less coarse-grained parameters in the analysis; we will see examples shortly.

The worst-case complexity of known algorithms for the *single-source shortest-paths problem* depends heavily on whether or not arc costs are allowed to be negative. In fact, if all arc costs are non-negative, then Dijkstra's algorithm[2] [20] solves the single-source shortest-paths problem in near-linear time $O(m + n \log n)$, if implemented with efficient data structures such as Fibonacci heaps [28]; see also [21, 9, 81]. (We use log to denote logarithms to base e and $\log_2$ to denote logarithms to base 2.) In the general case of possibly negative arc costs, Dijkstra's algorithm has exponential worst-case running time [47, 76], but the Bellman–Ford algorithm [7, 26] solves the single-source shortest-paths problem in time $O(\nu m)$, where ν is the maximum number of arcs on a shortest path. (This follows from the usual correctness proof for the algorithm; see, for example, [1, p. 142].) The quantity ν is a first example of what we announced as less coarse-grained complexity measures in the preceding paragraph. In the worst case, however, ν can be of order n, which gives us an $O(nm)$ bound on the running time of the Bellman–Ford algorithm.

Somewhat better running times for algorithms solving the single-source shortest-paths problem are known if the arc costs are assumed to be integers from some fixed range; see [2, 33, 12, 75]. Recently, Thorup [82, 83] has shown that on *undirected* graphs with positive arc costs (which may be integers or floating-point numbers of size w bits each), the single-source shortest-paths problem can be solved in linear time $\Theta(m+n)$ in the *word RAM* model of computation. Thorup's algorithm

[2] an "obscure but powerful piece of graph theory", according to digital-economy expert Menduno [62]

and the data structures used therein exploit the fact that in the word RAM model with word size w, bitwise logical operations, arbitrary bit shifts, and arithmetic (addition and multiplication) on $O(w)$-bit operands can be performed in constant time. In this thesis, however, we adhere to the standard RAM model with unit-cost measure, and we also allow for real arc costs.

The *all-pairs shortest-paths problem* can be solved by Floyd's dynamic-programming algorithm [25], which runs in $\Theta(n^3)$ time. Another approach to the all-pairs shortest-paths problem simply solves n separate single-source shortest-paths problems, one for each source vertex. As has been observed by many authors [84, 23, 67, 48], the solution of *one* single-source shortest-paths problem allows us to transform a problem with arbitrary real arc costs into an equivalent problem with non-negative arc costs. All-pairs shortest-paths problems with arbitrary real arc costs can thus be solved by a single execution of the Bellman–Ford algorithm, followed by at most n calls of Dijkstra's algorithm. Since transformed arc costs and distances in the original problem can be computed in $\Theta(m + n^2)$ time, this results in a running time of $O(nm + n^2 \log n)$ for the all-pairs shortest-paths problem in the general case. Two algorithms for the all-pairs shortest-paths problem with non-negative arc costs were proposed that dismiss the idea of iterating over the n possible source vertices and of solving the corresponding single-source shortest-paths problem in each iteration. Instead, the algorithms of McGeoch [57] and Karger, Koller, and Phillips [49] iterate over the arcs, thereby solving the single-source shortest-paths problems simultaneously. The running time of their algorithms is $O(n|H| + n^2 \log n)$, where H denotes the set of arcs that are a shortest path between their starting point and their endpoint. The arcs in H are essential for shortest-path computations in that any shortest path is built of arcs from H. In the worst case, however, the algorithms of McGeoch or Karger, Koller, and Phillips do not improve upon the straightforward approach, since there are networks on which $|H|$ is $\Theta(m)$. For example, any network in which all arcs have the same positive cost, has $|H| = m$. (In the word RAM model of computation with word size w, Hagerup [40] describes an algorithm for the all-pairs shortest-paths problem on directed graphs with integer arc costs in the range $\{-2^w, \ldots, 2^w\}$ that runs in time $O(nm + n^2 \log\log n)$.)

All algorithms for the all-pairs shortest-paths problem mentioned above (and even the algorithm of Bellman–Ford) have a cubic worst-case running time $\Theta(n^3)$ on dense graphs, that is, on graphs where m is of the order of n^2. The known algorithms with subcubic worst-case running time do not really improve upon this situation.[3] For applications of practical relevance, this leaves only

[3]Fredman's algorithm [27] for the all-pairs shortest-paths problem with non-negative arc costs uses an efficient technique for the multiplication of matrices over the semiring $(\mathbb{R}, \min, +)$ and results in a near-cubic running time of $O(n^3((\log\log n)/\log n)^{1/3})$, which was slightly improved to $O(n^3((\log\log n)/\log n)^{1/2})$ by Takaoka [80]. If arc costs are assumed to be small integers, then shortest-paths problems can also be solved by a series of matrix multiplications over the ring of integers. If one follows this approach, the current best running time of $\tilde{O}(C^{0.681} n^{2.575})$ for the all-pairs shortest-paths problem on directed graphs can be proved for an algorithm of Zwick [91, Theorem 5.2]; see [90, 92] for a detailed account. (C denotes an upper bound on the absolute value of the arc costs; the $\tilde{O}$-notation hides factors

the combinatorial algorithms with cubic worst-case running times. If cubic running times were indeed incurred on many "natural" networks, then this would render large-scale problem instances, as they arise in applications, quite inefficient to solve. Experimental evaluations of shortest-paths algorithms show however that most of the combinatorial algorithms "usually" perform much better than predicted by their worst-case running times. These observations can sometimes be explained in the context of worst-case analysis by sound arguments, for example, if the quantities ν and $|H|$ introduced above can be proved to be substantially smaller than their worst-case bounds. (In fact, in the analysis of Chapter 4, we exploit this in the case of the networks that we consider there.) It seems however that no simple fine-grained parameter is known that would truly reflect the complexity of a shortest-paths problem in the sense that the parameter tends to be small whenever the problem is easy to solve. Worst-case analysis, even in its refined variants, has thus failed so far to provide insight into the good practical performance of algorithms. The shortest-paths algorithm of Pape [70, 71, 72] is a striking example in this respect. This algorithm, which is a variant of the Bellman–Ford algorithm, has turned out to be among the fastest algorithms in many experimental evaluations [18, 43, 11, 89], though only an exponential worst-case bound on its running time is known [76]. In general, it seems that high running times are encountered only on few but exceptionally difficult-to-solve problem instances. In fact, not many types of networks are known on which the shortest-paths algorithms mentioned above actually require their worst-case running times, and these networks appear to be quite artificial; see [58, Sect. 7.5.8] (Bellman–Ford) or [1, Ex. 5.27] (Pape).

1.2 Pros and cons of average-case analysis

Surely the most natural remedy to the failings of worst-case analysis would be to evaluate the performance of (shortest-paths) algorithms on real-life data. However, road maps and other such realistic networks usually lack two requirements that seem essential if one heads for generating input instances that allow a meaningful performance evaluation. (These requirements are discussed, for example, by Iri [44].) First, real-life networks are usually available as benchmark instances of a fixed size, though one would require them to be scalable to a wide range of input sizes. Second, it is hard to identify, let alone control, what primarily influences the behavior of algorithms on these networks.

The approach most often suggested as an alternative to real-life input instances is to generate input instances (of a certain size) according to a probability distribution on the set of possible inputs of this size. This input model gives rise to the *average-case analysis* of an algorithm. The

that are polynomial in $\log(Cn)$.) However, the approach is marred by the fact that the current techniques for fast matrix multiplication are known to be "notoriously impractical" [3, p. 1167].

quantity of interest for performance evaluation is then, for example, the *expected* running time of the algorithm—the running times of the algorithm on specific instances are averaged with respect to the probability distribution on the set of instances.

It should be stressed that for the average-case analysis of an algorithm to be meaningful, the choice of the probability distribution on the set of possible inputs is crucial. Any reasonable probability distribution fulfills the two requirements mentioned above (namely, that it can be scaled and that it can be parameterized). Beyond this mere prerequisite for asymptotic analysis, we require the probability distribution to exhibit also the following two properties.

1. The probability distribution should be realistic in the sense that problem instances that rarely occur in practice are unlikely to be generated according to the distribution and vice versa.

2. The probability distribution should be simple enough so that the behavior of algorithms on randomly generated inputs is still amenable to mathematical analysis.

It turns out however that this is rather difficult to fulfill. The first property alone is already hard to establish. For the reasons we mentioned at the beginning of this section, real-life data evades any statistical analysis; we therefore cannot hope to deduce any reasonable probabilistic information from real-life instances. By simply choosing a probability distribution on one's own, one very easily puts too much emphasis on extreme input instances. Li and Vitányi [54] provide an intricate example of a "malignant" probability distribution, which renders average-case analysis meaningless. They prove that if inputs are generated according to the so-called *universal distribution*, then the average-case complexity of algorithms essentially *equals* their worst-case complexity.[4] Naturally, we should also ensure that the distribution does not trivialize the problem to be solved by putting too much emphasis on those instances that are easily-solved special cases of the problem. Again, it is not always obvious that a probability distribution is flawed in such a way. Any specific probabilistic input model might have further drawbacks in this respect, and we will mention them when discussing the specific models. Related topics are discussed by Frieze and Reed [32, Section 7].

The second property, too, is hard to establish. The correct analysis of even simple quantities related to the average-case behavior of an algorithm usually turns out to be subtle and involved. Even if quantities (say, arc costs) were generated as the outcomes of independent random experiments, intricacies may arise from the fact that the very execution of the algorithm establishes dependencies

[4]Roughly speaking, under the universal distribution, those inputs occur with high probability that are easily algorithmically described (in the sense of the Kolmogorov complexity measure; see, for example, [53]), whereas complex or "truly" random inputs occur with low probability. Since worst-case inputs are more structured, however artificial this structure might be, than truly random inputs, the behavior on worst-case inputs dominates the average-case evaluation of any algorithm with respect to the universal distribution.

between the quantities it computes (say, the costs of arcs on a shortest path). Many analyses can be found in the literature that are based on the unjustified assumption that the quantities can be considered as independent random variables, and which are thus flawed.

The most popular model for randomly generating networks with non-negative arc costs is the so-called *uniform model*. In its basic variant, input networks in the uniform model are complete graphs with arc costs drawn at random, independently of each other, according to a common probability distribution. On the one hand, this model certainly allows for an elegant analysis of many network problems and thus has the second property required above; many of these results are surveyed by Frieze and McDiarmid [30, Sections 7 and 8]. On the other hand, complete graphs in which arc costs occur as values of independent, identically distributed random variables exhibit a homogeneity (with respect to both connectedness and distribution of arc costs) that is missing from most realistic problem instances. This makes it difficult to argue convincingly that "typical real" networks look as if they were generated according to this model.

Another severe drawback of the uniform model is that it cannot be extended straightforwardly to an input model for studying shortest-paths problems on networks with arbitrary real, possibly negative, arc costs. Suppose that networks are generated according to the uniform model with a distribution function F chosen so that arc costs have a (constant) non-zero probability of being negative. It is then very likely for an instance of the shortest-paths problem on such a network that all distances are $-\infty$. Namely, with very high probability, each vertex has an outgoing arc of negative cost, which implies that a negative cycle exists that can be reached from all vertices. Shortest-paths problems thus become solvable in *linear* expected time under the uniform model, since topological sorting allows us to decide in linear time whether the subnetwork formed by the arcs of negative costs contains a cycle.

This simple example points out yet another pitfall of average-case analysis. The chosen probability distribution might very likely generate input instances with artifacts that render average-case analysis meaningless. In fact, the algorithm in the example just given, which "solves" the all-pairs shortest-paths problem in linear time on average, was designed to exploit such artifacts of the input networks—the algorithm does not make a serious attempt to compute distances, but checks for a negative cycle. In most cases, due to the deficiencies of the input model, this allows the algorithm to finish its computations with the correct result (that is, all distances are $-\infty$) rather quickly. The algorithm can be characterized by the following features. It is clearly tailored to inputs generated according to the specific input model, and the algorithm gains its efficiency by exploiting the structural properties exhibited by the inputs. (Most likely, the algorithm performs poorly if inputs are generated according to another model.) We call algorithms of this kind *probabilistic algorithms*, taking up a notion that has been suggested by Snyder and Steele [77].

At first sight, the average performance of probabilistic algorithms might seem to be rather uncorrelated to the performance of existing, non-probabilistic algorithms or heuristics. However, probabilistic algorithms should not be underrated. Their design usually builds on insights into the structure of instances generated according to a certain probabilistic input model. The study of the structure of these randomly generated instances might add to our understanding of what makes instances difficult in the worst case. Even more importantly, this study can indeed help to better understand the performance of existing algorithms. In fact, a huge bulk of literature evaluates the performance of non-probabilistic algorithms on randomly generated instances. The outcomes of these experiments are surely influenced by the structure (the structural artifacts, possibly) that most of these instances exhibit. We feel that in the interpretations of these empirical evaluations, this influence is usually not taken into account to a sufficient extent.

1.3 Extensions of the uniform model

The arguments that we have collected in the preceding section might lead the reader to believe that the uniform model oversimplifies the task of average-case analysis. Certainly, the complete connectedness of the networks generated by the uniform model, the strong homogeneity of arc costs caused by the assumptions of independence and identical distribution, and the rare occurrence of networks with reasonably few negative cycles are shortcomings of the uniform model when it is used for the average-case analysis of shortest-paths algorithms. In this thesis, we study three extensions of the uniform model, each of which addresses one of these shortcomings.

As one such extension, it has been suggested in the literature that one considers (what we call) the *extended uniform model*. In the extended uniform model, not only the cost of an arc but also its presence is random. The *structure* of a network (on n vertices) can be interpreted as the outcome of an experiment in the well-known $\mathscr{D}(n,p)$ model—each arc in a directed graph on n vertices is present with probability p, independently of the presence of all other arcs. (We allow the parameter p to depend on n.) *Cost*s of arcs that were determined as present in this first experiment are then drawn at random in a second experiment, independently of each other, according to a common distribution function F. The advantage of the extended uniform model over the uniform model is obvious, as the extended uniform model generates not necessarily complete graphs. It would be overly optimistic, however, to consider the extended uniform model to be the cure of all the failings of the uniform model. Inputs generated according to the extended uniform model still show a considerable regularity with respect to the number of arcs leaving or entering any vertex. More severely, the extended uniform model suffers from the same drawback as the uniform model when it comes to modeling input networks for shortest-paths problems with possibly negative arc costs. Whenever $p \geq C \cdot (\log n)/n$, for a sufficiently large constant C, then it is very likely that all

vertices in a network generated according to the extended uniform model belong to a single strongly connected component and that the subnetwork formed by the arcs of negative cost contains a cycle. However, if both these events occur, then all distances are $-\infty$, and the all-pairs shortest-paths problem can be "solved" in linear time on such a network.

The other two extensions of the uniform model that we study in this thesis address these drawbacks of both the uniform model and the extended uniform model. The *vertex-potential model* extends the uniform model to a more reasonable input model for the average-case analysis of shortest-paths algorithms on networks with arbitrary real arc costs. (It generates networks that contain no negative cycles at all.) When inputs are generated according to the *endpoint-independent model*, a set of arc costs may be fixed arbitrarily, even by an adversary; only the assignment of the costs to individual arcs is random.

In the vertex-potential model, there is a *potential* $\pi(v)$ for each vertex $v \in V$ and a random variable $r(v, w)$ for each arc $(v, w) \in V \times V$. (We set $r(v, v) \equiv 0$, for $v \in V$. As usual, V denotes the vertex set of the input networks.) The arc costs are defined by

$$c(v, w) = r(v, w) - \pi(v) + \pi(w) \ , \text{ for all arcs } (v, w) \ .$$

Only the $c(v, w)$'s are revealed to an algorithm and the $r(v, w)$'s and $\pi(v)$'s are hidden parameters of the model. The variables $r(v, w)$, $v, w \in V$, $v \neq w$, are assumed to be independent, identically distributed random variables with values in the interval $[0, 1]$. (That is, the $r(v, w)$'s are generated as in the uniform model. Some additional assumptions on the common distribution function F of the $r(v, w)$'s are needed. This is made more precise in Section 4.2.) The assumption $r(v, w) \geq 0$, $v, w \in V$, guarantees that networks generated according to the vertex-potential model do not contain any negative cycles. The vertex potentials $\pi(v)$, $v \in V$, may be arbitrarily chosen. The vertex-potential model was used previously by Cherkassky, Goldberg, and Radzik [11] in an experimental evaluation of shortest-paths algorithms, but a theoretical analysis of the average behavior of algorithms with respect to the vertex-potential model was first provided by Cooper, Frieze, Mehlhorn, and Priebe [14].

The *endpoint-independent model* preserves the following property of the uniform model: If the arcs leaving a specific vertex are sorted according to their costs, then the associated endpoints occur in random order. To generate an input instance according to the endpoint-independent model, we randomly fix, for each vertex $v \in V$ independently of the other vertices, an order of the endpoints of the arcs as they appear in the sorted adjacency list of vertex v. This corresponds to drawing a permutation σ_v of V uniformly at random for each $v \in V$. The actual arc costs are then arbitrarily fixed as lists of (sorted) non-negative arc costs. It was first noticed by Bloniarz [8] that the average-case behavior of many algorithms for solving shortest-paths problems can still be analyzed when instances are generated according to the endpoint-independent model.

The endpoint-independent model has been extended to arbitrary real arc costs by Kolliopoulos and Stein [51], and to the best of our knowledge, this is the only probability distribution proposed for networks with arbitrary real arc costs (in order to generate instances of shortest-paths problems) besides the vertex-potential model.

1.4 Our results

We are now ready to describe the contributions of this thesis to the average-case analysis of shortest-paths algorithms. Throughout this description, we refer to problem instances on n vertices. We say that an event occurs *with high probability* on these instances if it occurs with probability at least $1 - O(n^{-\gamma})$ for any arbitrary but fixed constant γ.

In Chapter 3, we deal with the all-pairs shortest-paths problem under the assumption that networks are generated according to the extended uniform model. We prove in Section 3.4 that the maximum number ν of arcs on a shortest path is $O(\log n)$ with high probability. (The result was conjectured by Hassin [41].) We also show that the all-pairs shortest-paths problem can be solved in $O(n^2 \log n)$ time with high probability. Our proofs are based on a detailed review of results by Frieze and Grimmett [31], which these authors used in their design of a probabilistic algorithm that solves the all-pairs shortest-paths problem in $O(n^2 \log n)$ time with respect to the uniform model. (A similar result was independently obtained by Hassin and Zemel [42].) In Section 3.3, we extend these results to the extended uniform model. We prove that if $p \geq C \cdot (\log n)/n$, for a sufficiently large constant C, and given that the distribution of arc costs "is linear" in a neighborhood of 0, the maximal shortest-path distance in networks generated according to the extended uniform model is $O((\log n)/(np))$ with high probability. Furthermore, under the same assumptions, the set H of essential arcs can be shown to be of cardinality $O(n \log n)$ with high probability.

Our second contribution, which is presented in Chapter 4, concerns the average-case complexity of algorithms for shortest-paths problems with arbitrary real arc costs. We show that on networks generated according to the vertex-potential model, the single-source shortest-paths problem can be solved in $O(n^2)$ expected time and that the all-pairs shortest-paths problem can be solved in $O(n^2 \log n)$ expected time. In both cases our algorithms are reliable, that is, finish their computations within the respective time bounds with high probability. The running times have to be compared with the currently tightest worst-case running times of $O(\nu m)$ for the single-source shortest-paths problem and of $O(\nu m + n|H| + n^2 \log n)$ for the all-pairs shortest-paths problem on networks with arbitrary real arc costs. (Kolliopoulos and Stein [51] proposed an algorithm that needs time $O(n^2 \log n)$ with high probability for solving the single-source shortest-paths problem in the endpoint-independent model with arbitrary real arc costs. The differences between this model

and the vertex-potential model complicate a direct comparison of their result and ours.)

The algorithms that we propose in Chapters 3 and 4 are probabilistic, that is, they exploit structural properties that are exhibited with high probability by networks generated according to the respective input models. The mutual independence of arc costs allows us to apply a quite natural pruning idea, which we describe in Section 2.1.1. This idea already lies at the core of the algorithm of Frieze and Grimmett. We argue in Section 4.3 that this idea can also be utilized if networks are generated according to the vertex-potential model. Our analysis partly builds on the close relation between the vertex-potential model and the uniform model; in particular, we use our result on the maximum number of arcs on shortest paths from Section 3.4.

We also study the performance of the (non-probabilistic) algorithm of Moffat and Takaoka [65] for the all-pairs shortest-paths problem when networks are generated according to the endpoint-independent model. Earlier results for the all-pairs shortest-paths problem with respect to this model include an algorithm of Spira [78] that has expected running time $O(n^2(\log n)^2)$, which was later improved by Bloniarz [8] to $O(n^2 \log n \log^* n)$. (It is $\log^* x := 1$ for $x \leq e$ and $\log^* x := 1 + \log^* \log x$ for $x > e$.) The algorithm of Moffat and Takaoka first sorts all adjacency lists in order of increasing costs. It then solves n single-source shortest-paths problems, one for each vertex. Each single-source shortest-paths problem can be solved in $O(n \log n)$ expected time; this is due to the fact that by the assumptions of the endpoint-independent model, endpoints appear in random order in the sorted adjacency lists. The algorithm of Moffat and Takaoka thus solves the all-pairs shortest-paths problem in expected time $O(n^2 \log n)$; this also subsumes the time spent on sorting. The expected running time of the algorithm has been analyzed before. In Chapter 5, we point out some mistakes in the analyses and show how to avoid them. Moreover, we prove that the running time of the algorithm is $O(n^2 \log n)$ with high probability and not just in expectation. This running time has to be compared with the worst-case running time $O(n|H| + n^2 \log n)$ of the algorithms of McGeoch or Karger, Koller, and Phillips. Note that no better bound on $|H|$ than $O(n^2)$ can be proved in the endpoint-independent model.

Our final contribution, which we present in Chapter 6, is a result on the "inherent" complexity of the single-source shortest-paths problem, that is, its average complexity in the following sense. We provide a lower bound for the single-source shortest-paths problem in a fairly general probabilistic model. We assume that for every vertex v and each integer k, for $1 \leq k \leq n$, there is exactly one arc with cost k. (This means that algorithms are allowed to exploit the fact that arc costs are integers!) We prove that if these arc costs are distributed as in the endpoint-independent model, then any algorithm solving the single-source shortest-paths problem has to inspect $\Omega(n \log n)$ arcs with high probability. This has to be compared to our upper bound for solving single-source shortest-paths problems in Chapter 5.

Chapter 2

Preliminaries

2.1 Shortest-paths problems on directed graphs

We model shortest-paths problems on directed graphs $D = (V, A)$, where, as usual, V denotes the vertex set and $A \subseteq V \times V$ the arc set of the graph D; we often write D_n for a directed graph on n vertices.[5] Note that we allow loops, that is, arcs (v, v), for any $v \in V$. For any arc (v, w), we call v the *starting point* and w the *endpoint* of the arc. We also refer to $(v, w) \in A$ as one of v's *outgoing* arcs, as an arc *leaving* v, or as an arc *entering* w. The number of arcs leaving (entering) a vertex is called the *out-degree* (*in-degree*) of this vertex. The *adjacency list* of a vertex v contains all vertices w with $(v, w) \in A$; we interpret a corresponding entry in the adjacency list of v either as the endpoint w of the arc with starting point v or as the arc (v, w) itself, as is convenient.

A *directed path* P from v to w in D is a sequence $[(v_1, v_2), \ldots, (v_m, v_{m+1})]$ of arcs in A with $v_1 = v$ and $v_{m+1} = w$, for some $m \geq 1$. We refer to the vertices v and w as the starting point and the endpoint of P, respectively. *Directed cycles* are those directed paths for which starting point and endpoint coincide. The graph D is said to be *strongly connected* if it contains a directed path from v to w, for each tuple $(v, w) \in V \times V$.

In fact, our model for shortest-paths problems are *network*s, that is, directed graphs $D = (V, A)$ in which arc costs are given by a function from A to the reals, and we write (D, c) for a network D with arc costs c. The cost of $(v, w) \in A$ is denoted by $c(v, w)$. We extend this notation to directed paths P in (D, c), for which we define the cost $c(P)$ of P with respect to c as $\sum_{(v,w)\in P} c(v, w)$. For any pair v, w of vertices, let $\delta_c(v, w)$ be the infimum of the costs of all paths from v to w. The quantity $\delta_c(v, w)$ is referred to as the *distance* of w from v (with respect to c). All distances are

[5] We parameterize complexity bounds by the cardinality n of the vertex set. Therefore, we sometimes refer to n as the size of an instance, though, naturally, a graph on n vertices might have as many as n^2 arcs.

finite real numbers if and only if (D, c) is strongly connected and does not contain any *negative cycles*, that is, directed cycles of negative cost. The maximum of the distances (over all tuples of vertices) is called the *diameter* of D with respect to c; it is denoted by Δ_c. In Chapters 3 and 4, we denote arc costs by r if we want to stress that they are non-negative.

We concentrate on two types of shortest-paths problems on a network (D, c). For a given source vertex $s \in V$, the *single-source shortest-paths problem* asks for the distances of all vertices $v \in V$ from s. If s is fixed and no confusion is possible, we denote these distances by $\delta_c(v)$ instead of $\delta_c(s, v)$, for $v \in V$. In the *all-pairs shortest-paths problem*, we want to compute the distance between all pairs of vertices. Our output convention is as follows. If w cannot be reached by any directed path from v, then $\delta_c(v, w)$ should be set to $+\infty$. We allow any shortest-paths algorithm to stop its computations immediately whenever it encounters a negative cycle in the input graph.[6]

The following observation is crucial for solving shortest-paths problems efficiently.

> Any subpath of a shortest path is itself a shortest path from its starting point to its endpoint.

In fact, this is a special case of *Bellman's principle of optimality* [6], which lies at the core of the theory of dynamic programming. When solving a single-source shortest-paths problem with source s in a network (D, c) without negative cycles, we deduce from Bellman's principle that the distances $\delta_c(v) = \delta_c(s, v)$ must satisfy the following system of equations [7],

$$\delta_c(s) \;=\; 0 \quad \text{and} \quad \delta_c(w) \;=\; \min_{(v,w)\in A} \delta_c(v) + c(v, w) \;\;, \;\; \text{for } w \in V - \{s\} \;\; . \tag{2.1}$$

The distances $\delta_c(v)$, $v \in V$, can be determined from these equations by computing, for $v \in V$, successive approximations $d^{(1)}(v) \geq d^{(2)}(v) \geq \ldots$ that converge to $\delta_c(v)$, or by applying a relaxation procedure that is derived from a linear-program formulation of the system (2.1). A particularly nice exposition of this well-known material is given by Lawler [52, Chapter 3]. The system (2.1) of equations actually governs the optimality of path costs for any path-cost function from a rather general class, which includes, for example, the function assigning cost $\max_{(v,w)\in P} c(v, w)$ to path P; see Frieze [29].

Specifications of the two shortest-paths problems mentioned above usually include not only the task of computing the distances, but also the task of computing information on the actual shortest

[6] Mehlhorn and Näher [58, Section 7.5] require shortest-paths algorithms to output $\delta_c(v, w) = -\infty$ if there is a path from v to w that contains a negative cycle; see also [61]. In this case, paths from v to w can traverse the negative cycle arbitrarily often, and to output $\delta_c(v, w) = -\infty$ would be in accordance with our definition of distances. We have nevertheless chosen to adopt the output convention mentioned in the text, since it appears to be more common in the literature.

paths. By Bellman's principle, the collection of shortest paths from s to all other vertices, for an arbitrary but fixed source s in a network without negative cycles, forms a subtree in which all paths are directed away from s, that is, forms a so-called *arborescence rooted at vertex s*. It is easily checked that any of the shortest-paths algorithms that we encounter in the following chapters could compute these arborescences without an increase in its asymptotic running time (by simply maintaining, for any vertex v and any source s, a pointer to the predecessor of v on a shortest path from s to v), but we omit this part of the computations for ease of presentation.

2.1.1 A pruning idea

To motivate the algorithms in Chapters 3 and 4, we introduce a rather simple but general *pruning idea* (and some terminology). The idea can be used to reduce the search space of many algorithms that solve combinatorial-optimization problems modeled by directed graphs $D = (V, A)$ with arc costs c. (It was described before, for example, in work of Hassin [41].)

Given an input instance of the all-pairs shortest-paths problem with possibly negative arc costs c, suppose that we know the diameter Δ_c. By definition, the diameter is an upper bound on the cost of any shortest path. It follows from Bellman's principle that any arc a contained in a shortest path must satisfy $c(a) \leq \Delta_c$, since a itself is a shortest path from its starting point to its endpoint. All arcs whose costs are strictly greater than Δ_c are therefore *insignificant* for solving the all-pairs shortest-paths problem and could thus be pruned away from A in a preprocessing step without changing any of the distances. In turn, we call arcs whose costs are less than or equal to Δ_c *significant*, since they are possibly contained in a shortest path. As a consequence, any shortest-paths algorithm could solve the problem instance more efficiently if it were run on the network with the sparser arc set of only the significant arcs (given that the running time of the algorithm increases with the cardinality of the arc set). If only the diameter Δ_c were known before the execution of the algorithm!

However, we may just try *any cut-off value* $\bar{\Delta}$, and solve the all-pairs shortest-paths problem on the network whose arc set was pruned down to the set of *low-cost* arcs $\{a \in A\,;\, c(a) \leq \bar{\Delta}\}$. Let Δ' be the diameter on this pruned instance; it clearly satisfies $\Delta' \geq \Delta_c$. We have succeeded in determining all distances correctly if $\bar{\Delta} \geq \Delta'$, since then $\bar{\Delta} \geq \Delta_c$, and only insignificant arcs were removed in the pruning step. If we failed, then we could simply solve the optimization problem on the original input network without pruning down the arc set. Or we could take Δ' as the new cut-off value and apply the pruning idea to the original network again. This time, an optimal solution would be computed, since $\Delta' \geq \Delta_c$.

Clearly, the pruning idea is more widely applicable. For example, it can be applied to any network problem whose objective is to minimize a function φ that is defined on a family of subsets of the arc

set A, where φ has the property that for any set $\{a_1, \ldots, a_k\}$ in the domain of φ, $\varphi(\{a_1, \ldots, a_k\}) \geq \max\{c(a_1), \ldots, c(a_k)\}$.

2.2 Probabilistic preliminaries

When studying the average-case complexity of shortest-paths algorithms, we assume that input instances (of a certain size) are generated according to a probability distribution on the set of possible inputs of this size. We consider several input models (probability spaces); they are described in detail in the respective chapters. For any given probability space, we denote by **Pr** and **E** the probability measure and the corresponding expectation operator, respectively. For an instance of size n, we say that an event occurs *with high probability* if it occurs with probability at least $1 - O(n^{-\gamma})$ for any arbitrary but fixed constant $\gamma \geq 1$. To ensure a probability of failure $O(n^{-\gamma})$, in most of our statements, we have to choose sufficiently large constants, depending on the actual value of γ.

2.2.1 Stochastic dominance and order statistics

Given two random variables X and Y with distribution functions F_X and F_Y, respectively, we say that X *equals* Y *in distribution*, written $X \stackrel{\mathrm{d}}{=} Y$, if for any real t, $F_X(t) = F_Y(t)$. We say that X is *stochastically dominated* by Y, written $X \leq_{\mathrm{st}} Y$, if

$$\text{for any real } t, \quad \mathbf{Pr}(X > t) \;\leq\; \mathbf{Pr}(Y > t) \quad \text{ or, equivalently, } \quad F_X(t) \;\geq\; F_Y(t)\ . \tag{2.2}$$

In particular, $X \leq_{\mathrm{st}} Y$ implies $\mathbf{E}[X] \leq \mathbf{E}[Y]$ if the expected values of X and Y are finite. Note that equality in distribution and stochastic dominance are properties of the distribution functions of X and Y alone; the random variables X and Y need not be defined on the same probability space.

We recall two basic results about stochastic dominance, namely that, after a distribution-preserving transformation, stochastic dominance can be considered (almost surely) a *pointwise* property of random variables, and that stochastic dominance of random variables is preserved when taking sums.

Proposition 2.1 *Suppose that the random variable X is stochastically dominated by the random variable Y. Then there exist random variables X^* and Y^*, defined on the same probability space and distributed as X and Y, respectively, for which $X^* \leq Y^*$ (with probability 1).*

Proof. For any distribution function G, we consider its *generalized inverse* $G^{\leftarrow}$, defined by

$$G^{\leftarrow}(y) \;=\; \inf\{x \in \mathbb{R}\,;\, G(x) \geq y\} \;\in\; \mathbb{R} \cup \{\pm\infty\}$$

for $y \in \mathbb{R}$, where we set $\inf \varnothing := +\infty$. Since $\lim_{x \to -\infty} G(x) = 0$ and $\lim_{x \to +\infty} G(x) = 1$, $G^{\leftarrow}(y)$ is a finite real number if $y \in (0,1)$. Furthermore, since the distribution function G is increasing and right-continuous, the following equivalence is true for any $x, y \in \mathbb{R}$,

$$G^{\leftarrow}(y) \leq x \quad \iff \quad y \leq G(x) \ . \tag{2.3}$$

Let F_X and F_Y be the distribution functions of X and Y, respectively, and let U be a (single) random variable that is uniformly distributed on $(0,1)$. If we define $X^* := F_X^{\leftarrow}(U)$ and $Y^* := F_Y^{\leftarrow}(U)$, then the random variable X^* has distribution function F_X, since by (2.3), for any real x,

$$\mathbf{Pr}(X^* \leq x) \ = \ \mathbf{Pr}(F_X^{\leftarrow}(U) \leq x) \ = \ \mathbf{Pr}(U \leq F_X(x)) \ = \ F_X(x) \ ,$$

and similarly, Y^* has distribution function F_Y. Furthermore, it follows from the assumption $X \leq_{\rm st} Y$ that for any $u \in (0,1)$,

$$\{x \in \mathbb{R} \, ; \, F_X(x) \geq u\} \ \supseteq \ \{x \in \mathbb{R} \, ; \, F_Y(x) \geq u\} \ ,$$

see (2.2), which means that for any $u \in (0,1)$, $F_X^{\leftarrow}(u) \leq F_Y^{\leftarrow}(u)$. This implies $X^* \leq Y^*$. □

It is worth noting that the proof of the following lemma would work for any (component-wise) increasing function $\phi : \mathbb{R}^m \to \mathbb{R}$, not just for $\phi(x_1, \ldots, x_m) = x_1 + \cdots + x_m$.

Proposition 2.2 *Let $X_1, \ldots, X_m$ be independent random variables (defined on the same probability space), and suppose that for each i, $1 \leq i \leq m$, X_i is stochastically dominated by some random variable Y_i, where $Y_1, \ldots, Y_m$ are independent random variables (defined on the same probability space). The random variable $X_1 + \cdots + X_m$ is then stochastically dominated by $Y_1 + \cdots + Y_m$.*

Proof. For $1 \leq i \leq m$, define random variables X_i^*, Y_i^* as in Proposition 2.1, that is, $X_i^* \stackrel{\rm d}{=} X_i$, $Y_i^* \stackrel{\rm d}{=} Y_i$, and $X_i^* \leq Y_i^*$ with probability 1. We may also assume that both $X_1^*, \ldots, X_m^*$ and $Y_1^*, \ldots, Y_m^*$ are collections of independent random variables; hence, $X_1 + \cdots + X_m \stackrel{\rm d}{=} X_1^* + \cdots + X_m^*$ and $Y_1 + \cdots + Y_m \stackrel{\rm d}{=} Y_1^* + \cdots + Y_m^*$. It follows from the pointwise dominance that $X_1^* + \cdots + X_m^* \leq Y_1^* + \cdots + Y_m^*$. This implies that for any t,

$$\mathbf{Pr}(\textstyle\sum_{i=1}^m X_i > t) \ = \ \mathbf{Pr}(\sum_{i=1}^m X_i^* > t) \ \leq \ \mathbf{Pr}(\sum_{i=1}^m Y_i^* > t) \ = \ \mathbf{Pr}(\sum_{i=1}^m Y_i > t) \ ,$$

that is, $X_1 + \cdots + X_m \leq_{\rm st} Y_1 + \cdots + Y_m$. □

If the random variables X and Y take values in the positive integers, then (2.2) is clearly equivalent to the condition that $\mathbf{Pr}(X > k) \leq \mathbf{Pr}(Y > k)$ for all $k \geq 0$. In particular, for 0-1 (*Bernoulli*) random variables I' and I, we have $I' \leq_{\rm st} I$ if and only if $\mathbf{Pr}(I' > 0) = p' \leq p = \mathbf{Pr}(I > 0)$. The

random variable S is said to be *binomially distributed with parameters n and p* if $S \stackrel{\mathrm{d}}{=} I_1 + \cdots + I_n$, where $I_1, \ldots, I_n$ are independent copies of the 0-1 random variable I with *probability of success* $p = \mathbf{Pr}(I > 0)$.

Proposition 2.3 *Suppose that $n \geq 1$ and $0 \leq p' \leq p \leq 1$. If S', S, and T are random variables that are binomially distributed with parameters (n, p'), (n, p), and $(n+1, p)$, respectively, then $S' \leq_{\mathrm{st}} S \leq_{\mathrm{st}} T$.*

Proof. $S' \leq_{\mathrm{st}} S$ follows directly from the discussion preceding this proposition and from Proposition 2.2. Let I_0 and I be 0-1 random variables with $\mathbf{Pr}(I_0 > 0) = 0$ and $\mathbf{Pr}(I > 0) = p$, respectively. Observe that $S \stackrel{\mathrm{d}}{=} I_0 + S \leq_{\mathrm{st}} I + S \stackrel{\mathrm{d}}{=} T$ by Proposition 2.2. □

We also need the following result, which is slightly more general than Proposition 2.2. (A special case of this lemma appears in [74, Lemma 7].)

Lemma 2.4 *Let $X_1, \ldots, X_n, Y_1, \ldots, Y_n$ be random variables that take values in the positive integers. Suppose that each X_i, $1 \leq i \leq n$, conditioned on any possible tuple of values for $X_1, \ldots, X_{i-1}$, is stochastically dominated by Y_i, and that $Y_1, \ldots, Y_n$ are independent. Then $X^{(n)} := X_1 + \cdots + X_n$ is stochastically dominated by $Y^{(n)} := Y_1 + \cdots + Y_n$.*

Proof. The proof is by induction on n. For the base case ($n = 1$), we have $X^{(1)} = X_1$, $Y^{(1)} = Y_1$, and $\mathbf{Pr}(X_1 > k) \leq \mathbf{Pr}(Y_1 > k)$ for any $k \geq 0$ by assumption. For the induction step ($n - 1 \to n$), note that, one the one hand, for any $k \geq n$,

$$\mathbf{Pr}(Y^{(n)} > k) \;=\; \mathbf{Pr}(Y_n > k - n + 1) + \sum_{j=1}^{k-n+1} \mathbf{Pr}(Y_n = j) \cdot \mathbf{Pr}(Y^{(n-1)} > k - j) \;, \tag{2.4}$$

since $\mathbf{Pr}(Y^{(n-1)} \geq n - 1) = 1$. On the other hand, for any $k \geq n$,

$$\begin{aligned}
& \mathbf{Pr}(X^{(n)} > k) \\
= \;& \mathbf{Pr}(X^{(n-1)} > k - 1) + \sum_{j=1}^{k-n+1} \mathbf{Pr}(X_n > j \text{ and } X^{(n-1)} = k - j) \\
= \;& \mathbf{Pr}(X^{(n-1)} > k - 1) + \sum_{j=1}^{k-n+1} \mathbf{Pr}(X_n > j \mid X^{(n-1)} = k - j) \cdot \mathbf{Pr}(X^{(n-1)} = k - j) \\
\leq \;& \mathbf{Pr}(X^{(n-1)} > k - 1) + \sum_{j=1}^{k-n+1} \mathbf{Pr}(Y_n > j) \cdot \mathbf{Pr}(X^{(n-1)} = k - j) \;,
\end{aligned}$$

since X_n is stochastically dominated by Y_n, regardless of the value of $X^{(n-1)}$. Using $\mathbf{Pr}(X^{(n-1)} = k-j) = \mathbf{Pr}(X^{(n-1)} \geq k-j) - \mathbf{Pr}(X^{(n-1)} > k-j)$ and rearranging the sum, we get

$$\mathbf{Pr}(X^{(n)} > k) \;\leq\; \mathbf{Pr}(Y_n > k-n+1) + \sum_{j=1}^{k-n+1} \mathbf{Pr}(Y_n = j) \cdot \mathbf{Pr}(X^{(n-1)} > k-j) \;. \tag{2.5}$$

By the induction hypothesis and (2.4), we deduce from (2.5) that

$$\begin{aligned} \mathbf{Pr}(X^{(n)} > k) &\leq \mathbf{Pr}(Y_n > k-n+1) + \sum_{j=1}^{k-n+1} \mathbf{Pr}(Y_n = j) \cdot \mathbf{Pr}(Y^{(n-1)} > k-j) \\ &= \mathbf{Pr}(Y^{(n)} > k) \;. \end{aligned}$$

Since $\mathbf{Pr}(X^{(n)} > k) = 1 = \mathbf{Pr}(Y^{(n)} > k)$ for $0 \leq k < n$, we have thus proved that $X^{(n)} \leq_{\mathrm{st}} Y^{(n)}$. □

Let G be an arbitrary but fixed distribution function, and let X be distributed according to G. For a sample $X_1, \ldots, X_n$ of independent copies of X, we will denote by $(X_{(1:n)}, \ldots, X_{(n:n)})$ the *order statistics* of $X_1, \ldots, X_n$, that is, $X_{(1:n)} = \min\{X_1, \ldots, X_n\}$ and for $2 \leq k \leq n$, $X_{(k:n)} = \min(\{X_1, \ldots, X_n\} - \{X_{(1:n)}, \ldots, X_{(k-1:n)}\})$. The distribution functions of the order statistics can be characterized as follows. For any real x, if $I_1, \ldots, I_n$ denote independent Bernoulli variables with probability $G(x) = \mathbf{Pr}(X \leq x)$ of success each, then for any k with $1 \leq k \leq n$,

$$\mathbf{Pr}\left(X_{(k:n)} > x\right) \;=\; \mathbf{Pr}(I_1 + \cdots + I_n < k) \;. \tag{2.6}$$

The random variable $I_1 + \cdots + I_n$ in (2.6) is binomially distributed with parameters n and $G(x)$, and we can therefore apply to order statistics the stochastic-dominance relations provided in Proposition 2.3. For the sake of later reference, we mention two consequences of this kind. Since binomially distributed random variables are "stochastically increasing" in their first parameter, it follows from (2.6) that for any k, $1 \leq k \leq n$, the order statistics $X_{(k:k)}, X_{(k:k+1)}, \ldots, X_{(k:n)}$ are a "stochastically decreasing" sequence of random variables, that is,

$$X_{(k:n)} \;\leq_{\mathrm{st}}\; \cdots \;\leq_{\mathrm{st}}\; X_{(k:k)} \;. \tag{2.7}$$

Suppose that the distribution function G_X dominates the distribution function G_Y (pointwise, as in (2.2)), and let $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_n$ be independent copies of the random variables X and Y distributed according to G_X and G_Y, respectively. By our assumptions, $X \leq_{\mathrm{st}} Y$, and this carries over to the order statistics of the samples. Namely, for any real x, we have $G_X(x) \geq G_Y(x)$. Thus if the random variables $I_1 + \cdots + I_n$ and $J_1 + \cdots + J_n$ are assumed to be binomially distributed with parameters $(n, G_X(x))$ and $(n, G_Y(x))$, respectively, then, for any k,

$$\mathbf{Pr}(I_1 + \cdots + I_n < k) \;\leq\; \mathbf{Pr}(J_1 + \cdots + J_n < k)$$

since binomially distributed random variables are "stochastically increasing" in their second parameter; see Proposition 2.3. It now follows directly from (2.6) that for any k with $1 \leq k \leq n$,

$$X_{(k:n)} \ \leq_{\mathrm{st}} \ Y_{(k:n)} \ . \tag{2.8}$$

2.2.2 Concentration of random variables

For a sum X of independent, not necessarily identically distributed random variables, we are interested in bounds on the tail of the distribution of X. More precisely, our analyses will make frequent use of so-called *large-deviation estimates* for X, that is, bounds on $\mathbf{Pr}(X > x)$ or $\mathbf{Pr}(X < x)$, where x is of about the order of $\mathbf{E}[X]$. We use the following form of the well-known *Chernoff–Hoeffding bound*.

Lemma 2.5 *Let X be the sum of independent random variables $X_1, \ldots, X_m$ with values in $[0,1]$. (The X_i's need not be identically distributed.) Then for any $\varepsilon > 0$,*

$$\mathbf{Pr}(X \leq (1-\varepsilon) \cdot \mathbf{E}[X]) \ \leq \ e^{-\varepsilon^2 \mathbf{E}[X]/2} \tag{2.9}$$

and

$$\mathbf{Pr}(X \geq (1+\varepsilon) \cdot \mathbf{E}[X]) \ \leq \ \left(\frac{e^{\varepsilon}}{(1+\varepsilon)^{(1+\varepsilon)}} \right)^{\mathbf{E}[X]} \ . \tag{2.10}$$

See, for example, [56, Section 2] for a detailed account of the proof and related inequalities; we comment on the proof of Lemma 2.5 in the next paragraph. It follows from (2.10) that under the assumptions of Lemma 2.5 and for any ε with $0 \leq \varepsilon \leq 1$,

$$\mathbf{Pr}(X \geq (1+\varepsilon) \cdot \mathbf{E}[X]) \ \leq \ e^{-\varepsilon^2 \mathbf{E}[X]/3} \ ; \tag{2.11}$$

the bound in (2.10) also implies that

$$\mathbf{Pr}(X \geq x) \ \leq \ e^{-x} \quad \text{for } x \geq 9 \cdot \mathbf{E}[X] \ . \tag{2.12}$$

We now would like to pinpoint precisely where the assumptions on the random variables are used in the proof of Lemma 2.5. Its proof is based on the observation that, for any real x and any $h > 0$, $\mathbf{Pr}(X \geq x) = \mathbf{Pr}\left(e^{hX} \geq e^{hx}\right) \leq e^{-hx} \cdot \mathbf{E}[e^{hX}]$. (This rather elementary chain of inequalities is sometimes referred to as the *Bernstein inequality*.) The term $\mathbf{E}[e^{hX}]$ can be rewritten as

$$\mathbf{E}\left[e^{hX}\right] \ = \ \mathbf{E}\left[e^{h(X_1+\cdots+X_m)}\right] \ = \ \mathbf{E}\left[\prod_{i=1}^{m} e^{hX_i}\right] \ = \ \prod_{i=1}^{m} \mathbf{E}\left[e^{hX_i}\right] \ , \tag{2.13}$$

where the last equality holds, since $X_1, \ldots, X_m$ are assumed to be independent random variables. Since $0 \leq X_i \leq 1$ for any i, the convexity of the exponential function implies that $\mathbf{E}[e^{hX_i}] \leq$

$1 - \mathbf{E}[X_i] + e^h \cdot \mathbf{E}[X_i]$. Putting these observations together, we get for $x = \mathbf{E}[X] + mt$, setting $p = \mathbf{E}[X]/m$,

$$e^{m(p+t)h} \cdot \mathbf{Pr}(X \geq \mathbf{E}[X]+mt) \;\leq\; \prod_{i=1}^{m} \mathbf{E}\left[e^{hX_i}\right] \;\leq\; \prod_{i=1}^{m} (1-\mathbf{E}[X_i]+e^h \cdot \mathbf{E}[X_i]) \;\leq\; \left(1 - p + e^h \cdot p\right)^m \;,$$

where the last bound follows again from a convexity argument (namely, from the inequality between the geometric mean and the arithmetic mean) and the linearity of expectation. With a simple substitution of variables, this chain of inequalities may, for any t with $0 \leq t < 1 - p$, be rewritten as

$$\mathbf{Pr}(X \geq \mathbf{E}[X] + mt) \;\leq\; \left(\left(\tfrac{p}{p+t}\right)^{p+t} \left(\tfrac{1-p}{1-p-t}\right)^{1-p-t}\right)^m \;. \tag{2.14}$$

It is now an exercise in calculus (using no assumptions at all about the random variables) to deduce inequality (2.10) from (2.14); see [56, Theorem 2.3(b)]. If we apply the bound (2.14) on the right tail of $X = \sum_{i=1}^{m} X_i$ to the random variables $X_i' = 1 - X_i$, $1 \leq i \leq m$, for which $X' := \sum_{i=1}^{m} X_i' = m - X$, we get for any t with $0 \leq t < p = 1 - \mathbf{E}[X']/m$,

$$\mathbf{Pr}(X \leq \mathbf{E}[X] - mt) \;=\; \mathbf{Pr}\left(X' \geq \mathbf{E}[X'] + mt\right) \;\leq\; \left(\left(\tfrac{1-p}{1-p+t}\right)^{1-p+t} \left(\tfrac{p}{p-t}\right)^{p-t}\right)^m \;,$$

and mere calculus again allows us to deduce inequality (2.9); see [56, Theorem 2.3(c)]. This completes the proof of Lemma 2.5.

We have used the assumption that the X_i's are independent random variables only in equation (2.13), but equality is actually not essential there. In fact, the statement of Lemma 2.5 remains true if we replace the independence assumption by the weaker requirement that the inequality

$$\mathbf{E}\left[\prod_{i=1}^{m} e^{hX_i}\right] \;\leq\; \prod_{i=1}^{m} \mathbf{E}\left[e^{hX_i}\right] \tag{2.15}$$

holds for the random variables $X_1, \ldots, X_m$ and for any $h > 0$. Inequality (2.15) holds for a class of "strongly" negatively dependent random variables, namely, so-called negatively associated random variables, as we now argue.

Informally speaking, random variables are said to be *negatively dependent*, if they have the following property: if any one subset of the variables takes "high" values, then other (disjoint) subsets of the variables take "low" values. Many formal definitions of negative dependence of random variables have been studied in the literature. We employ a notion of negative dependence called *negative association* that first appeared in [4, Section 4] and [46]. It generalizes negative correlation (and other notions of negative dependence) to the case of vectors of random variables.

Definition 2.6 *The random variables* $(X_1, \ldots, X_m)$ *are* negatively associated *if for every index set* $I \subseteq \{1, \ldots, m\}$ *and for all non-decreasing functions* $f : \mathbb{R}^{|I|} \to \mathbb{R}$ *and* $g : \mathbb{R}^{m-|I|} \to \mathbb{R}$, *the random*

variables $f(X_i, i \in I)$ and $g(X_i, i \in \bar{I})$ are negatively correlated, that is,

$$\mathbf{E}[f(X_i, i \in I)g(X_i, i \in \bar{I})] \;\leq\; \mathbf{E}[f(X_i, i \in I)] \cdot \mathbf{E}[g(X_i, i \in \bar{I})] \;. \tag{2.16}$$

(For $I \subseteq \{1, \ldots, m\}$, we denote the set $\{1, \ldots, m\} - I$ by $\bar{I}$. A function $h : \mathbb{R}^k \to \mathbb{R}$ is said to be non-decreasing, if $h(\boldsymbol{x}) \leq h(\boldsymbol{y})$ whenever $\boldsymbol{x} \leq \boldsymbol{y}$ in the component-wise ordering on $\mathbb{R}^k$.)

The negatively associated random variables that we encounter in Section 6.1 are 0-1 vectors. Hence, we can safely assume that all expectations in Definition 2.6 exist. It readily follows from this definition by induction on m that for negatively associated random variables $(X_1, \ldots, X_m)$ and non-decreasing functions f_i, $1 \leq i \leq m$, which take values in the non-negative reals, $\mathbf{E}\left[\prod_{i=1}^m f_i(X_i)\right] \leq \prod_{i=1}^m \mathbf{E}\left[f_i(X_i)\right]$. Since for any $h > 0$, the functions $f_i(X_i) := \exp(hX_i)$, $1 \leq i \leq m$, are non-decreasing with values in the non-negative reals, this observation proves that (2.15) is satisfied for negatively associated random variables $(X_1, \ldots, X_m)$. To extend the proof of the bound on the left tail in Lemma 2.5 to the case of negatively associated random variables, we remark that for negatively associated random variables $(X_1, \ldots, X_m)$, inequality (2.16) also holds if f and g are both non-*increasing* functions. Hence, if $(X_1, \ldots, X_m)$ are negatively associated random variables with values in $[0, 1]$, so are $(1 - X_1, \ldots, 1 - X_m)$.

The following large-deviation estimates for sums of negatively associated random variables can thus be proved completely along the lines of argument for Lemma 2.5.

Lemma 2.7 *Let X be the sum of negatively associated random variables $X_1, \ldots, X_n$ with values in $[0, 1]$. (The X_i's need not be identically distributed.) Then for any $\varepsilon > 0$,*

$$\mathbf{Pr}(X < (1 - \varepsilon) \cdot \mathbf{E}[X]) \;\leq\; e^{-\varepsilon^2 \mathbf{E}[X]/2} \tag{2.17}$$

and

$$\mathbf{Pr}(X > (1 + \varepsilon) \cdot \mathbf{E}[X]) \;\leq\; \left(\frac{e^{\varepsilon}}{(1 + \varepsilon)^{(1+\varepsilon)}}\right)^{\mathbf{E}[X]} \;.$$

We use this result in Chapter 6; see page 64. For the proof that the random variables considered there are negatively associated, we utilize the fact that negative association of random variables is preserved when forming unions of independent sets, and under forming sets of non-decreasing functions that are defined on disjoint subsets of the random variables. The following proposition makes these properties more precise; see [46] for a proof.

Proposition 2.8

(a) *If the random variables $(X_1, \ldots, X_n)$ and $(Y_1, \ldots, Y_m)$ are both negatively associated and mutually independent, then the random variables $(X_1, \ldots, X_n, Y_1, \ldots, Y_m)$ are also negatively associated.*

(b) *Suppose the random variables* $(X_1, \ldots, X_n)$ *are negatively associated. Let* $I_1, \ldots, I_k \subseteq \{1, \ldots, n\}$ *be disjoint index sets, for some positive integer* k*. For* $1 \leq j \leq k$*, let* $h_j : \mathbb{R}^{|I_j|} \to \mathbb{R}$ *be non-decreasing functions, and define* $Y_j := h_j(X_i, i \in I_j)$*. Then the random variables* $(Y_1, \ldots, Y_k)$ *are also negatively associated. That is, non-decreasing functions of disjoint subsets of negatively associated variables are also negatively associated. The same is true if each* h_j *is a non-increasing function.*

Chapter 3

The extended uniform model

The basic *uniform model* for generating complete directed graphs with arc costs is defined with respect to a distribution function F. The cost of any arc is drawn according to F, independently of the costs of other arcs. Once chosen, the function F is considered to be fixed for all sizes n of instances that are to be generated according to the model (in our context, for all numbers n of vertices of the graphs). In the *extended uniform model*, the presence of each arc (and not only its cost) is random. To this end, a third parameter p (besides the distribution function F and the size n of the instances) is introduced; the parameter p is a real number from the unit interval $(0, 1]$ and may depend on n. The *structure* of a graph (on n vertices) generated according to the extended uniform model can be interpreted as the outcome of an experiment in the well-known $\mathscr{D}(n, p)$ model—each arc in a directed graph on n vertices is present with probability p, independently of the presence of all other arcs. *Costs* of arcs that were determined as present in this first experiment are then drawn at random in a second experiment, independently of each other, according to the common distribution function F. Clearly, the extended uniform model contains the uniform model as the special case $p = 1$.

Both the uniform model and the extended uniform model have been studied in the literature as input models for the probabilistic analysis of network algorithms, the uniform model quite intensively so. In the course of this chapter, we discuss work related to shortest-paths problems (in Section 3.5, in particular), and otherwise refer to the survey by Frieze and McDiarmid [30].

3.1 A probabilistic algorithm for shortest-paths problems

In this chapter, we are concerned with the all-pairs shortest-paths problem under the assumption that networks are generated according to the extended uniform model. We prove in Section 3.4

that each shortest path consists of at most $O(\log n)$ arcs with high probability; see Lemma 3.10. Apparently, the result of this lemma is new, though it was conjectured by Hassin [41] that such a result should hold. (We use this result in Chapter 4, where we analyze probabilistic shortest-paths algorithms on networks (D, c) with possibly negative arc costs.) We also show that the all-pairs shortest-paths problem can be solved in $O(n^2 \log n)$ time with high probability. Our proofs are based on a detailed review of results by Frieze and Grimmett [31], which these authors used in their design of an algorithm that solves the all-pairs shortest-paths problem in $O(n^2 \log n)$ if networks are generated according to the uniform model. (The algorithm was also suggested by Hassin and Zemel [42] and Hassin [41].)

The algorithm of Frieze and Grimmett is a prime example of what we called a *probabilistic algorithm* in the introduction. It exploits the following properties that are exhibited by graphs (D, r) with non-negative arc costs r that are generated according to the uniform model. If such instances are of size n, their diameter Δ_r is $O((\log n)/n)$ with high probability. (To prove this bound, we need the additional assumption that $F'(0) > 0$; see Section 3.2.2 for more details. We comment on the case $F'(0) = 0$ in Remark 3.8.) Since the diameter is rather small, one can make good use of the pruning idea of Section 2.2.1. (In the following, we refer to the terminology introduced in that section.) If we choose $\bar{r} = C(\log n)/n$ as a cut-off value, for a sufficiently large constant C, then $\bar{r} \geq \Delta_r$, and the pruned network that contains only the low-cost arcs (with respect to $\bar{r}$) still allows us with high probability to compute the correct distances. As for the efficacy of the pruning step, for this choice of a cut-off value $\bar{r}$, the arc set of the pruned graph has cardinality $O(n \log n)$ with high probability, since for each vertex, only the $O(\log n)$ shortest arcs leaving this vertex are low-cost (with respect to $\bar{r}$). The probabilistic algorithm of Frieze and Grimmett thus solves the all-pairs shortest-paths problem on instances generated according to the uniform model in time $O(n^2 \log n)$ with high probability by running Dijkstra's algorithm n times on the pruned network, once for each source vertex.

We prove in Section 3.3 that this algorithm can easily be adjusted to deal within the same time bounds with instances generated according to the extended uniform model, given that the parameter p is large enough to ensure with high probability that the instances are strongly connected; see Theorem 3.7. (It has been mentioned already by both Hassin and Zemel [42] and Hassin [41] that this extension should be possible, but no proofs were given in these two papers.) Namely, in Lemma 3.4, we provide an $O((\log n)/(np))$ bound on the diameter, which holds with high probability. In Lemma 3.6, we prove that on instances generated according to the extended uniform model, the corresponding bound on the number of low-cost arcs is $O(n \log n)$ with high probability. We give a detailed account of our assumptions on the extended uniform model in the following section.

3.2 Model interpretation and additional assumptions

It is convenient for our analyses in the following sections to view the extended uniform model with parameters F, n, and p as a variant of the basic uniform model, that is, as a model in which a random arc cost is assigned to each arc in the *complete* directed graph (on n vertices) according to some distribution function F_p. Namely, if we define F_p as $p \cdot F$ on the non-negative reals and with a point mass of $1 - p$ at $+\infty$, then both views of the extended uniform model, either with parameters F, n, and p or with distribution function F_p and size parameter n, are equivalent. Arcs that are assigned infinite cost by F_p correspond to arcs that are not present in the corresponding instance of the extended uniform model with parameters F, n, and p. Note that the view of the extended uniform model as a variant of the uniform model stretches the assumptions on the basic uniform model slightly, since in general, $\lim_{x\to+\infty} F_p(x) = p < 1$, and F_p depends on n through the parameter p, but we need not be concerned about this; see Lemma 3.1.

3.2.1 Two-round exposure of instances in the (extended) uniform model

Our analyses in the following sections actually focus on networks *whose adjacency lists have been sorted with respect to increasing arc costs*. According to our discussion in the last paragraph, we may assume that the unsorted arc costs were generated according to the uniform model with respect to some distribution function G, where, for the time being, G may denote either a distribution function F with $\lim_{x\to\infty} F(x) = 1$ or a distribution function F_p with a point mass at $+\infty$. If $G = F_p$, we assume in addition that ties among the arcs of cost $+\infty$, when sorted, are resolved randomly. If the arcs with starting point v, for any vertex v, are sorted according to their costs, then the order of their endpoints is a random permutation of the vertex set: In the uniform model, the costs of the arcs with starting point v are values of independent, identically distributed random variables, and therefore each permutation of the endpoints is equally likely to occur. It is helpful to formalize this observation and to interpret directed graphs (on n vertices) with sorted adjacency lists as being exposed in two *rounds*. For problem instances on n vertices, we denote their vertex sets by $V(n)$.

In the *first* round, permutations of $V(n)$ are drawn uniformly at random, independently of each other, for every vertex $v \in V(n)$. For each vertex, the corresponding permutation determines in which order the *endpoints* of the arcs appear in the sorted adjacency list of this vertex.

In the *second* round, the actual arc costs are determined. For any vertex v, the costs of the arcs as they appear in the sorted adjacency list of v are determined as the order statistics $(R_{(1:n)}, \ldots, R_{(n:n)})$ of a sample $R_1, \ldots, R_n$ of independent random variables, all distributed according to G; see Section 2.2.1. The n-tuples of sorted arc costs are determined independently of each other for any vertex

$v \in V(n)$. Furthermore, the n-tuples of sorted arc costs are independent of the outcomes of the experiments in the first round.

It should be stressed that the interpretation above applies without changes to the extended uniform model, that is, to the case $G = F_p$. The exposure of instances can there be split into the same two rounds of independent random experiments. For the sake of clarity, we mark random variables by an extra $\hat{}$ in this case. At many places in our analyses, however, it is possible to put away the cumbersome point mass at $+\infty$ of the distribution function F_p, since there are "sufficiently many" *finite* elements in the samples with common distribution function F_p. The following lemma provides a precise formulation of this intuitive idea.

Lemma 3.1 *Let F be a fixed distribution function with $\lim_{x\to\infty} F(x) = 1$, and for some $p \in (0,1)$, let F_p be the corresponding distribution function that equals $p \cdot F(x)$ for real x and has a point mass of $1-p$ at $+\infty$. Consider the order statistics $\hat{R}_{(1:n)}, \ldots, \hat{R}_{(n:n)}$ of a sample $\hat{R}_1, \ldots, \hat{R}_n$ of independent random variables, all distributed according to F_p. If the random variable L denotes the number of* finite *elements in the sample $\hat{R}_1, \ldots, \hat{R}_n$, then for any k, ℓ with $k \le \ell \le n$ and any real x,*

$$\mathbf{Pr}\left(\hat{R}_{(k:n)} > x \;\middle|\; L \ge \ell\right) \;\le\; \mathbf{Pr}\left(R_{(k:\ell)} > x\right) \;. \tag{3.1}$$

That is, conditioned on $L \ge \ell$, *the random variable $\hat{R}_{(k:n)}$ is stochastically dominated by the random variable $R_{(k:\ell)}$, the k-th order statistic of a sample of ℓ independent random variables, all distributed according to F.*

Proof. For a sample $\hat{R}_1, \ldots, \hat{R}_n$ as in the statement of the lemma, the characterization of the distribution of $\hat{R}_{(k:n)}$ from (2.6) is still true, that is, for $1 \le k \le n$ and any real x,

$$\mathbf{Pr}(\hat{R}_{(k:n)} > x) \;=\; \mathbf{Pr}(I_1 + \cdots + I_n < k) \;, \tag{3.2}$$

where $I_1, \ldots, I_n$ is a sequence of independent Bernoulli variables with probability $p \cdot F(x)$ of success each. In the present context, however, we prefer to replace the variable $I_1 + \cdots + I_n$ by a compound binomial random variable. Let $J_1, \ldots, J_n$ be a sequence of independent Bernoulli variables with probability $F(x)$ of success each. We also assume that $J_1, \ldots, J_n$ are independent of the number L of finite elements in the sample $\hat{R}_1, \ldots, \hat{R}_n$; according to our assumptions, the random variable L is binomially distributed with parameters n and p. (Clearly, L and the variables $\hat{R}_{(k:n)}$, $1 \le k \le n$, are *not* independent.) It is easy to check that the compound random variable $J_1 + \cdots + J_L$ is then binomially distributed with parameters n and $p \cdot F(x)$, where, conditioned on the event $\{L = l\}$, $J_1 + \cdots + J_L$ is distributed as $J_1 + \cdots + J_l$. For a proof of the stochastic-dominance relation in (3.1), fix any $\ell \le n$. For any k with $1 \le k \le \ell$ and any real x, it follows from (2.7) (applied to the

order statistics $R_{(k:m)}$ with $\ell \leq m \leq n$, all of which are stochastically dominated by $R_{(k:\ell)}$),

$$\begin{aligned}\mathbf{Pr}\left(\hat{R}_{(k:n)} > x \text{ and } L \geq \ell\right) &= \mathbf{Pr}(J_1 + \cdots + J_L < k \text{ and } L \geq \ell) \\ &\leq \mathbf{Pr}\left(R_{(k:\ell)} > x \text{ and } L \geq \ell\right) \ .\end{aligned}$$

The stochastic-dominance relation (3.1) follows from this inequality, since the two events $\{R_{(k:\ell)} > x\}$ and $\{L \geq \ell\}$ are independent. □

3.2.2 Assumptions on the model parameters

From a formal point of view, the extended uniform model can be studied for any triple of parameters F, n, and p. However, if we want to generate reasonable instances of shortest-paths problems, then we should restrict the range from which we choose the parameters. Otherwise, the average-case analysis of the complexity of these algorithms might not be meaningful. For example, concerning the distribution function F, if we assume $F(0) > 0$, both the uniform model and the extended uniform model do not allow a particularly enlightening analysis of shortest-paths algorithms. Indeed, as we have already argued for the uniform model in the introduction (see page 6), if $\lim_{x \uparrow 0} F(x) > 0$, that is, if there is a non-zero probability for the cost of any arc of being negative, then all distances are $-\infty$ with high probability. If F is concentrated on $[0, +\infty)$, then the assumption $F(0) > 0$ means that there is a positive probability for any arc of having cost 0, which implies that the subgraph of zero-cost arcs consists of a single strongly connected component with high probability. In this case, all distances equal 0. The same pitfalls occur in the extended uniform model if $p \geq C \cdot (\log n)/n$, for a sufficiently large constant C; see page 7 in the introduction. We assume throughout this chapter that this assumption on p holds; we fear the least confusion if we refer to it as "$np/\log n$" being "sufficiently large". The assumption is natural, since it ensures that the instances generated according to the extended uniform model are strongly connected with high probability. This could be deduced from a known threshold for strong connectivity [69], but it also follows from our analysis; see Remark 3.5.

We describe our actual assumptions on F in more detail. The first two assumptions should appear natural after the discussion above.

(A1) F is concentrated on $[0, +\infty)$ and has the properties that $F(0) = 0$ and that $F'(0)$ exists and is strictly positive.

In particular, the cost of any arc is positive with probability 1. We now explain how we exploit the assumption on $F'(0)$. (See Remark 3.8 for the case of a vanishing derivative.) By definition of

$F'(0)$ and since we assume $F(0) = 0$,

$$F(x) \;=\; \mathbf{Pr}\,(r(v,w) \le x) \;=\; F'(0)\cdot x + o(x)\ , \qquad \text{as } x \downarrow 0\ , \tag{3.3}$$

where for any arc (v,w), $r(v,w)$ denotes its cost. (In this chapter, we denote arc costs by r to stress the point that they are non-negative.) The assumption $\lim_{x\downarrow 0} F(x)/x = F'(0) > 0$ implies that the distribution of the arc costs can be approximated in a neighborhood of 0 by uniform distributions, both from above and from below. More precisely, for a distribution function F as in (A1), we fix $\varepsilon_0 = \varepsilon_0(F)$ with $0 < \varepsilon_0 < 1/(1.2\cdot F'(0))$ such that

$$0.9\cdot F'(0)\cdot x \;\le\; F(x) \;\le\; 1.1\cdot F'(0)\cdot x\ , \qquad \text{for } 0 \le x \le \varepsilon_0\ . \tag{3.4}$$

The function $F^{(\mathrm{a})}$ defined by

$$F^{(\mathrm{a})}(x) \;:=\; \begin{cases} 0.9\cdot F'(0)\cdot x & ,\quad 0 \le x < \varepsilon_0\ , \\ F(x) & ,\quad x \ge \varepsilon_0\ , \end{cases} \tag{3.5}$$

is a distribution function that satisfies

$$F^{(\mathrm{a})}(x) \;\le\; F(x) \qquad \text{for all } x \ge 0\ . \tag{3.6}$$

If $R^{(\mathrm{a})}$ and R are random variables distributed according to $F^{(\mathrm{a})}$ and F, respectively, then (3.6) just means that $R \le_{\mathrm{st}} R^{(\mathrm{a})}$; see (2.2). Note that for $0 \le x < \varepsilon_0$, $F^{(\mathrm{a})}(x) = \mathbf{Pr}(U^{(\mathrm{a})} \le x)$, where $U^{(\mathrm{a})}$ is a random variable that is uniformly distributed on $[0, 1/(0.9\cdot F'(0))]$.

From now on, we assume the distribution function F to be fixed. Furthermore, we fix a parameter ε_0 in accordance with (3.4) and a distribution function $F^{(\mathrm{a})}$ as in (3.5). Also take $C_F = 1/(F'(0)\cdot\varepsilon_0)$, and observe that, by the assumption on ε_0, we have $C_F > 1$.

Remark 3.2 The choice of uniform distributions as distributions approximating F is for technical convenience only; see how we use it in (3.13). It is clear that the assumption (3.3) on F allows for a much wider range of possible approximating distributions. In fact, Janson argues in his study [45] of quantities related to shortest-paths problems in the uniform model that a distribution function as in (3.3) can be approximated by the distribution function of an exponential distribution, which permits the use of elegant arguments exploiting the "lack of memory" property of exponential distributions; see [55] also.

3.3 Diameter and number of significant arcs

Let $D = (V, V\times V)$ denote the complete directed graph (with loops). Recall that given a designated vertex s, an arborescence rooted at vertex s is any subtree of D in which all paths are directed away

from s. Frieze and Grimmett [31] construct, for every vertex s, a spanning arborescence rooted at vertex s. For any vertex w, the cost of the path from s to w in the spanning arborescence then serves as an upper bound on the distance $\delta(s, w)$ of vertex w from vertex s, from which we obtain an upper bound on the diameter of (D, r).

3.3.1 Construction of the spanning arborescence

We assume that for any vertex v, the arcs with starting point v appear in v's adjacency list in the order of increasing arc costs $r(v, \cdot)$. An arc is said to have *v-rank k*, for $1 \leq k \leq |V|$, if it is the k-th entry in the sorted adjacency list of v. For an arbitrary but fixed vertex s, a spanning arborescence T rooted at s is constructed in stages. In the zero stage, T just consists of the single vertex s, and we define $S^{(0)} := \{s\}$. In the first stage, if (s, u) has s-rank 1, then $S^{(1)} := S^{(0)} \cup \{u\}$, and the arc (s, u) is inserted in the arc set of T if $u \neq s$. In this case, the arc (s, w_s) of s-rank 2 and the arc (u, w_u) of u-rank 1 are considered in the second stage. We set $S^{(2)} := S^{(1)} \cup \{w_s, w_u\}$; (s, w_s) is inserted in the arc set of T if $w_s \neq s$, and (u, w_u) is inserted in the arc set of T if $w_u \notin \{s, w_s, u\}$. (In the case $u = s$, only the arc (s, w_s) of s-rank 2 is considered in the second stage.) In general, suppose that after the $(k-1)$-th stage, $k \geq 1$, the vertex set of T is $S^{(k-1)} := \{v_0, v_1, \ldots, v_l\}$, for some $l = l(k) \geq 0$, and for $0 \leq j \leq l$, let k_j be the index of the stage in which v_j was added to the vertex set of T. Suppose further that (v_j, w_j) is the arc with v_j-rank $(k - k_j)$. In the k-th stage, the vertex set of T is then expanded to $S^{(k)} := S^{(k-1)} \cup \{w_j \,;\, 0 \leq j \leq l\}$. The arcs (v_j, w_j), $0 \leq j \leq l$, are incrementally tested for insertion in the arc set of T (in the order of increasing j, say), and arc (v_j, w_j) is actually inserted if and only if $w_j \neq s$ and w_j has in-degree 1 after the insertion. This maintains the invariant that T is an arborescence rooted at vertex s. The construction process terminates as soon as the vertex set of T equals V, that is, when T is indeed a spanning arborescence. It is a key ingredient for the analysis that for any k and for any vertex w added to T in the k-th stage, the ranks of the arcs that constitute the path in T from s to w sum to k. This is independent of the order in which arcs are tested for insertion in the arc set of T in one of the stages; the order of insertion does also not affect the set of vertices added to T in this stage.

It is clear that, in general, T is not the shortest-paths tree for the single-source shortest-paths problem with source s. The "greedy" construction of T, however, suggests that the paths from the root s to any other vertex are of low cost. Indeed, this can be proved if instances (D, r) are generated according to the extended uniform model; see Section 3.3.2.

For the construction process itself, Frieze and Grimmett prove that in the *uniform model* for networks on n vertices, the random number $K(n)$ of stages that is needed to complete the construction of T, for an arbitrary $s \in V(n)$, is $O(\log n)$ with high probability. More precisely, it is proved in

[31, Theorem 5.2 and Corollary 5.3] that for any $\gamma > 0$ and all $\varepsilon > 0$,

$$K(n) \;\leq\; (1+\varepsilon)(\log_2 n + (\gamma+1)\log_e n) \text{ with probability at least } 1 - o(n^{-\gamma}) \;, \qquad (3.7)$$

which implies that, for any $\gamma > 0$, $K(n) \leq (\gamma + 2.45)\log_e n = (\gamma + 2.45)\log n$ with probability at least $1 - O(n^{-\gamma})$. Intuition on this bound is given in Remark 3.3. We first explain why the analysis of $K(n)$ carries over to the extended uniform model without any changes.

How much of the randomness of the *uniform model* is used in the argument leading to the bound on $K(n)$ in (3.7)? It should be clear that the actual arc costs are *not* relevant for the random construction process under consideration. When $K(n)$ is determined, it suffices to know that, originally, for any vertex v, the costs of the arcs with starting point v are values of independent, identically distributed random variables, and hence, that if the arcs with starting point v are sorted according to their costs, then the order of their endpoints is a random permutation of the vertex set $V(n)$.

In Section 3.2.1, we formalized what we had observed in the previous paragraph as the first round of a two-round exposure of instances in the extended uniform model: In the first round, random permutations of $V(n)$ are drawn, which determine the endpoints of the sorted adjacency lists. We have already argued in Section 3.2.1 that the exposure of instances in the *extended uniform model* can be split into the very same two rounds of independent random experiments. Therefore, the bound on $K(n)$, that is, on the time needed to complete the construction of a spanning arborescence T, holds in the extended uniform model as well.

Remark 3.3 A different argument leading to a result similar to (3.7) has been given in the extended uniform model by Feige, Peleg, Raghavan, and Upfal [24]. We sketch an intuitive explanation by Pittel [73] how a bound on $K(n)$ as in (3.7), comprising two logarithmic terms with different bases, can be obtained. It has already been argued in [31, proof of Corollary 5.3] that the completion of the random arborescence "is only delayed" (this could be made precise by a stochastic-dominance argument) if each of the vertices already added to the arborescence proposes, over the consecutive stages, vertices for possible expansion of the arborescence by sampling *with* replacement from $V(n)$. (The permutations of $V(n)$ that we fix for each vertex in the random process studied above correspond to sampling *without* replacement.) For any stage $k \geq 0$, let y_k denote the cardinality of the vertex set of the "delayed" arborescence after the k-th stage. Surely, $y_0 = 1$, and it is not hard to see that for $k \geq 0$,

$$\mathbf{E}[n - y_{k+1} \mid y_k] \;=\; (n - y_k)\left(1 - \tfrac{1}{n}\right)^{y_k} \;\approx\; (n - y_k)\cdot e^{-y_k/n} \;.$$

Pittel [73, Section 2] proves that for every k, the conditional distribution of $n - y_{k+1}$ given $y_k = y$ is indeed sharply concentrated around $(n-y)\cdot e^{-y/n}$. Therefore, the (deterministic) difference

equation $n - y_{k+1} = (n - y_k) \cdot e^{-y_k/n}$ or, equivalently,

$$y_{k+1} \;=\; n - (n - y_k) \cdot e^{-y_k/n} \tag{3.8}$$

approximates quite closely how the actually random quantities y_k, $k \geq 0$, behave. If we set $x_k = y_k/n$, then (3.8) reads $x_{k+1} = f(x_k)$, for $k \geq 0$, where $f(x) = 1 - (1 - x) \cdot e^{-x}$, and $x_0 = 1/n$. The completion time of the construction process, that is, the minimum k for which $y_k > n-1$, then corresponds to the minimum k for which $x_k > 1 - 1/n$. Observe that $f(0) = 0$, $f(1) = 1$, $f'(0) = 2$, and $f'(1) = 1/e$. Therefore, x_k is almost doubled at every stage as long as it is rather small, and $1 - x_k$ decreases with a factor of approximately $1/e$ once $1 - x_k$ becomes sufficiently small. These observations correspond to the $\log_2 n$ bound and the $\log_e n$ bound, respectively, on the number of stages in (3.7). One also expects from this heuristic argument that the high-probability parameter γ appears in front of the $\log_e n$ term.

3.3.2 Path costs in the spanning arborescence

For an arbitrary but fixed vertex $s \in V(n)$, let T denote a spanning arborescence rooted at s that is constructed as in Section 3.3.1. Without any restriction on the parameter p in the extended uniform model, it might happen that T contains arcs of cost $+\infty$, that is, arcs that we actually consider as nonexistent. We now prove that for $np/\log n$ being sufficiently large, the costs of all arcs in T are finite. More precisely, we extend the analysis of Frieze and Grimmett [31] to prove that the cost of any path in T is $O((\log n)/(np))$ with high probability if networks (D_n, r) are generated according to the extended uniform model. (This result has been stated, though without proof, by Hassin and Zemel [42]. Their paper discusses in detail only the case where $p = 1$ and arc costs are uniformly distributed in the unit interval.)

For the following analysis of path costs in T, we interpret the extended uniform model as the uniform model with distribution function F_p, as described in the introductory paragraph of Section 3.2. We start by fixing the outcomes of the experiments in the first round of the two-round exposure of random instances that we described in Section 3.2.1, that is, we fix a permutation π_v of $V(n)$ for each $v \in V(n)$. This determines the shape of T, that is, the number of stages needed to construct T, and the rank of any particular arc in the adjacency list of its starting point.

For an arbitrary vertex $w \neq s$, let the unique path P_w in T from s to w consist of the arcs $(s = v_1, v_2)$, $\ldots$, $(v_m, v_{m+1} = w)$, for some $m \geq 1$, and given P_w, let k_j, for $1 \leq j \leq m$, denote the rank of (v_j, v_{j+1}) in the sorted adjacency list of vertex v_j. According to our discussion in Section 3.2.1, the cost of this path is the value of a sum of independent random variables $\hat{Q}_1, \ldots, \hat{Q}_m$, where for $1 \leq j \leq m$, the variable $\hat{Q}_j$ is distributed as $\hat{R}_{(k_j:n)}$, the k_j-th order statistic of a sample of n independent random variables, identically distributed according to the distribution function F_p.

We now prove that, provided $np/\log n$ is sufficiently large and P_w is what we call *non-extreme*, P_w has cost $O((\log n)/(np))$ with high probability. More precisely, for an arbitrary but fixed $\gamma \geq 1$, we assume that $np \geq 8(\gamma+3)C_F \cdot \log n$, where $C_F = 1/(F'(0) \cdot \varepsilon_0) > 1$, and that (this is the definition of non-extreme paths)

$$k_1 + \cdots + k_m \;\leq\; (\gamma + 3.45) \cdot \log n \;. \tag{3.9}$$

Under these assumptions, the cost $\hat{Q}_1 + \cdots + \hat{Q}_m$ of P_w can be proved to be less than $C_Q \cdot (\log n)/(np+1)$ with probability at least $1 - O(n^{-(\gamma+2)})$, where $C_Q = 7(\gamma + 3.45)/F'(0)$. For an interpretation of the definition in (3.9), recall that it follows from the rules governing the construction of T that the ranks satisfy $k_1 + \cdots + k_m \leq K(n)$ in any case, where $K(n)$ is the number of stages needed to construct T. Indeed, the definition above is closely related to the high-probability bound on $K(n)$ in (3.7).

The proof proceeds in three steps. In the first step, we show that our assumption on p ensures that with high probability, any vertex is the starting point of sufficiently many arcs of finite costs. In the second step, we exploit the fact that the result of Lemma 3.1 together with the assumptions on the distribution function F allows us to show that the random variable in question, $\hat{Q}_1 + \cdots + \hat{Q}_m$, is with high probability stochastically dominated by a sum of order statistics derived from samples of random variables that are uniformly distributed on the unit interval. In the third step, we derive a sharp bound on the distribution of this sum. Steps 2 and 3 adapt arguments already present in [31].

For the first step of the proof, let, for each vertex v, the random variable $L(v)$ describe how many of the arcs leaving v have finite costs. For n-vertex instances generated according to the extended uniform model, the random variables $L(v)$, $v \in V(n)$, are independent and each of them is binomially distributed with parameters n and p. According to the large-deviation estimate (2.9), for any vertex v, $\mathbf{Pr}(L(v) \leq \lfloor np/2 \rfloor) \;\leq\; e^{-np/8}$. Hence, since we assume that $np \geq 8(\gamma+3)C_F \cdot \log n$ with $C_F \geq 1$, the event

$$\mathscr{L}_p \;:=\; \{L(v) \geq \ell_p := \lceil np/2 \rceil \text{ for all } v \in V(n)\}$$

occurs with probability at least $1 - O(n^{-(\gamma+2)})$ if instances of size n are generated according to the extended uniform model. That is, with probability at least $1 - O(n^{-(\gamma+2)})$, *each* vertex in $V(n)$ is starting point of at least $\ell_p = \lceil np/2 \rceil$ arcs with finite costs. Hence, for any x,

$$\mathbf{Pr}\left(\hat{Q}_1 + \cdots + \hat{Q}_m > x\right) \;\leq\; \mathbf{Pr}\left(\hat{Q}_1 + \cdots + \hat{Q}_m > x \;\middle|\; \mathscr{L}_p\right) + O(n^{-(\gamma+2)}) \;.$$

For our argument in the second step, observe that the event $\mathscr{L}_p$ defined above ensures that the finiteness condition $L \geq \ell$ in Lemma 3.1 holds with $\ell = \ell_p$ for any vertex. Our assumptions imply in particular, that $\ell_p \geq np/2 \geq (\gamma + 3.45) \cdot \log n$, so that, by the definition in (3.9), for any of the

ranks k_j, $1 \le j \le m$, we have $k_j \le \ell_p$, as required for (3.1) in Lemma 3.1 to hold. It follows from inequality (3.1) that for any $1 \le j \le m$, the variable $\hat{Q}_j$ is stochastically dominated by a random variable Q_j that is distributed as $R_{(k_j:\ell)}$, the k_j-th order statistic of a sample of $\ell = \ell_p = \lceil np/2 \rceil$ independent random variables, identically distributed according to the distribution function F. We now exploit what we assume on the behavior of F in a neighborhood (to the right) of 0. If $F^{(\mathrm{a})}$ denotes the distribution function defined in (3.5), then we have $F^{(\mathrm{a})} \le F$, as observed in (3.6). Thus, by (2.8), any of the variables Q_j, $1 \le j \le m$, is stochastically dominated by a random variable $Q_j^{(\mathrm{a})}$ that is distributed as $R^{(\mathrm{a})}_{(k_j:\ell)}$, where $R^{(\mathrm{a})}_{(k_j:\ell)}$ denotes the k_j-th order statistic of a sample of ℓ independent random variables, identically distributed according to the distribution function $F^{(\mathrm{a})}$. Since stochastic dominance is preserved under taking sums of independent random variables (Proposition 2.2), these arguments imply that for any x,

$$\mathbf{Pr}\left(\hat{Q}_1 + \cdots + \hat{Q}_m > x \,\middle|\, \mathscr{L}_p\right) \;\le\; \mathbf{Pr}\left(Q_1 + \cdots + Q_m > x\right) \;\le\; \mathbf{Pr}\left(Q_1^{(\mathrm{a})} + \cdots + Q_m^{(\mathrm{a})} > x\right) \;. \tag{3.10}$$

To derive a bound on the rightmost probability in (3.10), recall that for any x with $0 \le x < \varepsilon_0$ and ε_0 as defined before (3.4), $F^{(\mathrm{a})}(x) = \mathbf{Pr}(U^{(\mathrm{a})} \le x)$, where the random variable $U^{(\mathrm{a})}$ is uniformly distributed on $[0, 1/(0.9 \cdot F'(0))]$. Hence, if $U^{(\mathrm{a})}_{(k:\ell)}$ denotes the k-th order statistic of a sample of ℓ independent copies of $U^{(\mathrm{a})}$, then, for any x with $0 \le x < \varepsilon_0$ and any $k \le \ell$,

$$\mathbf{Pr}\left(R^{(\mathrm{a})}_{(k:\ell)} \le x\right) \;=\; \mathbf{Pr}\left(U^{(\mathrm{a})}_{(k:\ell)} \le x\right) \;, \tag{3.11}$$

since both terms equal $\mathbf{Pr}(J_1 + \cdots + J_\ell \ge k)$, where $J_1, \ldots, J_\ell$ are independent Bernoulli variables, each with probability $F^{(\mathrm{a})}(x) = \mathbf{Pr}(U^{(\mathrm{a})} \le x)$ of success. It follows from (3.11) that for any $k \le \ell$, the random variables $\min\{R^{(\mathrm{a})}_{(k:\ell)}, \varepsilon_0\}$ and $\min\{U^{(\mathrm{a})}_{(k:\ell)}, \varepsilon_0\}$ are equal in distribution. Thus, if, as above, $Q_1^{(\mathrm{a})}, \ldots, Q_m^{(\mathrm{a})}$ denote independent random variables with $Q_j^{(\mathrm{a})} \stackrel{\mathrm{d}}{=} R^{(\mathrm{a})}_{(k_j:\ell)}$, for $1 \le j \le m$, and $Y_1^{(\mathrm{a})}, \ldots, Y_m^{(\mathrm{a})}$ are independent random variables with $Y_j^{(\mathrm{a})} \stackrel{\mathrm{d}}{=} U^{(\mathrm{a})}_{(k_j:\ell)}$, for $1 \le j \le m$, then, for any $x < \varepsilon_0$,

$$\begin{aligned}
\mathbf{Pr}\left(Q_1^{(\mathrm{a})} + \cdots + Q_m^{(\mathrm{a})} > x\right) &= \mathbf{Pr}\left(\min\{Q_1^{(\mathrm{a})}, \varepsilon_0\} + \cdots + \min\{Q_m^{(\mathrm{a})}, \varepsilon_0\} > x\right) \\
&= \mathbf{Pr}\left(\min\{Y_1^{(\mathrm{a})}, \varepsilon_0\} + \cdots + \min\{Y_m^{(\mathrm{a})}, \varepsilon_0\} > x\right) \\
&= \mathbf{Pr}\left(Y_1^{(\mathrm{a})} + \cdots + Y_m^{(\mathrm{a})} > x\right) \;.
\end{aligned} \tag{3.12}$$

The third step of the proof, finally, uses the fact that a bound on the probability in (3.12) can be derived completely along the lines of the proof of Lemma 4.2(b) in [31]. Namely, if $Y_1, \ldots, Y_m$ denote independent random variables with Y_j distributed as $U_{(k_j:\ell)}$, for $1 \le j \le m$, where $U_{(k:\ell)}$ denotes the k-th order statistic of a sample of ℓ independent random variables, all uniformly distributed on the *unit* interval, then, for any x with $0 \le x < \varepsilon_0$,

$$\mathbf{Pr}\left(Y_1^{(\mathrm{a})} + \cdots + Y_m^{(\mathrm{a})} > x\right) \;=\; \mathbf{Pr}\left(Y_1 + \cdots + Y_m > 0.9 \cdot F'(0) \cdot x\right).$$

Frieze and Grimmett [31, Lemma 4.2(b)] prove that for any $\kappa \geq k_1 + \cdots + k_m$ and any $\mu > 1$,

$$\mathbf{Pr}\left(Y_1 + \cdots + Y_m > \tfrac{\mu\kappa}{\ell+1}\right) \;\leq\; e^{-\mu\kappa}(e\mu)^{\kappa} \;=\; e^{-\kappa(\mu-1-\log\mu)} \;, \tag{3.13}$$

which is independent of m and the specific value of $k_1 + \cdots + k_m$; see Appendix A of this thesis also. For any non-extreme path, $k_1 + \cdots + k_m \leq (\gamma + 3.45) \cdot \log n$ by definition, and we take $\kappa = (\gamma + 3.45) \cdot \log n$ for this reason. Observe that for any $\mu \geq 3.15$, we have $\mu - 1 - \log\mu \geq 1$. Furthermore, $\ell+1 = \lceil np/2 \rceil + 1 \geq (np+1)/2$, and with $\mu = 3.15$ and κ as defined above, we derive from (3.13) that

$$\mathbf{Pr}\left(Y_1 + \cdots + Y_m > \tfrac{6.3(\gamma+3.45)\cdot\log n}{np+1}\right) \;=\; O(n^{-(\gamma+2)}) \;, \tag{3.14}$$

which through combination of the above estimates reads

$$\mathbf{Pr}\left(\hat{Q}_1 + \cdots + \hat{Q}_m > C_Q \cdot \tfrac{\log n}{np+1}\right) \;=\; O(n^{-(\gamma+2)}) \;, \tag{3.15}$$

where $C_Q = 7(\gamma + 3.45)/F'(0)$. Our assumptions ensure that $np \geq C_Q/\varepsilon_0 \cdot \log n$, which implies that $C_Q \cdot \frac{\log n}{np+1}$ is indeed strictly less than ε_0, as we assumed in the calculations leading to (3.12).

Recall that so far all probabilities in this section have been implicitly conditioned on the n-tuple $\boldsymbol{\pi} = (\pi_v)_{v \in V(n)}$, where the π_v's are permutations of $V(n)$ drawn in the first-round experiments of the two-round exposure of instances generated according to the extended uniform model; see our discussion at the beginning of this section. In the proof of the following lemma, we make these conditional probabilities explicit. With the help of the considerations above, we can prove the following bound on the diameter $\Delta_r = \max\{\delta_r(v,w)\,;\, v,w \in V\}$ in networks (D,r) that are generated according to the extended uniform model.

Lemma 3.4 *Suppose networks (D_n, r) are generated according to the extended uniform model with parameters F and p, where F satisfies assumption (A1). If $np/\log n$ is sufficiently large, then the diameter of the generated graphs is $O((\log n)/(np))$ with high probability.*

Proof. Let $\Delta_r = \max\{\delta_r(v,w)\,;\, v,w \in V(n)\}$ denote the diameter. The quantities $\Delta_r(v) := \max\{\delta_r(v,w)\,;\, w \in V(n)\}$, for $v \in V(n)$, are identically distributed so that for any x,

$$\mathbf{Pr}(\Delta_r > x) \;\leq\; \sum_{v \in V(n)} \mathbf{Pr}(\Delta_r(v) > x) \;=\; n \cdot \mathbf{Pr}(\Delta_r(s) > x) \;, \tag{3.16}$$

where s is an arbitrary vertex in $V(n)$ that we assume to be fixed for the remaining proof. The lemma follows from (3.16) if we prove that $\Delta_r(s)$ is $O((\log n)/(np))$ with high probability.

Any n-tuple $\boldsymbol{\pi}$ of permutations of $V(n)$ that we choose in the first round of the two-round exposure of instances in the extended uniform model determines a spanning arborescence $T = T(\boldsymbol{\pi})$ rooted at vertex s; let $K(\boldsymbol{\pi})$ denote the number of stages that it takes to construct $T(\boldsymbol{\pi})$. If we write

$\Delta_r(T)$ for the maximum of path costs in $T = T(\boldsymbol{\pi})$, then surely $\Delta_r(s) \le \Delta_r(T)$ on this instance. Thus, for any x and any κ,

$$\begin{aligned}\mathbf{Pr}(\Delta_r(s) > x) &\le \sum_{\boldsymbol{\pi}} \mathbf{Pr}\left(\boldsymbol{\pi} \text{ chosen and } \Delta_r(T) > x\right) \\ &\le \sum_{\boldsymbol{\pi}, K(\boldsymbol{\pi}) > \kappa} \mathbf{Pr}(\boldsymbol{\pi}) \;+\; \sum_{\boldsymbol{\pi}, K(\boldsymbol{\pi}) \le \kappa} \mathbf{Pr}\left(\boldsymbol{\pi} \text{ chosen and } \Delta_r(T) > x\right) \;, \quad (3.17)\end{aligned}$$

where $\mathbf{Pr}(\boldsymbol{\pi})$ is a shorthand for $\mathbf{Pr}(\boldsymbol{\pi}$ chosen). Let $\gamma \ge 1$ be arbitrary but fixed. With $K(n)$ denoting, as before, the random number of stages needed to complete the construction of the spanning arborescence rooted at s on instances of size n, the first sum in (3.17) just equals $\mathbf{Pr}(K(n) > \kappa)$, and we know from (3.7) that this probability is $O(n^{-(\gamma+1)})$ if $\kappa \ge (\gamma+3.45)\cdot\log n =: C_K \cdot \log n$. Observe that if we condition on the choice of $\boldsymbol{\pi}$ with $K(\boldsymbol{\pi}) \le C_K \cdot \log n$, then any path in $T = T(\boldsymbol{\pi})$ is non-extreme according to the definition in (3.9). For any vertex $w \in V(n)$, denote by P_w the unique path in T from vertex s to w, and let $r(P_w)$ be the cost of this path. By the observation we just made, it follows from (3.15) that for $x \ge C_Q \cdot (\log n)/(np+1) = 7(\gamma+3.45)/F'(0)\cdot(\log n)/(np+1)$,

$$\mathbf{Pr}(r(P_w) > x \mid \boldsymbol{\pi} \text{ chosen with } K(\boldsymbol{\pi}) \le C_K \cdot \log n) \;=\; O(n^{-(\gamma+2)}) \;;$$

recall that the proof of this estimate requires that $np/\log n$ is sufficiently large. The estimate above implies for any of the summands in the second sum of (3.17),

$$\mathbf{Pr}\left(\boldsymbol{\pi} \text{ chosen and } \Delta_r(T) > x\right) \;\le\; \sum_{w \in V(n)} \mathbf{Pr}(r(P_w) > x \mid \boldsymbol{\pi}) \cdot \mathbf{Pr}(\boldsymbol{\pi}) \;=\; \mathbf{Pr}(\boldsymbol{\pi}) \cdot n \cdot O(n^{-(\gamma+2)})$$

if x and κ are chosen as above. We conclude that with $x \ge C_Q \cdot (\log n)/(np+1)$

$$\mathbf{Pr}(\Delta_r(s) > x) \;=\; O(n^{-(\gamma+1)}) \;+\; \mathbf{Pr}(K(n) \le (\gamma+3.45)\cdot\log n) \cdot O(n^{-(\gamma+1)}) \;=\; O(n^{-(\gamma+1)}) \;,$$

which, by (3.16), proves that Δ_r is less than $C_Q \cdot (\log n)/(np+1) = O((\log n)/(np))$ with probability at least $1 - O(n^{-\gamma})$. □

Remark 3.5 Networks with finite diameter are necessarily strongly connected, and if we prune away all arcs of cost greater than the diameter, then the pruned network is still strongly connected. If networks are generated according to the extended uniform model, then these pruned networks occur as if they were generated according to the $\mathscr{D}(n, p \cdot F(\Delta_r))$ model. For any $\gamma \ge 1$, if $\Delta_r = 7(\gamma+3.45)/F'(0)\cdot(\log n)/(np+1)$ and if $np/\log n$ is as large as required in the proof of Lemma 3.4, we have $\Delta_r < \varepsilon_0$, and thus

$$p \cdot F(\Delta_r) \;\le\; p \cdot 1.1 \cdot F'(0) \cdot \Delta_r \;\le\; 7.7(\gamma+3.45)\cdot(\log n)/n \;.$$

This implies that for any $p' \ge 7.7(\gamma+3.45)\cdot(\log n)/n$, directed graphs generated according to the $\mathscr{D}(n, p')$ model are strongly connected with probability at least $1 - O(n^{-\gamma})$.

3.3.3 Number of significant arcs

We also prove an analogue of [31, Lemma 4.3]: For networks (D_n, r) generated according to the extended uniform model, the cost of any arc with rank k, where $k > B \log n$ (for a sufficiently large constant B), is greater than the diameter Δ_r with high probability. In view of the discussion in Section 3.1, we state this result more algorithmically in terms of a cut-off value.

Lemma 3.6 *Suppose networks (D_n, r) are generated according to the extended uniform model with parameters F and p, where F satisfies assumption (A1). Then there are a cut-off value $\bar{\Delta} = \Theta((\log n)/(np))$ and some $k = \Theta(\log n)$ so that for any vertex, its arc of rank k has cost greater than $\bar{\Delta}$ with high probability. The set $\bar{A} = \{(v, w)\,;\, r(v, w) \leq \bar{\Delta}\}$ of low-cost arcs (with respect to this cut-off value) is thus of cardinality $O(n \log n)$ with high probability. Furthermore, if $np/\log n$ is sufficiently large, the set $\bar{A}$ contains with high probability the set $\{(v, w)\,; r(v, w) \leq \Delta_r\}$ of arcs that are significant for shortest-paths computations in (D_n, r).*

Proof. Let $\gamma \geq 1$ be arbitrary but fixed. We prove that for any vertex, the cost of its arc of rank $\lceil 70(\gamma + 3.45)C_F \cdot \log n \rceil$, with $C_F = 1/(F'(0) \cdot \varepsilon_0) > 1$, is greater than the cut-off value $\bar{\Delta} = 7(\gamma + 3.45)/F'(0) \cdot (\log n)/(np)$ with probability at least $1 - O(n^{-(\gamma+1)})$. We distinguish two cases depending on the value of p. The condition of the first case, $np \geq 7(\gamma + 3.45)C_F \cdot \log n$, can equivalently be stated as $\bar{\Delta} \leq \varepsilon_0$. For an arbitrary but fixed vertex v, let the random variable $L_{\bar{\Delta}}$ then describe how many of the arcs leaving v have costs at most $\bar{\Delta}$. If networks (D_n, r) are generated according to the extended uniform model, the random variable $L_{\bar{\Delta}}$ is binomially distributed with parameters n and $p \cdot F(\bar{\Delta})$; in particular, by (3.4),

$$\mathbf{E}[L_{\bar{\Delta}}] \;=\; np \cdot F(\bar{\Delta}) \;\leq\; np \cdot 1.1 \cdot F'(0) \cdot \bar{\Delta} \;=\; 7.7(\gamma + 3.45) \cdot \log n \;\;.$$

Hence, $k = \lceil 70(\gamma + 3.45)C_F \cdot \log n \rceil$ satisfies $k \geq 9 \cdot \mathbf{E}[L_{\bar{\Delta}}]$, and it follows from the Chernoff–Hoeffding bound (2.12) that $\mathbf{Pr}(L_{\bar{\Delta}} \geq k)$ is at most $e^{-k} = O(n^{-(\gamma+1)})$. Intuitively speaking, in the second case, only logarithmically many arcs have finite costs anyway. More formally speaking, let the random variable L describe how many of the arcs leaving an arbitrary but fixed vertex v have finite costs. In the extended uniform model, L is binomially distributed with parameters n and p. Hence, if $np < 7(\gamma + 3.45)C_F \cdot \log n$, then $k = \lceil 70(\gamma + 3.45)C_F \cdot \log n \rceil$ is strictly greater than $9 \cdot \mathbf{E}[L]$, and it follows again from the Chernoff–Hoeffding bound (2.12) that $\mathbf{Pr}(L \geq k)$ is at most $e^{-k} = O(n^{-(\gamma+1)})$.

It follows immediately that the set $\bar{A}$ is of cardinality at most $n \cdot \lceil 70(\gamma + 3.45)C_F \cdot \log n \rceil$ with probability $1 - O(n^{-\gamma})$. The last statement in the lemma follows from the fact that for $np/\log n \geq 9(\gamma+3)C_F$, the event $\{\Delta_r \leq \bar{\Delta}\}$ occurs with probability at least $1 - O(n^{-\gamma})$, as proved in Lemma 3.4. □

Theorem 3.7 *Suppose networks (D_n, r) are generated according to the extended uniform model with parameters F and p, where F satisfies assumption (A1). If $np/\log n$ is sufficiently large, then the all-pairs shortest-paths problem can be solved in time $O(n^2 \log n)$ with high probability.*

Proof. Let $\bar{\Delta}$ be chosen as in Lemma 3.6. We prune away all arcs whose costs exceed $\bar{\Delta}$ and run Dijkstra's algorithm n times on the pruned network, once for each source vertex. According to Lemma 3.6, this takes total time $O(n^2 \log n)$ with high probability, since the arc set of the pruned network has cardinality $O(n \log n)$ with high probability. We have solved the all-pairs shortest-paths problem correctly, if the maximum of the computed distances is less than or equal to $\bar{\Delta}$, which, according to Lemma 3.6, happens with high probability. If the inequality does not hold, we simply run Floyd's algorithm in time $\Theta(n^3)$ on the original network. □

Remark 3.8 Suppose networks are generated according to the uniform model with parameter G, where $G'(0) = 0$. Can bounds on the diameter and the number of significant arcs be derived also in this case? This has been studied by Walley and Tan [85], and we give a plausibility argument in support of their statements. For some $a > 1$, let $G(x) = G^* \cdot x^a + o(x^a)$, as $x \downarrow 0$; we assume furthermore that G^{-1} exists and is concave in a neighborhood (to the right) of 0. If R is distributed according to G and U is uniformly distributed on $(0, 1)$, then $G^{-1}(U)$ equals R in distribution. Likewise, for the corresponding order statistics, we have $(R_{(1:n)}, \ldots, R_{(n:n)}) \stackrel{\mathrm{d}}{=} (G^{-1}(U_{(1:n)}), \ldots, G^{-1}(U_{(n:n)}))$. Similarly to the proofs in Section 3.3.2, we have to bound the right tail of sums of independent random variables Q_i, $1 \le i \le m$, where $Q_i \stackrel{\mathrm{d}}{=} R_{(k_i:n)}$, for $k_1, \ldots, k_m$ as in (3.9). If Y_i, $1 \le i \le m$, denote independent random variables with $Y_i \stackrel{\mathrm{d}}{=} U_{(k_i:n)}$, then $\sum_{i=1}^m Q_i \stackrel{\mathrm{d}}{=} \sum_{i=1}^m G^{-1}(Y_i)$. By the assumption that G^{-1} is concave, $\sum_{i=1}^m G^{-1}(y_i)/m \le G^{-1}\left((\sum_{i=1}^m y_i)/m\right)$, for any $m \ge 1$ and any $y_1, \ldots, y_m \ge 0$. Thus, with $\sum_i$ as a shorthand for $\sum_{i=1}^m$, for any x,

$$\mathbf{Pr}\left(\textstyle\sum_i G^{-1}(Y_i) > x\right) \;\le\; \mathbf{Pr}\left(m \cdot G^{-1}((\textstyle\sum_i Y_i)/m) > x\right) \;=\; \mathbf{Pr}\left(\textstyle\sum_i Y_i > m \cdot G(x/m)\right) \;.$$

Since $m = O(\log n)$ with high probability by (3.7), we deduce from the tail bound (3.14) that $\sum_i Q_i$ is $O((\log n)/n^{1/a})$ with high probability. From arguments as in the proof of Lemma 3.6, it follows that with high probability, $O((\log n)^a)$ arcs are significant for shortest-paths computations.

3.4 Number of arcs on shortest paths

We have argued above that each of the spanning arborescences T rooted at s, for some $s \in V(n)$, has depth $O(\log n)$, that is, that these short (in cost, though not necessarily shortest-cost) paths in T consist of $O(\log n)$ arcs with high probability. One might imagine that a considerable fraction of *shortest* paths has more (but shorter) arcs, but we will prove in Lemma 3.10 that, with high

probability, shortest paths also consist of only logarithmically many arcs. Put differently, paths with more than logarithmically many arcs have with high probability cost greater than the diameter. This will be made precise with the help of the following lemma. Similar observations, together with the efficiency of quite involved data structures, have been exploited recently by Meyer [63] in the design of an algorithm that solves the single-source shortest-paths problem on an *arbitrary* network in *linear* time if arc costs are chosen independently at random, according to the uniform distribution on $[0,1]$. (This result is of interest, for example, if the network has $o(n \log n)$ arcs. The analysis leading to this result has been simplified by Goldberg [34].)

Lemma 3.9 *Let G be a fixed distribution function with the properties that $G(0) = 0$ and that $G'(0)$ exists. Then there are constants $S = S_G$ and ε_G such that for any collection $X_1, \ldots, X_m$ of independent random variables, all distributed according to G, and for any $\varepsilon < \varepsilon_G$,*

$$\mathbf{Pr}(X_1 + \cdots + X_m \leq \varepsilon) \;\leq\; \left(\frac{eS\varepsilon}{m}\right)^m \;. \tag{3.18}$$

Proof. By our assumptions on G, $G(\varepsilon) = G'(0) \cdot \varepsilon + o(\varepsilon)$, as $\varepsilon \downarrow 0$, and there exist constants $S = S_G > G'(0)$ and $\varepsilon_G \leq 1/S$ so that, for all $\varepsilon < \varepsilon_G$, $G(\varepsilon) \leq S \cdot \varepsilon$. Let X be a random variable that is distributed according to G, and let Y be a random variable that is uniformly distributed on $[0, 1/S]$. For all $\varepsilon < \varepsilon_G$, we have by construction,

$$\mathbf{Pr}(X \leq \varepsilon) \;=\; G(\varepsilon) \;\leq\; S \cdot \varepsilon \;=\; \mathbf{Pr}(Y \leq \varepsilon) \;,$$

which implies that

$$\mathbf{Pr}\Big(\min\{X, \varepsilon_G\} \leq \zeta\Big) \;\leq\; \mathbf{Pr}\Big(\min\{Y, \varepsilon_G\} \leq \zeta\Big) \quad \text{for all } \zeta \geq 0 \;.$$

This means that the random variable $\min\{Y, \varepsilon_G\}$ is stochastically dominated by the random variable $\min\{X, \varepsilon_G\}$. Let $Y_1, \ldots, Y_m$ be independent copies of Y. Since stochastic dominance is preserved under taking sums of independent random variables (see Proposition 2.2), for any $\varepsilon < \varepsilon_G$, we get the following bound on $\mathbf{Pr}(X_1 + \cdots + X_m \leq \varepsilon)$,

$$\begin{aligned} \mathbf{Pr}(X_1 + \cdots + X_m \leq \varepsilon) &= \mathbf{Pr}\Big(\min\{X_1, \varepsilon_G\} + \cdots + \min\{X_m, \varepsilon_G\} \leq \varepsilon\Big) \\ &\leq \mathbf{Pr}\Big(\min\{Y_1, \varepsilon_G\} + \cdots + \min\{Y_m, \varepsilon_G\} \leq \varepsilon\Big) \\ &= \mathbf{Pr}(Y_1 + \cdots + Y_m \leq \varepsilon) \;. \end{aligned} \tag{3.19}$$

It is easily seen that $\mathbf{Pr}(Y_1 + \cdots + Y_m \leq \varepsilon) = (S\varepsilon)^m/m!$, for any $\varepsilon < \varepsilon_G \leq 1/S$, since the probability in question equals the volume of a suitably scaled standard simplex in $\mathbb{R}^m$. We thus get from (3.19) that

$$\mathbf{Pr}(X_1 + \cdots + X_m \leq \varepsilon) \;\leq\; \frac{(S\varepsilon)^m}{m!} \;\leq\; \left(\frac{eS\varepsilon}{m}\right)^m \;,$$

for any $\varepsilon < \varepsilon_G$, as desired. □

Lemma 3.10 *Suppose networks (D_n, r) are generated according to the extended uniform model with parameters F and p, where F satisfies assumption (A1). If $np/\log n$ is sufficiently large, then shortest paths in (D_n, r) consist of $O(\log n)$ arcs with high probability.*

Proof. For this proof, it will be convenient to view the extended uniform model as we described it in the introductory paragraph of this chapter. We determine the structure of the graph in a first experiment according to the $\mathscr{D}(n,p)$ model. In a second experiment, costs of arcs that were determined as present in the first experiment are drawn at random, independently of each other, according to F. For an arbitrary but fixed $\gamma \geq 1$, let a threshold value ε_s be defined as $7(\gamma + 3.45)/F'(0) \cdot (\log n)/(np) =: C(\log n)/(np)$. (This value is motivated by the proof of Lemma 3.6.) If $np/\log n$ is sufficiently large, then $\varepsilon_s < \varepsilon_0$, with ε_0 as defined before (3.4). Let us call a path P in (D_n, r) *short* if $r(P) = \sum_{(v,w)\in P} r(v,w) \leq \varepsilon_s$. For a given (directed) path consisting of a fixed number of m arcs that are all present according to the first, structural experiment, the cost of this path is the value of a sum of m independent, identically distributed random variables, whose common distribution function F satisfies the assumptions of Lemma 3.9 because of assumption (A1). More precisely, we can apply Lemma 3.9 with $G = F$, $S = 1.1$, und $\varepsilon_F = \varepsilon_0$. Lemma 3.9 implies that a given path with m arcs is short with probability at most $(eS\varepsilon_s/m)^m = ((eSC\log n)/(npm))^m$. This upper bound is at most $(enp)^{-m}$ if $m \geq m_0 := M\log n$, for some $M \geq e^2SC$. Hence, taking the first, structural experiment also into account, short paths with more than m_0 arcs exist in (D_n, r) only with probability at most $\sum_{m\geq m_0}(m+1)!\binom{n}{m+1} \cdot p^m \cdot (enp)^{-m} \leq n^2 \cdot n^{-M}$. If we also ensure that $M \geq \gamma+2$, this probability is $O(n^{-\gamma})$. It follows from Lemma 3.4 and the considerations above that with probability at least $1 - O(n^{-\gamma})$, the diameter Δ in (D_n, r) does not exceed the threshold value ε_s *and* all short paths consist of $O(\log n)$ arcs. If this is the case, shortest paths in (D_n, r) (with respect to r) are also short (with respect to the definition above), which proves the lemma. □

3.5 Related work

In the uniform model, that is, for the case $p = 1$, the bounds on the quantities with which we were concerned in Lemmata 3.4 and 3.10 are essentially tight, as is indicated by the following results. Davis and Prieditis [17] showed that for r distributed exponentially (with parameter 1), the expected cost of a shortest path is of order $(\log n)/n + O(1/n)$. This is also true for r uniformly distributed; see [17, 57]. In fact, this result holds if the distribution of r is chosen from a fairly general class of distributions, including both exponential and uniform distributions, as was proved (for undirected graphs) by Janson [45] recently. Janson also studied the number of arcs on shortest paths, proving (among other results) that for ν_s the maximum number of arcs on shortest paths

from a given vertex s, $\nu_s / \log n$ converges in probability to e. A similar result is conjectured in [45] for the maximum number of arcs on any shortest path. The proof techniques of Davis and Prieditis or Janson seem to break down as soon as the underlying graph is not complete.

In the variant of the uniform model in which arc costs are assumed to be positive *integers*, the distribution of distances and the number of arcs on shortest paths were studied by Walley, Tan, and Viterbi [86, 87]. We admit that we found it difficult to distill the essence of their results from their papers. We note, however, that for the case when the arc costs of networks on n vertices are drawn uniformly at random from the set $\{1, \ldots, n\}$, the evolution of distance levels is very closely related to the processes mentioned in Remark 3.3. A result related to Lemma 3.10 (but somewhat weaker) appeared in a recent paper by Subramanian [79]. He considered the variant of the extended uniform model in which arc costs for instances on n vertices are integers drawn uniformly at random from the set $\{0, \ldots, n-1\}$. Subramanian proved that if $p \geq 19(\log_2 n)/n$, then there exists with high probability between any pair of vertices a path of cost $O((\log_2 n)/p)$ that consists of exactly $\lfloor \log_2 n \rfloor$ arcs. Note that this statement leaves open whether a shortest path consists of fewer arcs. The proof is based on elaborate counting arguments and thus works only for integer costs. It uses a generalization of the Janson inequalities (see, for example, [5]), exploiting that the costs of most paths exhibit only a limited dependence (depending on the number of arcs that they share).

Chapter 4

The vertex-potential model

In this chapter, we address the problem of analyzing the average-case complexity of shortest-paths problems on directed graphs D with *arbitrary real* arc costs c. As already mentioned in the introduction, it is not obvious how to define an input model that allows negative arc costs but does not trivialize the problem by overemphasizing instances with negative cycles; see our discussion on pages 6–7. We study the so-baptized vertex-potential model, which generates instances of shortest-paths problems with arbitrary real arc costs but without negative cycles. This model was previously used by Cherkassky, Goldberg, and Radzik [11] in an experimental evaluation of shortest-paths algorithms. We show in Section 4.3 that if networks (D_n, c) are generated according to the vertex-potential model, then the single-source shortest-paths problem can be solved in $O(n^2)$ expected time, and the all-pairs shortest-paths problem can be solved in $O(n^2 \log n)$ expected time. In both cases our algorithms are reliable, that is, finish their computations within the respective time bounds with high probability. The algorithms are probabilistic, that is, tailored to networks generated according to the vertex-potential model. The results presented in this chapter are the joint work of Colin Cooper, Alan Frieze, Kurt Mehlhorn, and myself [13, 14].

4.1 Absence of negative cycles in (D, c) and reduced arc costs

If an algorithm is supposed to solve the single-source shortest-paths problem with source s on (D, c), the correctness of its output can be checked by the following *correctness conditions*; see, for example, [1, Section 5.2] for a proof.

Proposition 4.1 (Correctness conditions) *For every vertex $v \in V$, let $d_c(v)$ denote the cost (with respect to c) of some directed path from s to v, with $d_c(s) = 0$. The $d_c(v)$'s are equal to the*

distances $\delta_c(s,v)$, $v \in V$, if and only if they satisfy

$$c(v,w) + d_c(v) - d_c(w) \;\geq\; 0 \;\text{ , for all arcs } (v,w) \;. \tag{4.1}$$

Suppose that on (D,c), a *(vertex) potential* $\pi(v)$ is given for each vertex v. In the vein of Proposition 4.1, we define, for each arc (v,w), its *reduced (arc) cost* with respect to the $\pi(v)$'s by $c(v,w) + \pi(v) - \pi(w)$. The following proposition characterizes arc costs c for which (D,c) does not contain any negative cycles. In fact, Proposition 4.2 is hardly more than a reformulation of Proposition 4.1, since in a strongly connected directed graph D with arc costs c, finite real shortest-path distances $d_c(v)$, $v \in V$, exist if and only if (D,c) does not contain any negative cycles, and shortest-paths distances $d_c(v)$, $v \in V$, satisfy (4.1).

Proposition 4.2 *The absence of negative cycles in a strongly connected network (D,c) is equivalent to the existence of finite real vertex potentials $\pi(v)$, $v \in V$, so that the reduced arc costs r (of c with respect to the $\pi(v)$'s) are non-negative, that is,*

$$r(v,w) \;:=\; c(v,w) + \pi(v) - \pi(w) \;\geq\; 0 \;\text{ , for all arcs } (v,w) \;.$$

Shortest-paths problems are invariant under reduction of arc costs with respect to vertex potentials. This well-known fact is a key ingredient of our analyses of shortest-paths problems in the vertex-potential model. We summarize this knowledge in the following proposition.

Proposition 4.3 *Suppose that we associate a vertex potential $\hat{\pi}(v) \in \mathbb{R}$ with each vertex $v \in V$ and that we define reduced arc costs $\hat{c}$ (of c with respect to the $\hat{\pi}(v)$'s) by $\hat{c}(v,w) = c(v,w) + \hat{\pi}(v) - \hat{\pi}(w)$, for each arc (v,w). Then, for any directed path P from vertex s to vertex t,*

$$\hat{c}(P) \;=\; \sum_{(v,w)\in P} (c(v,w) + \hat{\pi}(v) - \hat{\pi}(w)) \;=\; c(P) + \hat{\pi}(s) - \hat{\pi}(t) \;. \tag{4.2}$$

Therefore, distances with respect to c and $\hat{c}$ relate to each other through $\delta_{\hat{c}}(s,t) = \delta_c(s,t) + \hat{\pi}(s) - \hat{\pi}(t)$, for all $s,t \in V$. It also follows from (4.2) that, for any directed cycle C, $\sum_{(v,w)\in C} \hat{c}(v,w) = \sum_{(v,w)\in C} c(v,w)$; in particular, $(D,\hat{c})$ contains a negative cycle if and only if there is a negative cycle in (D,c).

4.2 The vertex-potential model

The *vertex-potential model* is motivated by the propositions in the previous section. In the vertex-potential model, arc costs $c(v,w)$ are generated according to

$$c(v,w) \;=\; r(v,w) - \pi(v) + \pi(w) \;\text{ , for all arcs } (v,w) \;, \tag{4.3}$$

with random quantities $r(v,w) \geq 0$ and arbitrary vertex potentials $\pi(v)$, $v, w \in V$. We set $r(v,v) \equiv 0$, for $v \in V$. Instances generated according to the vertex-potential model thus have real, possibly negative arc costs. However, it follows from the definition in (4.3), from the assumption $r(v,w) \geq 0$, for $v, w \in V$, and from Proposition 4.3 that these instances do not contain any negative cycles. Of course, only the $c(v,w)$'s are revealed to our algorithms, and the $r(v,w)$'s and $\pi(v)$'s are hidden parameters of the vertex-potential model. Our precise assumptions for these variables are as follows:

(A1) The quantities $r(v,w)$, $v, w \in V$, $v \neq w$, are drawn at random, independently of each other, according to a common distribution function F. We assume that $F(0) = 0$, $F(1) = 1$, and that $F'(0)$ exists and is strictly positive.

(A2) The vertex potentials $\pi(v)$, $v \in V$, are arbitrary real numbers.

The reader might be worried about our use of label (A1) for both the first assumption on the parameters in vertex-potential model, as defined above, and the assumption on the distribution function in the uniform model, as defined in Section 3.2.2. However, the use of the same label is fully justified and should not lead to any confusion, since assumption (A1) above states nothing more than that the (hidden) arc costs r in the vertex-potential model are indeed generated according to the uniform model. Put differently, the uniform model is contained in the vertex-potential model as the special case $\pi \equiv 0$. The assumption $F(1) = 1$, which we added in this section to (A1), is not restrictive. Lemma 4.5 is the only place where we use the assumption that the random variables are bounded, and the assumption is there more for convenience than necessity. It could be replaced by bounds on the tails of the distributions. It should be clear from our analyses in Chapter 3 that the independence assumptions in (A1) are the most important (and restrictive) ones.

4.3 Shortest-paths algorithms in the vertex-potential model

In Section 3.1, we discussed a probabilistic algorithm for solving shortest-paths problems on networks that are generated according to the *uniform model*. The algorithm is based on the pruning idea presented in Section 2.1.1. This idea could also be applied to networks (D, c) generated according to the vertex-potential model. At first sight, however, it seems that because of the completely arbitrary vertex potentials, we have lost any control over the size of the diameter Δ_c, and thus no effective cut-off value can be derived. This is indeed true for arc costs c, but we can effectively exploit the fact that shortest paths are invariant under reduction of arc costs with respect to vertex potentials; see Proposition 4.3.

More precisely, our algorithm for the single-source shortest-paths problem computes vertex potentials $\hat{\pi}(v)$, $v \in V$, with the property that for any $v, w \in V$, the difference $\hat{\pi}(v) - \hat{\pi}(w)$ is close to

$\pi(v) - \pi(w)$. The $\hat{\pi}(v)$'s are then used to define reduced arc costs $\hat{c}$ by

$$\hat{c}(v,w) \;:=\; c(v,w) + \hat{\pi}(v) - \hat{\pi}(w) \;=\; r(v,w) + (\hat{\pi}(v) - \pi(v)) - (\hat{\pi}(w) - \pi(w)) \;, \text{ for all arcs } (v,w) \;.$$

The shortest-paths problems (D_n, c) and $(D_n, \hat{c})$ are equivalent. However, $(D_n, \hat{c})$ is more efficiently solvable, since arc costs $\hat{c}$ allow a substantial pruning of the arc set. If (v,w) is contained in a shortest path (with respect to $\hat{c}$), then, by Bellman's principle, $\hat{c}(v,w) = \delta_{\hat{c}}(v,w) \leq \Delta_{\hat{c}}$. Note that arc costs $\hat{c}$ can also be interpreted as arc costs r reduced with respect to vertex potentials $(\hat{\pi}(v) - \pi(v))$, for $v \in V$. It thus follows from Proposition 4.3 that for any two vertices u, t, $\delta_{\hat{c}}(u,t) = \delta_r(u,t) + (\hat{\pi}(u) - \pi(u)) - (\hat{\pi}(t) - \pi(t))$, which implies

$$\Delta_{\hat{c}} \;\leq\; \Delta_r + \max_{u,t} |(\hat{\pi}(u) - \pi(u)) - (\hat{\pi}(t) - \pi(t))| \;. \tag{4.4}$$

We know a tight bound on Δ_r from Lemma 3.4, and we prove one for the second term on the right-hand side of (4.4) in Lemma 4.5. As a consequence, we can prune away from instances of size n all but $O(n^{3/2}(\log n)^{1/2})$ arcs with high probability without changing the shortest-path distances; see the discussion in Section 4.3.2.

The reduced arc costs $\hat{c}$ are still not necessarily non-negative. The single-source shortest-paths problem is thus solved by running the Bellman–Ford algorithm [7, 26] on the pruned network. The running time of the algorithm depends on the maximum number ν of arcs on a shortest path. On instances of size n, ν is of order n in the worst case. Recall that on instances generated according to the *uniform model*, ν is $O(\log n)$ with high probability, though, as we proved in Lemma 3.10. We exploit once again the fact that shortest paths are invariant under reduction of arc costs with respect to vertex potentials: the following is an immediate consequence of Lemma 3.10.

Corollary 4.4 *Let arc costs r be distributed as specified in (A1), and let arc costs $\hat{c}$ be obtained by reducing the arc costs r with respect to some vertex potentials. Shortest paths in $(D_n, \hat{c})$ consist then of $O(\log n)$ arcs with high probability.*

The Bellman–Ford algorithm thus computes the correct distances for the single-source shortest-paths problem in $O(n^{3/2}(\log n)^{3/2}) = o(n^2)$ time with high probability on the pruned graph. This is dominated by the time spent on pruning the arc set and on checking the computed distances, which are responsible for the $O(n^2)$ running time of our algorithm; see Theorem 4.7. Given a solution of one single-source shortest-paths problem, we transform a problem with arbitrary real arc costs into an equivalent problem with non-negative arc costs and apply one of the algorithms of McGeoch [57] and Karger, Koller, and Phillips [49], which finish their computations in $O(n^2 \log n)$ time with high probability; see Theorem 4.9.

We now describe our probabilistic algorithms in more detail. For instances of size n, we denote their vertex sets by $V(n)$.

4.3.1 Approximating the vertex-potential differences

We first show how to compute vertex potentials $\hat{\pi}(v)$, $v \in V(n)$, so that, for any $u, t \in V(n)$, with high probability, the difference $\hat{\pi}(u) - \hat{\pi}(t)$ is a good approximation of the actual vertex-potential difference $\pi(u) - \pi(t)$.

Lemma 4.5 *Let arc costs c be generated according to the vertex-potential model. For any $v \in V(n)$, define*

$$\hat{\pi}(v) \;:=\; -\frac{1}{n} \cdot \sum_{w \in V(n)} c(v,w) \;. \tag{4.5}$$

Then, for any $u, t \in V(n)$, the term $|(\hat{\pi}(u) - \hat{\pi}(t)) - (\pi(u) - \pi(t))|$ is of order $O(\sqrt{(\log n)/n})$ with high probability.

Proof. For any $u, w \in V(n)$, we have $-c(u,w) = \pi(u) - r(u,w) - \pi(w)$ by the definition in (4.3). Thus, if $\sum_w$ denotes summation over all vertices $w \in V(n)$,

$$-\textstyle\sum_w c(u,w) \;=\; n \cdot \pi(u) - \sum_w r(u,w) - \sum_w \pi(w) \;,$$

from which we conclude with (4.5) that

$$\hat{\pi}(u) - \pi(u) \;=\; -\tfrac{1}{n} \cdot \textstyle\sum_w r(u,w) - \frac{1}{n} \cdot \sum_w \pi(w) \;.$$

Note that the rightmost term, $\frac{1}{n} \cdot \sum_w \pi(w)$, is independent of u. Hence, for any $u, t \in V(n)$,

$$|(\hat{\pi}(u) - \pi(u)) - (\hat{\pi}(t) - \pi(t))| \;\le\; \left| \tfrac{1}{n-1} \cdot (\textstyle\sum_w r(u,w) - \sum_w r(t,w)) \right| \;. \tag{4.6}$$

By the assumptions on the vertex-potential model, the sums $\sum_w r(\cdot, w)$ on the right-hand side of (4.6) are determined independently as values of $\sum_{i=1}^{n-1} R_i$, where the R_i's are independent copies of a random variable R distributed according to F as in (A1). Let $\rho = \mathbf{E}[R]$; for any arbitrary but fixed constant γ, the Chernoff–Hoeffding bounds (2.9) and (2.11) imply that, for sufficiently large n,

$$\mathbf{Pr}\left(\left| \tfrac{1}{n-1} \cdot \textstyle\sum_{i=1}^{n-1} R_i - \rho \right| \ge \sqrt{3(\gamma+1) \cdot (\log n)/((n-1)\rho)} \cdot \rho \right) \;\le\; 2 \cdot e^{-(\gamma+1)\cdot \log n} \;=\; O(n^{-(\gamma+1)}) \;. \tag{4.7}$$

(Observe that we normalized the sum on the left-hand side.) By applying the triangle inequality to (4.6), we get, for any $u, t \in V(n)$,

$$|(\hat{\pi}(u) - \pi(u)) - (\hat{\pi}(t) - \pi(t))| \;\le\; 2 \cdot \max\nolimits_{v \in V(n)} \left| \tfrac{1}{n-1} \cdot \textstyle\sum_w r(v,w) - \rho \right| \;.$$

We conclude from (4.7) that $2 \cdot \max_{v \in V(n)} |\frac{1}{n-1} \cdot \sum_w r(v,w) - \rho|$ and thus $|(\hat{\pi}(u) - \pi(u)) - (\hat{\pi}(t) - \pi(t))|$, for any $u, t \in V(n)$, are all $O(\sqrt{(\log n)/n})$ with high probability. □

Remark 4.6 In [13], we assumed the $\pi(v)$'s to be independent, identically distributed random variables with values in $[-1,1]$. Using these stronger assumptions on the $\pi(v)$'s, one can prove that $|\hat{\pi}(v) - \pi(v)| = O(\sqrt{(\log n)/n})$ with high probability, if the approximate vertex potentials $\hat{\pi}(v)$ are defined as $\hat{\rho} - \frac{1}{n-1} \cdot \sum_w c(v,w)$, for $v \in V(n)$, with $\hat{\rho} = \frac{1}{n(n-1)} \cdot \sum_{v,w} c(v,w) = \frac{1}{n(n-1)} \cdot \sum_{v,w} r(v,w)$. (The "observed mean" $\hat{\rho}$ is a good approximation of ρ.) It turns out, however, that the approximation of single vertex potentials is not needed in the analysis of our algorithms, and that the generalized assumption (A2) suffices.

4.3.2 Solving shortest-paths problems in the vertex-potential model

We are now ready to explain in detail how we solve shortest-paths problems on a network (D_n, c) if arc costs in (D_n, c) are generated according to the vertex-potential model. Our algorithm proceeds in three phases, a preprocessing phase, a computation phase, and a postprocessing phase, in which the correctness of the solution from the second phase is checked. Let the source vertex for the single-source shortest-paths problem under consideration be denoted by s.

Preprocessing. The algorithm computes, for every vertex $v \in V(n)$, a vertex potential $\hat{\pi}(v)$ as in Lemma 4.5 and transforms the arc costs $c(v,w)$ to

$$\hat{c}(v,w) \;:=\; c(v,w) + \hat{\pi}(v) - \hat{\pi}(w) \;=\; r(v,w) + (\hat{\pi}(v) - \pi(v)) - (\hat{\pi}(w) - \pi(w)) \;, \text{ for all arcs } (v,w) \;. \tag{4.8}$$

Recall from (4.4) that

$$\Delta_{\hat{c}} \;\leq\; \Delta_r + \max_{u,t} |(\hat{\pi}(u) - \pi(u)) - (\hat{\pi}(t) - \pi(t))| \;.$$

For $\gamma \geq 1$ arbitrary but fixed, we know from Lemmata 3.4 and 4.5 that constants C_γ and M_γ exist so that

$$\Delta_r \;\leq\; C_\gamma (\log n)/n \quad \text{and} \quad \max_{u,t} |(\hat{\pi}(u) - \pi(u)) - (\hat{\pi}(t) - \pi(t))| \;\leq\; M_\gamma \sqrt{(\log n)/n} \tag{4.9}$$

with probability at least $1 - O(n^{-\gamma})$. For the time being, we assume that (4.9) holds. For an arbitrary (but fixed) constant $L_\gamma > M_\gamma$ and sufficiently large n, all *significant* arcs (that is, arcs that are possibly contained in a shortest path) are then contained in

$$\hat{A} := \left\{ (v,w) \,;\, \hat{c}(v,w) \leq L_\gamma \sqrt{(\log n)/n} \right\} \;.$$

(It follows from the proof of Lemma 4.5 that we could set $L_\gamma = 5\sqrt{\gamma}$. This value is not optimal but nevertheless indicates that, for given γ, some explicit constant L_γ is easily derivable.) The vertex potentials $\hat{\pi}(v)$, $v \in V(n)$, the reduced arc costs $\hat{c}$, and the set $\hat{A}$ can be computed in $O(n^2)$ time. Let $\hat{D}_n$ denote the graph $(V(n), \hat{A})$.

Computation. We now solve a single-source shortest-paths problem with source s on the pruned graph $(\hat{D}_n, \hat{c})$ by running the Bellman–Ford algorithm [7, 26]. This algorithm maintains tentative distances $d(v)$ for every vertex v. The $d(v)$'s are initially set to ∞ (except for $d(s) = 0$), and $d(v)$ always represents the cost of some path in $(\hat{D}, \hat{c})$ from s to v. The Bellman–Ford algorithm proceeds in passes over the arc set $\hat{A}$, maintaining the following invariant. After the k-th pass, the Bellman–Ford algorithm has correctly computed the distances of all vertices to which there is a shortest path from s consisting of at most k arcs. The algorithm actually checks the correctness conditions (4.1) for all arcs (in $\hat{A}$) in each pass, and it therefore terminates (with all distances in $(\hat{D}_n, \hat{c})$ computed correctly) after the ν-th pass, where ν is the maximum number of arcs on a shortest path in $(\hat{D}_n, \hat{c})$. The running time of the Bellman–Ford algorithm is therefore $O(\nu \cdot |\hat{A}|)$, which is $O(n^3)$ in the worst case but $o(n^2)$ with high probability, as we now argue.

By (4.8) and (4.9),

$$\hat{A} \subseteq \left\{(v, w)\,;\, r(v, w) \leq (L_\gamma + M_\gamma)\sqrt{(\log n)/n}\right\} \quad .$$

Since we assume that the distribution function of the arc costs can be approximated in a neighborhood of 0 by a uniform distribution (see (A1) and (3.3)), any arc (v, w) is an element of the set on the right-hand side with probability $\hat{p} = \Theta(\sqrt{(\log n)/n})$ for sufficiently large n, independently of all the other arcs. The random variable $|\hat{A}|$, the cardinality of $\hat{A}$, is therefore stochastically dominated by a random variable that is binomially distributed with parameters $n(n-1)$ and $\hat{p}$. We apply the tail estimate (2.12) to deduce that $|\hat{A}| = O(n(n-1)\sqrt{(\log n)/n}) = O(n^{3/2}\sqrt{\log n})$ with high probability. As we have argued in Corollary 4.4, $\nu = O(\log n)$ with high probability. Hence, with high probability, it takes $O(n^{3/2}(\log n)^{3/2}) = o(n^2)$ time to run the Bellman–Ford algorithm on the pruned graph $(\hat{D}, \hat{c})$.

Postprocessing. The second phase has failed to compute all distances correctly only if (4.9) does not hold, which happens only with probability $O(n^{-\gamma})$. The correctness conditions (4.1) (checked for all arcs) are an $O(n^2)$-time certificate for the correctness of the solution. Since the worst-case running time of the Bellman–Ford algorithm on $(D_n, \hat{c})$ is $O(n^3)$, we can easily afford to run the Bellman–Ford algorithm on $(D_n, \hat{c})$ in case of failure, without affecting the bounds on the running time. Finally, it takes $O(n^2)$ time to compute the distances $\delta_c(v) = \delta_{\hat{c}}(v) - \hat{\pi}(s) + \hat{\pi}(v)$, for all $v \in V(n)$.

The discussion above is summarized in the following theorem.

Theorem 4.7 *Assume that the arc costs in (D_n, c) are generated according to the vertex-potential model. The single-source shortest-paths problem can then be solved in time $O(n^2)$ with high probability.*

Remark 4.8 Theorem 4.7 still holds if the Bellman–Ford algorithm takes time $O(n^2)$ on $(\hat{D}, \hat{c})$ (with high probability). This means that because of Corollary 4.4, we could afford to restrict the Bellman–Ford algorithm to the arc set $\{(v,w)\,;\,\hat{c}(v,w) \le R/\log n\}$, which is a set of cardinality $O(n^2/\log n)$ with high probability for a suitable constant R. In fact, even if we omitted the computation of both vertex potentials and reduced arc costs and just ran the Bellman–Ford algorithm on the complete input graph, we would solve the single-source shortest-paths problem in time $O(n^2 \log n)$ with high probability because of Corollary 4.4.

Theorem 4.9 *Assume that the arc costs in (D_n, c) are generated according to the vertex-potential model of Section 4.2. The all-pairs shortest-paths problem can then be solved in $O(n^2 \log n)$ with high probability.*

Proof. We first compute distances $\delta_c(v)$, $v \in V(n)$, with respect to an arbitrary source vertex s by solving a single-source shortest-paths problem as in the proof of Theorem 4.7 or as indicated in Remark 4.8. With high probability, this takes time $O(n^2)$ or $O(n^2 \log n)$, respectively. Let $\widetilde{c}$ be the reduced arc costs of c with respect to the vertex potentials $\delta_c(v)$, $v \in V(n)$, that is, for all arcs (v,w),

$$\begin{aligned} \widetilde{c}(v,w) &= c(v,w) + \delta_c(v) - \delta_c(w) \\ &= r(v,w) + (\delta_c(v) - \pi(v)) - (\delta_c(w) - \pi(w)) \ . \end{aligned} \tag{4.10}$$

It follows from the first equality and the correctness conditions (4.1) that $\widetilde{c}(v,w) \ge 0$ for all arcs (v,w). The reduced arc costs $\widetilde{c}$ can be computed in $O(n^2)$ time, and the same time bound later allows us to transform distances $\delta_{\widetilde{c}}(v,w)$ into distances $\delta_c(v,w)$, for all $v, w \in V(n)$.

To compute the $\delta_{\widetilde{c}}(v,w)$'s, we run one of the algorithms of Karger, Koller, and Phillips [49] or McGeoch [57] on $(D_n, \widetilde{c})$, which efficiently solve the all-pairs shortest-paths problem with non-negative arc costs. Both algorithms run in time $O(n^2 \log n + n|H|)$ where $H = H(\widetilde{c})$ is the set of essential arcs, that is, of arcs that are a shortest path (with respect to $\widetilde{c}$) between their endpoints. We apply the arguments of Section 4.1 again. Shortest paths are invariant under reduction of the arc costs with respect to vertex potentials, and it follows from (4.10) that the arcs in H are also a shortest path between their endpoints with respect to arc costs r. The set H is therefore contained in the set $\{(v,w)\,;\,r(v,w) \le \Delta_r\}$, and it follows by Lemma 3.6 that $|H| = O(n \log n)$ with high probability. This yields a running time of $O(n^2 \log n)$ with high probability for the algorithms of McGeoch and Karger, Koller, and Phillips. Since $|H| = O(n^2)$ in the worst case, our algorithm has a running time of $O(n^3)$ with probability $O(n^{-\gamma})$, $\gamma \ge 1$, which still gives an expected running time of $O(n^2 \log n)$. □

Chapter 5

The endpoint-independent model

In this chapter, we analyze the average-case complexity of shortest-paths algorithms with respect to the endpoint-independent model. The graph underlying any instance is the complete directed graph with loops. We are interested in networks whose adjacency lists have been sorted with respect to increasing arc costs. To generate such a network according to the endpoint-independent model, we randomly fix, for each vertex $v \in V$ independently of the other vertices, an order of the endpoints of the arcs as they appear in the sorted adjacency list of vertex v. This corresponds to drawing a permutation σ_v of V uniformly at random for each $v \in V$. The actual arc costs are then arbitrarily fixed as lists of non-negative arc costs in increasing order. How does the endpoint-independent model generalize the uniform model? This becomes particularly clear if we compare the definition of the endpoint-independent model given above to the two-round view of the uniform model that we described in Section 3.2.1.

Since the arc costs may be arbitrarily fixed in the endpoint-independent model, it does not allow us to deduce anything on the distribution of arc costs. In particular, it is impossible to devise a probabilistic algorithm in the spirit of Section 3.1. It was first noticed by Bloniarz [8] that, nevertheless, the average-case behavior of algorithms for solving shortest-paths problems can be analyzed when instances are generated according to the endpoint-independent model. Noshita [68] (and later Goldberg and Tarjan [35]) analyzed the average-case complexity of Dijkstra's algorithm in a restricted version of the endpoint-independent model; the time bound, however, does not improve over the worst-case complexity of the best known implementations of the algorithm. Spira [78] dealt first with the average-case complexity of the all-pairs shortest-paths problem. He proved an expected time bound of $O(n^2(\log n)^2)$ on instances of size n, which was later improved by Bloniarz [8] and Frieze and Grimmett [31]. In [64] and [65], Moffat and Takaoka describe two algorithms with an expected time bound of $O(n^2 \log n)$. The algorithm in [65] is a simplified version of that of [64].

We review the algorithm in [65] and the analysis of its expected running time. We point out some mistakes in the analysis and show how to avoid them. Moreover, we show that the algorithm of Moffat and Takaoka is reliable, that is, it runs in time $O(n^2 \log n)$ with high probability and not just in expectation. These results represent joint work with Kurt Mehlhorn [60].

5.1 Experiments related to the endpoint-independent model

We refer to the following random-sampling experiments. Let an urn contain n balls that are either red or blue; let b be the number of blue balls. The balls are repeatedly drawn from the urn (without replacement) uniformly at random. Let W be the number of drawings until the first blue ball occurs; then

$$\mathbf{E}[W] \;=\; \textstyle\sum_{0\le k\le n-b} \mathbf{Pr}(W > k) \;=\; \sum_{0\le k\le n-b} \binom{n-b}{k}/\binom{n}{k} \;=\; \sum_{0\le k\le n-b} \binom{n-k}{b}/\binom{n}{b} \;,$$

and from $\binom{n-k}{b} = \binom{n+1-k}{b+1} - \binom{n-k}{b+1}$, we conclude that

$$\mathbf{E}[W] \;=\; \binom{n+1}{b+1}/\binom{n}{b} \;=\; \tfrac{n+1}{b+1} \;.$$

We are also interested in sampling with replacement. Suppose that in a sequence of independent trials, the probability of success is p for each of the trials. Let Z be the number of trials until the first successful one; then

$$\mathbf{E}[Z] \;=\; \textstyle\sum_{k\ge 0} \mathbf{Pr}(Z > k) \;=\; \sum_{k\ge 0}(1-p)^k \;=\; 1/p \;. \tag{5.1}$$

In the so-called *coupon-collector problem*, we are given a set of n distinct coupons and we try to complete a collection of all coupons. In each trial, a coupon is drawn (with replacement) uniformly and independently at random. We call a trial a success if it results in adding a new coupon to our collection. Let Z^* denote the completion time, that is, the number of trials required to see at least one copy of each coupon. We can write Z^* as $Z^* = 1 + Z_1 + \cdots + Z_{n-1}$, where for $1 \le i < n$, the random variable Z_i is the number of trials (with probability of success $\frac{n-i}{n}$ each) between the i-th and $(i+1)$-th success (excluding the former, including the latter). By the argument in (5.1), $\mathbf{E}[Z_i] = \frac{n}{n-i}$, and hence, $\mathbf{E}[Z^*] = \sum_{0\le i<n} \frac{n}{n-i} \le n(\log n + 1) = O(n \log n)$.

In the coupon-collector problem (with n coupons), the probability that a particular coupon has not been collected after t trials equals $(1 - \frac{1}{n})^t$. Hence, for any $\beta > 1$,

$$\mathbf{Pr}(Z^* > \beta n \log n) \;\le\; n\left(1 - \tfrac{1}{n}\right)^{\beta n \log n} \;\le\; n e^{-\beta \log n} \;=\; n^{-(\beta-1)} \;; \tag{5.2}$$

that is, the completion time Z^* in the coupon-collector problem (with n coupons) is $O(n \log n)$ with high probability. (In fact, one can prove even stronger sharp-threshold results on the variable Z^*; see, for example, [66, Section 3.6].)

In our analyses, we often (and sometimes only implicitly) refer to the following special case of the *principle of deferred decisions* [66, p. 55]. Suppose that from a permutation σ of V that was drawn uniformly at random, it is only revealed that σ maps $V' \subseteq V$ to $\sigma(V') \subseteq V$. Then the yet unknown part of σ, that is, the restriction of σ to the domain $V - V'$, is still a random mapping of $V - V'$ to $V - \sigma(V')$. In particular, we may interpret the process of increasing the set V' one by one as fixing the permutation σ by $|V|$ independent random experiments.

5.2 The algorithm of Moffat and Takaoka

For the sake of future reference, we briefly review the algorithm of Moffat and Takaoka in [65]; its main ideas can actually be traced back to work of Dantzig [16]. We are given a complete network (D_n, c) with non-negative arc costs c. The algorithm first sorts all adjacency lists in order of increasing costs (with ties resolved randomly, total time $O(n^2 \log n)$) and then solves n single-source shortest-paths problems, one for each vertex. A single-source shortest-paths problem with source $s \in V$ is solved by *labeling* the vertices in order of increasing distance from the source. If v is a labeled vertex, then its exact distance $\delta(v)$ from the source is known. We use S to denote the set of labeled vertices and $U = V - S$ to denote the set of unlabeled vertices. Initially, only the source vertex is labeled, that is, $S = \{s\}$ with $\delta(s) = 0$. For each labeled vertex v, one of its outgoing arcs is called its *current arc* and is denoted $ca(v)$; we maintain the invariant that all arcs preceding the current arc $ca(v)$ in v's (sorted) adjacency list have their endpoint already labeled. We say that the arcs preceding $ca(v)$ (as well as their endpoints) have been *scanned* by the algorithm. The *potential* of v's current arc is defined as $\delta(v) + c(ca(v))$. The algorithm proceeds in *iterations*. In each iteration, the algorithm selects the current arc of minimum potential; suppose that $ca(v)$ is selected and that w is its endpoint. If w is not yet labeled, then the algorithm labels w (that is, adds w to S) and sets $\delta(w)$ to $\delta(v) + c(ca(v))$. (It follows by a standard argument as for Dijkstra's algorithm that this indeed sets $\delta(w)$ to the distance of w from s.) Moreover, some current-arc pointers are updated. The precise nature of these updates depends on the size of U.

As long as $|U| > n/\log n$, the algorithm is said to be in *Phase I*, and the additional invariant is maintained that the endpoints of all current arcs are unlabeled. Whenever the algorithm selects a current arc $ca(v)$ of minimum potential, the endpoint of $ca(v)$ is therefore a vertex u in U. The algorithm labels u and sets $\delta(u)$ to $\delta(v) + c(ca(v))$. In order to maintain the invariant of Phase I, the algorithm advances the current-arc pointer of u and the current-arc pointers of all the vertices whose current arcs enter u; the pointers are advanced to the next arc with endpoint in $V - S$ in the respective adjacency lists. Phase I ends when $|U|$ becomes $\lfloor n/\log n \rfloor$; let U_0 be the set of unlabeled vertices at the end of Phase I.

When $|U| \leq \lfloor n/\log n \rfloor$, the algorithm is said to be in *Phase II*, and instead of the additional variant of Phase I, the weaker additional invariant is maintained that the endpoint of every current arc belongs to U_0. Suppose that the current arc $ca(v) = (v, w)$ is selected in an iteration. The vertex $w \in U_0$ is not necessarily unlabeled. If w is unlabeled, it is labeled, $\delta(w)$ is set to $\delta(v) + c(ca(v))$, and $ca(v)$ and $ca(w)$ are advanced to the next arc whose endpoint is in U_0. If w is already labeled, only $ca(v)$ is advanced.

Lemma 5.1 *The algorithm spends time $O((n + \xi) \log n + \mu)$ on solving a single-source shortest-paths problem, where ξ is the number of iterations in Phase II and μ is the total number of arcs scanned in the two phases.*

Proof. Since the algorithm does exactly $n - \lfloor n/\log n \rfloor$ iterations in Phase I, it performs a total number of $O(n+\xi)$ iterations. In each iteration, we select a current arc of minimum potential, and we have to update the current-arc pointers as well as the information on their potentials. The cost of updating the current-arc pointers in an iteration is proportional to the increase $\triangle\mu$ in the number of scanned arcs. The lemma follows if we prove that, in each iteration, selecting the current arc of minimum potential and updating the information on the potentials can be done in $O(\log n + \triangle\mu)$ time.

Both phases use a *priority queue* for maintaining information on arc potentials. A priority queue stores a set of pairs (x, k) where k, the *key* of the pair, is a real number. We may assume that our implementation of priority queues supports the insertion of a new pair in constant time, the deletion of a pair with minimum key (a *delete min* operation) in time $O(\log |Q|)$, where $|Q|$ is the number of pairs in the priority queue, and an operation *decrease key* in constant time. A *decrease key* operation takes a pointer to a pair (x, k) in the priority queue and allows the replacement of k by a smaller key k'; see [21, 9].

We propose the following implementation of Phase I. We batch the current arcs with respect to their endpoints, that is, the priority queue contains all unlabeled vertices. For each vertex $u \in U$, we maintain a list $L(u)$ of all vertices $v \in S$ whose current arc enters u; the key of a vertex $u \in U$ is $k_u := \min_{v \in L(u)} \delta(v) + c(v, u)$ (understood to be $+\infty$ if $L(u) = \varnothing$). An iteration of the algorithm corresponds to selecting the vertex $u \in U$ with minimal key value k_u and deleting u from the priority queue with a *delete min* operation. The current-arc pointer must be advanced for each vertex $v \in \{u\} \cup L(u)$. Let $ca(v) = (v, w)$ be the new current arc of v and denote w's current key by k_w. We add v to $L(w)$, and if $\delta(v) + c(v, w) < k_w$, we decrease k_w appropriately. This is realized through a *decrease key* operation on the priority queue. By our assumption on the implementation of the priority queue, all of this takes time $O(\log n + \triangle\mu)$ per iteration.

In Phase II, we represent current arcs by their starting points. We keep the vertices $v \in S$ in

the priority queue with key $\delta(v) + c(ca(v))$. In an iteration of Phase II, selecting the current arc of minimum potential and updating the information on the potentials thus requires a *delete min* operation and the insertion of at most two new pairs in the queue. This takes time $O(\log n)$. □

Remark 5.2 Moffat and Takaoka use a binary heap to realize the priority queue; the implementation described above had not been invented at that time. In the implementation of Phase I, they keep the vertices in S in the priority queue; the key of a vertex $v \in S$ is $\delta(v) + c(ca(v))$. Advancing the current-arc pointers then requires *increasing* the keys of certain labeled vertices, since the cost of the new current arc is greater than the cost of the old current arc. An *increase key* operation in general takes logarithmic time in a binary heap. However, if networks are generated according to the endpoint-independent model, Moffat and Takaoka show how to modify the implementation so that the *expected* cost of all the *increase key* operations in an iteration is $O(|S|/(n-|S|) + \log|S|)$, which is $O(\log n)$ during Phase I.

5.2.1 The probabilistic analysis (and its pitfalls)

If networks with sorted adjacency lists are generated according to the endpoint-independent model, the algorithm of Moffat and Takaoka solves the all-pairs shortest-path problem in this model in expected time $O(n^2 \log n)$; more precisely, it solves each single-source shortest-paths problem in expected time $O(n \log n)$. (Theorem 6.4 in Section 6.1 shows that the running time for solving the single-source shortest-paths problem is actually optimal for a large class of related probability distributions.) As indicated in Lemma 5.1, the quantities of interest in the analysis are the number ξ of iterations in Phase II and the total number μ of scanned arcs. We argue in Theorem 5.3 that the expected values of ξ and μ are $O(n)$ and $O(n \log n)$, respectively.

The analysis of μ turns out to be intricate. We want to mention two possible pitfalls. What is the total number of arcs scanned in Phase I? In [65], Moffat and Takaoka argue as follows. The cardinality of U_0, the set of unlabeled vertices at the end of Phase I, is $\lfloor n/\log n \rfloor$ by design, and at the end of Phase I, current-arc pointers have been advanced to the first vertex in U_0 in each adjacency list. Since, for every vertex v, the endpoints of the arcs out of v form a random permutation of V, the vertices in U_0 are randomly scattered throughout each adjacency list. We should therefore expect to scan about $\log n$ arcs in each adjacency list during Phase I and hence about $n \log n$ arcs altogether. This argument is incorrect as U_0 is determined by the orderings of the adjacency lists and cannot be fixed independently. The following example makes this clear. Assume that all arcs out of the source have cost 1 and all other arcs have cost 2. Then Phase I scans $n - \lfloor n/\log n \rfloor$ arcs out of the source and U_0 is determined by the last $\lfloor n/\log n \rfloor$ arcs in the adjacency list of the source.

In Phase II, the number of iterations is a random variable with expected value $O(n)$. Moreover, whenever the current arc of a vertex is advanced in Phase II, it is advanced to the next arc having its endpoint in U_0, and this requires scanning $O(\log n)$ arcs on average. It is tempting to state that the expected number of arcs scanned in Phase II is therefore $O(n \log n)$. The claimed result would follow if the expected number of scanned arcs, given that the algorithm finishes Phase II in κ iterations, were $O(\kappa \log n)$, and in fact, in a preliminary version of this chapter, [59], we analyzed Phase II along these lines. We now feel that a more careful argumentation is needed. It is not clear whether the number of iterations and the number of arcs that have to be scanned in an update of the current-arc pointer are independent or, for example, positively correlated random variables.

It is for these reasons that we give a new proof of the following theorem. Our proof evolved from suggestions by Alistair Moffat (personal communication) and by two anonymous journal referees and replaces a considerably more involved argument in earlier versions of this chapter.

Theorem 5.3 *On networks (D_n, c) generated according to the endpoint-independent model, the algorithm of Moffat and Takaoka runs in expected time $O(n^2 \log n)$.*

Proof. For the purpose of the analysis, we also consider Spira's algorithm [78, 10]. This algorithm is similar to the algorithm by Moffat and Takaoka, the only difference being that Spira's algorithm does not impose any condition on the endpoints of current arcs but always advances the current-arc pointer only to the next arc in the adjacency list. The algorithm does not distinguish between phases. It stops when all vertices have been labeled. Given an ordering of the adjacency lists, the algorithms by Moffat and Takaoka and by Spira show basically the same behavior. All arcs that the algorithm by Moffat and Takaoka selects as arcs of minimum potential are also selected by Spira's algorithm. However, upon termination, the current-arc pointers in the algorithm of Moffat and Takaoka may have been advanced beyond those in Spira's algorithm, since the invariants of the algorithm by Moffat and Takaoka require that the endpoint of every current-arc pointer is a vertex in U_0. (Nevertheless, the algorithm by Moffat and Takaoka is more efficient, since the scanning strategies of Phase I and II tend to reduce the number of priority-queue operations.)

The following observations are crucial for the analysis of the algorithms. Suppose we stop Spira's algorithm after its first κ iterations, where κ is an arbitrary but fixed number. The behavior of the algorithm in these iterations is completely determined by the arcs scanned so far. For a set A of arcs, denote by $\{\mathscr{A}_\kappa = A\}$ the event that Spira's algorithm scans exactly the arcs in A in the first κ iterations. We consider an arbitrary but fixed set A with $\mathbf{Pr}(\mathscr{A}_\kappa = A) > 0$; assume that for $v \in V$, A contains exactly n_v arcs with starting point v and W_v is the set of their endpoints. By the definition of the endpoint-independent model and the principle of deferred decisions, for each $v \in V$, the remaining part of v's adjacency list can be interpreted as a random permutation

of $V - W_v$, since the event $\{\mathscr{A}_\kappa = A\}$ does not yield any information about the remaining parts of the adjacency lists.

From this we conclude that the total number of arcs scanned by Spira's algorithm is stochastically dominated by the completion time of the coupon-collector problem with n coupons. Namely, assume that $\mathscr{A}_\kappa = A$ implies that exactly i vertices have been labeled in the first κ iterations. If the next arc scanned has starting point v, then the endpoint of the arc is already labeled with probability $(i - n_v)/(n - n_v) \leq i/n$, since every vertex in $V - W_v$ is equally likely to occur as the endpoint of v's current arc. More generally, the probability that the algorithm selects arcs with labeled endpoints in the next k iterations is bounded from above by $(i/n)^k = (1 - (n-i)/n)^k$ for every $k \geq 0$. For $1 \leq i \leq n$, let M_i denote the number of arcs scanned by Spira's algorithm between the labelings of the i-th and the $(i+1)$-th vertex, and let Z_i denote the random variable introduced in the analysis of the coupon-collector problem in Section 5.1. The conclusion we have just derived then reads $\mathbf{Pr}(M_i > k) \leq (1 - (n-i)/n)^k = \mathbf{Pr}(Z_i > k)$, for every $k \geq 0$, that is, $M_i \leq_{\text{st}} Z_i$. By Lemma 2.4 in Section 2.2.1 we conclude that $M := 1 + M_1 + \cdots + M_{n-1}$, the total number of arcs scanned by Spira's algorithm, is indeed stochastically dominated by the completion time $Z^* = 1 + Z_1 + \cdots + Z_{n-1}$ of the coupon-collector problem with n coupons. In particular,

$$\mathbf{E}[M] \;\leq\; \mathbf{E}[Z^*] \;\leq\; n(\log n + 1) \;, \tag{5.3}$$

that is, Spira's algorithm scans an expected number of $O(n \log n)$ arcs.

An argument of the same kind as in the previous paragraph applies to the endpoints of current arcs in Phase II of the algorithm by Moffat and Takaoka. If X denotes the number of iterations in Phase II, then X is stochastically dominated by the completion time of a coupon-collector problem with $b := |U_0| = \lfloor n/\log n \rfloor$ coupons; in particular,

$$\mathbf{E}[X] \;\leq\; b(\log b + 1) \;=\; O(n) \;.$$

The expected value of ξ in Lemma 5.1 is therefore $O(n)$.

It remains to analyze the number of extra arcs scanned by the algorithm of Moffat and Takaoka. For a set A of arcs, denote by $\{\mathscr{A}^* = A\}$ the event that Spira's algorithm scans exactly these arcs before it stops. We consider an arbitrary but fixed set A with $\mathbf{Pr}(\mathscr{A}^* = A) > 0$; assume that for $v \in V$, A contains exactly n_v arcs with starting point v and b_v of these arcs have endpoints in U_0. Given that $\{\mathscr{A}^* = A\}$ occurs, for each $v \in V$, the current-arc pointer in the algorithm by Moffat and Takaoka has finally been advanced to the $(n_v + Y_v)$-th position in v's adjacency list, where $Y_v = 0$ if $b_v = b = |U_0|$ and, otherwise, $n_v + Y_v$ is the position of the next vertex in U_0 in v's adjacency list. Again, by the principle of deferred decisions, the remaining part of v's adjacency list can be interpreted as a random permutation of $n - n_v$ elements, containing $b - b_v$ elements from

U_0. This implies that Y_v is distributed as in the urn experiment in Section 5.1 with

$$\mathbf{E}[Y_v \mid \mathscr{A}^* = A] \;=\; \frac{n - n_v + 1}{b - b_v + 1} \quad \text{if } b_v < b \;. \tag{5.4}$$

Note that $\mathbf{E}[Y_v \mid \mathscr{A}^* = A] \leq 2n/b$ as long as $b_v \leq b/2$.

The expected amount of extra work is $\mathbf{E}[\sum_v Y_v]$, where here and in the following, $\sum_v$ denotes the summation over all $v \in V$. Conditioning on $\{\mathscr{A}^* = A\}$, we get, using (5.4) (or the trivial bound $Y_v \leq n$ if $b_v > b/2$),

$$\mathbf{E}\left[\textstyle\sum_v Y_v\right| \mathscr{A}^* = A] \;\leq\; \tfrac{2n}{b} \cdot |\{v \,;\, b_v \leq b/2\}| + n \cdot |\{v \,;\, b_v > b/2\}| \;\leq\; \tfrac{2n^2}{b} + n \cdot \tfrac{2}{b} \cdot \textstyle\sum_v b_v \;. \tag{5.5}$$

Under the condition $\{\mathscr{A}^* = A\}$, the number X of iterations in Phase II equals $\sum_v b_v$. Hence, by summing over all sets A with $\mathbf{Pr}(\mathscr{A}^* = A) > 0$, (5.5) implies

$$\mathbf{E}[\textstyle\sum_v Y_v] \;=\; \sum_A \mathbf{E}[\sum_v Y_v \mid \mathscr{A}^* = A] \cdot \mathbf{Pr}(\mathscr{A}^* = A) \;\leq\; \tfrac{2n}{b} \cdot (n + \mathbf{E}[X]) \;,$$

and since $b = \lfloor n/\log n \rfloor$ and $\mathbf{E}[X] = O(n)$, we get $\mathbf{E}[\sum_v Y_v] = O(n \log n)$. Combining this with (5.3), we conclude that the expected value of μ in Lemma 5.1 is $O(n \log n)$.

We deduce from Lemma 5.1 that the algorithm of Moffat and Takaoka solves any single-source shortest-paths problem in expected time $O(n \log n)$ and has therefore an expected running time of $O(n^2 \log n)$. □

We next prove that the algorithm of Moffat and Takaoka is reliable, that is, that, with high probability, its running time does not exceed its expectation by more than a constant multiplicative factor.

Theorem 5.4 *On networks (D_n, c) generated according to the endpoint-independent model, the running time of the all-pairs shortest-path algorithm of Moffat and Takaoka is $O(n^2 \log n)$ with high probability.*

Proof. It suffices to prove that the algorithm solves any single-source shortest-paths problem in time $O(n \log n)$ with high probability. This follows from Lemma 5.1 if the total number of iterations and the total number of scanned arcs can be proved to be, with high probability, $O(n)$ and $O(n \log n)$, respectively. We use the notation introduced for the proof of Theorem 5.3.

As already observed in the proof of Theorem 5.3, the total number X of iterations in Phase II is stochastically dominated by the completion time of a coupon-collector problem with $b = \lfloor n/\log n \rfloor$ coupons. Using the tail estimate (5.2) in Section 5.1, we deduce that the number of iterations in

Phase II is $O(b \log b) = O(n)$ with high probability; for any arbitrary (but fixed) $\gamma > 0$, we can choose some constant K so that

$$\mathbf{Pr}(X > Kn) \;\leq\; \tfrac{n}{\log n}\left(1 - \tfrac{1}{\lfloor n/\log n\rfloor}\right)^{Kn} \;\leq\; \tfrac{n}{\log n}\cdot e^{-K\log n} \;=\; O(n^{-\gamma})\;. \tag{5.6}$$

Hence, the total number of iterations is $O(n)$ with high probability.

Again, by the tail estimate (5.2) for the coupon-collector problem, Spira's algorithm scans $O(n \log n)$ arcs with high probability. Therefore, we only have to prove that the extra scanning $\sum_v Y_v$ of the algorithm of Moffat and Takaoka (the sum $\sum_v$ is over all $v \in V$) is $O(n \log n)$ with high probability. As in the proof of Theorem 5.3, we condition on $\{\mathscr{A}^* = A\}$, that is, on the event that Spira's algorithm scans exactly the arcs in A before it stops. For $v \in V$, let A contain exactly n_v arcs with starting point v and let b_v of these arcs have endpoints in U_0. Given $\{\mathscr{A}^* = A\}$, we have, with $b = |U_0| = \lfloor n/\log n\rfloor$ and for sufficiently large n,

$$\textstyle\sum_{v,b_v>b/2} Y_v \;\leq\; n\cdot|\{v\,;\,b_v > b/2\}| \;\leq\; \frac{2n}{b}\cdot\left(\sum_v b_v\right) \;\leq\; (3\log n)\cdot X\;,$$

since under the condition $\{\mathscr{A}^* = A\}$, the number X of iterations in Phase II equals $\sum_v b_v$. Thus

$$\mathbf{Pr}\left(\textstyle\sum_{v,b_v>b/2} Y_v > 3Kn\log n \mid \mathscr{A}^* = A\right) \;\leq\; \mathbf{Pr}(X > Kn \mid \mathscr{A}^* = A)\;,$$

from which we conclude that

$$\begin{aligned} &\mathbf{Pr}\left(\textstyle\sum_v Y_v > 4Kn\log n \mid \mathscr{A}^* = A\right)\\ \leq\;& \mathbf{Pr}\left(\textstyle\sum_{v,b_v\leq b/2} Y_v > Kn\log n \;\middle|\; \mathscr{A}^* = A\right) + \mathbf{Pr}\left(X > Kn \mid \mathscr{A}^* = A\right)\;. \end{aligned} \tag{5.7}$$

To bound the first term in (5.7), observe that, conditionally on $\{\mathscr{A}^* = A\}$, $\sum_{v,b_v\leq b/2} Y_v/n$ is a sum of independent (not necessarily identically distributed) random variables with values in $[0,1]$. We can therefore derive a large-deviation estimate by applying a Chernoff–Hoeffding bound. By (5.4),

$$\mathbf{E}\left[\textstyle\sum_{v,b_v\leq b/2} Y_v/n \;\middle|\; \mathscr{A}^* = A\right] \;=\; \sum_{v,b_v\leq b/2} \frac{1}{n}\cdot\frac{n-n_v+1}{b-b_v+1} \;\leq\; \frac{2n}{b} \;\leq\; 3\log n\;,$$

for sufficiently large n, independently of A. Hence, for K chosen sufficiently large, we deduce from (2.12) in Section 2.2.2 that

$$\mathbf{Pr}\left(\textstyle\sum_{v,b_v\leq b/2} Y_v/n > K\log n \;\middle|\; \mathscr{A}^* = A\right) \;\leq\; e^{-K\log n}\;.$$

When plugged into the inequality (5.7), this large-deviation bound together with the one in (5.6) imply that $\sum_v Y_v = O(n \log n)$ with high probability. □

Chapter 6

Lower bounds

In this chapter, we are concerned with establishing lower bounds on the running time of algorithms that solve shortest-paths problems. If we took our task to an extreme, we would have to prove that, for networks of a given size, *each* algorithm performs at least N operations on *any* problem instance; it is to be understood that N should be a non-trivial lower bound. This task seems almost impossible to accomplish if we do not focus on a restricted class of algorithms, possibly with the additional simplification of proving only the existence of some network on which the algorithms from this class have to perform a certain number of operations to compute the distances correctly. Naturally, our statements will be the less meaningful the smaller the classes of competing algorithms or the number of considered instances are.

Kolliopoulos and Stein [51] studied the class of so-called *oblivious* algorithms, which includes the Bellman–Ford algorithm. (It is unlikely, however, that it includes many more algorithms, since the characterization of oblivious algorithms is very specific.) It follows from a counting argument that on any m-arc network, any oblivious algorithm must relax $\Omega(\nu m)$ arcs to solve a single-source shortest-paths problem, where ν denotes the maximum number of arcs on a shortest path.

Karger, Koller, and Phillips [49] considered the class of *path-comparison-based* algorithms. On any input network, these algorithms can gain information on the arc costs only by comparing the costs of two different paths, and the complexity of any such algorithm is the number of comparisons performed. Many of the known shortest-paths algorithms are indeed path-comparison-based (with the exception of shortest-paths algorithms based on matrix multiplication). For any n and any $m \geq 2n$ with $m = \Theta(n^2)$, Karger, Koller, and Phillips constructed a (single) network on $\Theta(n)$ vertices and m arcs with non-negative costs, on which any path-comparison-based algorithm must perform $\Omega(mn)$ comparisons to solve the all-pairs shortest-paths problem. (In fact, for any m^*, $2n \leq m^* \leq m$, a network can be constructed in which m^* of the m arcs are essential, that

is, contained in shortest paths, and any path-comparison-based algorithm compares the costs of $\Omega(m^*n)$ paths on these networks.)

The model of computation studied by Karger, Koller, and Phillips is a restricted version of the *linear decision-tree model*. In this model, an algorithm proceeds on n-vertex networks with arc costs c according to a rooted ternary tree. The internal nodes of the tree are labeled by tests of the form $\langle h, c\rangle \,?\, \eta$, for some $h \in \mathbb{R}^{n^2}$ and $\eta \in \mathbb{R}$, where $\langle \cdot, \cdot \rangle$ denotes the Euclidean scalar product on $\mathbb{R}^{n^2}$; the three arcs leaving an internal node are labeled $<$, $=$, and $>$, respectively. (For example, a comparison of path costs can be represented by a linear test.) The algorithms starts from the root; it proceeds by moving down the tree, branching at the internal nodes according to the outcomes of the linear tests. When a leaf is reached, the knowledge about the input deduced from the outcomes of the tests allows the algorithm to solve the shortest-paths problem under consideration on input c correctly. The complexity of the algorithm is defined to be the height of the tree, which is equal to the number of linear tests that the algorithm performs in the worst case.[7]

Graham, Yao, and Yao [37] argued that when solving the all-pairs shortest-paths problem on networks with non-negative arc costs, the information collected at any leaf of a linear decision tree suffices to compute the actual shortest paths between all pairs of vertices. Perhaps surprisingly (and in contrast to what had been claimed in an earlier paper [88]), there are, for some constant C, only $\exp(Cn^2)$ distinct shortest-paths patterns for networks on n vertices. Therefore, this information-theoretic argument only provides a lower bound of $\Omega(n^2)$ on the complexity of the all-pairs shortest-paths problem in the linear decision-tree model. Dietzfelbinger and Maass [19] considered the complexity of a decision problem corresponding to the single-source *single-sink* shortest-path problem: For a given pair s, t of vertices in a complete (undirected) graph with non-negative real arc costs, decide whether there is a path between s and t of cost less than 1; we denote by $\textsc{sp}(n)$ the language corresponding to this decision problem when inputs to the problem are networks on n vertices. The authors proved that any linear decision tree T_n that recognizes $\textsc{sp}(n)$ and that uses fewer than $f(n)$ negative coefficients in any of its tests must have height at least $2^{\lfloor\sqrt{n}/(4f(n))\rfloor}$; see [19, proof of Theorem 3]. The lower bound on the height of T_n is superpolynomial in the input size if $f(n) = o(\sqrt{n}/\log_2 n)$. Hence, it is inherent to shortest-paths problems that efficient algorithms solving them compare sums formed of many arc costs.

We prove a lower bound on the average-case complexity of the *single-source shortest-paths problem*. Our result is different from the work reviewed above in that the bound holds on almost all instances from a fairly broad class of networks. Our assumptions on the computational primitives are very

[7] *Upper* bounds on the complexity of algorithms derived in the linear decision-tree model have to be taken with a grain of salt. For example, Fredman [27] proved that the all-pairs shortest-paths problem on instances of size n can be solved with $O(n^{5/2})$ linear tests, but the sizes of the decision trees that realize these computations may grow exponentially in n.

natural and should exclude hardly any algorithm from our considerations. Furthermore, the algorithms are allowed to exploit the fact that arc costs are integers. We actually restrict ourselves to what we call *simple* cost functions: For every vertex v and each integer k, $1 \leq k \leq n$, there is exactly one arc with cost k and starting point v.

More precisely, for any n, the underlying input model generates complete directed graphs on n vertices with arc costs by the following random experiments. Similar to how instances are generated in the endpoint-independent model, a random cost function is given by n independent permutations of V, one for each vertex, drawn uniformly at random, which determine the order of the endpoints as they appear in the sorted adjacency lists. The i-th vertex in the sorted adjacency list of vertex v is the endpoint of the arc with cost i and starting point v. We prove that on a random simple cost function, any algorithm has to inspect $\Omega(n \log n)$ arcs with high probability to solve the single-source shortest-paths problem correctly; see Theorem 6.4. This result represents joint work with Kurt Mehlhorn [60].

6.1 The single-source shortest-paths problem for simple arc costs

We consider networks that are complete directed graph (with loops) on the set V of n vertices with simple cost functions. A single-source shortest-paths algorithm gets as its input the number n of vertices, a source vertex s, and a simple cost function c. We assume that c is provided in the form of an oracle that answers questions of the following kind:

(Q1) What is the cost $c(a)$ of a given arc a?

(Q2) Given a vertex $v \in V$ and an integer $k \in \{1, \ldots, n\}$, what is the endpoint of the arc with starting point v and cost k?

The algorithm is supposed to compute the function δ of shortest-path distances from s. It is allowed to ask the oracle questions of type (Q1) and (Q2), thereby gaining partial information about c. The complexity of the algorithm on a fixed simple cost function c is defined to be the number of questions asked by the algorithm in order to compute the distance function δ with respect to c.

For simple cost functions, the distance function δ maps the set of vertices into the non-negative integers. Let $\Delta = \max\{\delta(v)\ ;\ v \in V\}$ and for all i, $0 \leq i \leq \Delta$, define $V_i := \{v\ ;\ \delta(v) = i\}$. We call V_i the i-th layer with respect to δ. For all i, $0 \leq i \leq \Delta$, let $\ell(i) := |\{j\ ;\ j > i \text{ and } V_j \neq \varnothing\}|$ be the number of non-empty layers above layer i. Clearly, Δ, the sets V_i, and the function ℓ depend on c and s; for ease of notation, we do not make this dependence visible in the notation.

We first provide a lower bound on the complexity of a single-source shortest-paths algorithm in terms of ℓ.

Lemma 6.1 *Let c be a simple cost function and let δ be the distance function with respect to c. Then any single-source shortest-paths algorithm has complexity at least*

$$\sum_{v \in V - V_\Delta} \Big(\ell(\delta(v)) - 1 \Big) \ . \tag{6.1}$$

Proof. We consider the complexity of an arbitrary but fixed single-source shortest-paths algorithm. Let A' be the set of arcs queried by the algorithm by a question of either type (Q1) or type (Q2). For each vertex $v \in V$, let $A(v)$ be the set of arcs with starting point v and let $A'(v) := A' \cap A(v)$.

We prove that for each vertex v with $\ell(\delta(v)) \geq 2$, $|A'(v)| \geq \ell(\delta(v)) - 1$. This is clear if for all i with $1 \leq i < \ell(\delta(v))$, the set A' contains the arc $a \in A(v)$ of cost $c(a) = i$. If, instead, there is an arc $a_\circ \in A(v) - A'$ with cost $c_\circ := c(a_\circ) < \ell(\delta(v))$, then the argument is slightly more involved. Assume that another arc $a_j = (v, w)$ with $w \in V_j$, different from $a_\circ$, is not queried by the algorithm. Define the simple cost function c' by

$$c'(a) := \begin{cases} c(a), & \text{if } a \notin \{a_\circ, a_j\} \ ; \\ c(a_j), & \text{if } a = a_\circ \ ; \\ c_\circ, & \text{if } a = a_j \ . \end{cases}$$

Then $c'(a) = c(a)$ for all $a \in A'$, and therefore the algorithm outputs δ, the distance function with respect to c, on input c' as well. By the correctness condition (4.1) with respect to input c' and for $a_j = (v, w)$, the layer index j necessarily satisfies

$$j \ = \ \delta(w) \ \leq \ \delta(v) + c'(v, w) \ = \ \delta(v) + c_\circ \ .$$

In turn, this proves that all vertices in layers V_j with $j > \delta(v) + c_\circ$ must be endpoints of arcs in $A'(v)$. (Non-empty layers above layer $\delta(v) + c_\circ$ do indeed exist, since $\ell(\delta(v) + c_\circ) \geq \ell(\delta(v)) - c_\circ > 0$, by the assumption on $c_\circ$.) By the correctness of the single-source shortest-paths algorithm on input c, all arcs in $A(v)$ with endpoints in a layer V_j, $j > \delta(v)$, must have cost at least $j - \delta(v)$. If we choose $c_\circ = \min\{c(a) \, ; \, a \in A(v) - A'\}$, then the $c_\circ - 1$ arcs in $A'(v)$ with cost less than $c_\circ$ and the arcs in $A(v)$ with endpoints in a layer V_j with $j > \delta(v) + c_\circ$ are distinct, since the latter ones must have costs at least $j - \delta(v) > c_\circ$. Hence, $|A'(v)| \geq c_\circ - 1 + \ell(\delta(v) + c_\circ) \geq \ell(\delta(v)) - 1$, as argued above. The lower bound (6.1) follows, since, by definition, the complexity of the algorithm equals $|A'| = \sum_{v \in V} |A'(v)|$. □

Table 6.1 shows a typical distribution of vertices over distances for a random simple cost function on a graph of $n = 10,000$ vertices. Observe that during the first stages, there are indeed 2^i vertices with distances at most i, as argued in Remark 3.3. Most vertices have distance about 14 ($\approx \log n$) from the source but there are vertices that have distance as much as 24 ($\approx 2 \log n$). By the argument of Lemma 6.1, we can guess that any (correct) algorithm must inquire about $\Omega(n \log n)$ arcs.

Table 6.1: A typical distribution of vertices over distances for $n = 10,000$

distance δ	0	1	2	3	4	5	6	7	8	9	10	11	12
# vertices	1	1	2	4	8	16	32	64	120	237	449	796	1306

distance δ	13	14	15	16	17	18	19	20	21	22	23	24
# vertices	1845	1952	1562	910	415	181	58	20	16	2	1	2

In the remainder of this section, we make this argument more precise. We derive a lower bound of $\Omega(n \log n)$ on the expected value of $\sum_{v \in V} \ell(\delta(v))$ for networks on n vertices when simple cost functions are randomly generated. More generally, we show that any single-source shortest-paths algorithm has to ask $\Omega(n \log n)$ questions with high probability.

Our proof strategy is as follows. The lower bound given by Lemma 6.1 depends only on the distance function δ. For random simple cost functions, we re-interpret the calculation of δ and the construction of the layers V_i as the outcome of a random labeling process, very much in the spirit of the processes that we studied in Section 3.3.1 (construction of the spanning arborescence) and in Section 5.2.1 (analysis of the algorithm of Moffat and Takaoka). For an arbitrary but fixed source vertex s, the labeling process proceeds in *stages*. In the zero stage, V_0 is set to $\{s\}$ and $\delta(s)$ is set to 0. In the i-th stage, $i \geq 1$, each vertex $v \in S^{(i-1)} = \bigcup_{0 \leq j < i} V_j$ picks the $(i - \delta(v))$-th vertex in its adjacency list. The newly-reached vertices are put into V_i and their δ-values are set to i. We apply the principle of deferred decisions again—instead of fixing the n permutations beforehand, we view them as being fixed on-line. This leads to the following re-interpretation of the random labeling process: In the i-th stage, $i \geq 1$, each vertex in $S^{(i-1)} = \bigcup_{0 \leq j < i} V_j$ chooses a vertex uniformly and independently at random from the set of vertices it has not yet seen. The labeling process stops when $S^{(k)} = V$ for some k.

We have already referred in Section 3.3.1 to a result of Frieze and Grimmett [31], who proved that this labeling process takes $O(\log n)$ stages with high probability. However, we need a lower bound on the number of stages and therefore their result is of no use to us here.

For $i \geq 1$, we call stage i of the labeling process *central* if $n/e \leq |S^{(i)}| \leq n - \sqrt{n}$. Layers constructed in central stages are called central. Our proof proceeds in two steps. First, we show in Lemma 6.2 that there are $\Omega(\log n)$ central stages with high probability. Second, we prove in Lemma 6.3 that each central stage gives rise to a non-empty layer with high probability.

Lemma 6.2 *With high probability, the labeling process with respect to random simple cost functions has $\Omega(\log n)$ central stages.*

Proof. For a simple cost function c, let i_0 be the first central stage with respect to c. Then $n/e \leq |S^{(i_0)}| \leq 2n/e$, since $|S^{(i+1)}| \leq 2|S^{(i)}|$ for any $i \geq 0$. We show that if c was chosen at

random, then with high probability, the next $k = \lfloor (\log n)/17 \rfloor$ stages are also central, that is, $|S^{(i_0+k)}| \le n - \sqrt{n}$. Let $U = V - S^{(i_0)}$ be the set of vertices that are still unlabeled after stage i_0. Note that $m := |U| \ge (e-2)n/e \ge n/4$.

In an ancillary experiment, we construct an $n \times m$ matrix A with 0-1 entries as follows. The rows correspond to the vertices in V and the columns correspond to the vertices in U; entry A_{vu} is 1 if and only if the arc (v, u) is among the k shortest (with respect to c) of those arcs in v's adjacency list whose endpoints are elements of U. Let $f(A)$ be the number of all-zero columns in A. Then $|S^{(i_0+k)}| \le n - f(A)$ because no vertex in U corresponding to an all-zero column in A is labeled in the k stages following stage i_0. Since A models a process in which all vertices (and not only those that are currently labeled) are allowed to label new vertices, and in which each vertex is prevented from choosing vertices that have been labeled by other vertices before stage i_0, $f(A)$ may seem to be a rather crude lower bound on $|V - S^{(i_0+k)}|$. However, we now prove that for random simple cost functions, $f(A) \ge \sqrt{n}$ with high probability.

According to the labeling process with deferred decisions, each row of A is a random 0-1 vector of length $m = |U|$ with exactly k 1-entries, where each distribution of 1-entries occurs with probability $1/\binom{m}{k}$. Row entries $A_{v\cdot}$, $v \in V$, are independent random variables, but the entries A_{vu}, $u \in U$, of row v are certainly not independent. However, the vector $(A_{vu}, u \in U)$ is negatively associated (Definition 2.6); see [46, Theorem 2.11] or [22] for a proof. It thus follows from Proposition 2.8(a) that the whole vector $(A_{vu}, v \in V, u \in U)$ of the random entries in A is negatively associated. For $u \in U$, let C_u denote the 0-1 indicator variable that takes the value 1 if and only if column u in A is all-zero. For any $u \in U$, the variable C_u can thus be written as $1 - \operatorname{sgn} \sum_{v \in V} A_{vu}$, where $\operatorname{sgn} 0 := 0$ and $\operatorname{sgn} x := 1$ for $x > 0$. The random variables C_u, $u \in U$, are therefore non-increasing functions of pairwise disjoint subsets of the entries in A. By Proposition 2.8(b), the random variables $(C_u, u \in U)$ are negatively associated, and we can apply the large-deviation bound of Lemma 2.7 to $f(A) = \sum_{u \in U} C_u$.

The probability that a fixed column is all-zero is $\left(\binom{m-1}{k} / \binom{m}{k}\right)^n = (1 - k/m)^n$; therefore,

$$\mathbf{E}[f(A) \mid |U| = m] \;=\; m\left(1 - \tfrac{k}{m}\right)^n \;\ge\; me^{-2kn/m} \;, \tag{6.2}$$

since $(1 - 1/x)^x \ge e^{-2}$ for sufficiently large x. If we use in (6.2) that $m \ge n/4$ and $k = \lfloor (\log n)/17 \rfloor$, we get, for sufficiently large n,

$$\mathbf{E}[f(A) \mid |U| = m] \;\ge\; \tfrac{1}{4} \cdot n^{1-8/17} \;, \tag{6.3}$$

which is greater than $2\sqrt{n}$ for sufficiently large n. The lower bound in (6.3) is independent of m; therefore, $\mathbf{E}[f(A)] \ge 2\sqrt{n}$ for sufficiently large n. Hence, by the large-deviation bound (2.17)

$$\mathbf{Pr}(f(A) \le \sqrt{n}) \;\le\; \mathbf{Pr}\Big(f(A) \le \mathbf{E}[f(A)]/2\Big) \;\le\; e^{-\mathbf{E}[f(A)]/8} \;\le\; e^{-\sqrt{n}/4} \;=\; O(n^{-\gamma})$$

for any fixed $\gamma > 0$, if n is sufficiently large. We have thus proved that with high probability, more than $\sqrt{n}$ columns contain only zeroes in our ancillary matrix experiment. With i_0 and k as defined at the beginning of the proof, this implies that with high probability, $|S^{(i_0+k)}| \leq n - \sqrt{n}$ in the labeling process. □

Lemma 6.3 *With high probability, each central layer in the labeling process with respect to a random simple cost function contains at least one vertex.*

Proof. Suppose that $|S^{(i)}| = j$ for some i and j, that is, j vertices are already labeled after stage i. According to the labeling process with deferred decisions, for any vertex in $S^{(i)}$, the probability of selecting a vertex in $S^{(i)}$ during this stage is at most j/n. Therefore, the next layer remains empty with probability at most $(j/n)^j$. Note that $x \mapsto (x/n)^x$ is an increasing function for $x > n/e$.

Let $\mathscr{E}$ denote the event that in the labeling process on random simple cost functions, at least one central layer remains empty. By the estimates provided in the preceding paragraph,

$$\mathbf{Pr}(\mathscr{E}) \;\leq\; \sum_{j=n/e}^{n-\sqrt{n}} \left(\frac{j}{n}\right)^j \;\leq\; n\left(\frac{n-\sqrt{n}}{n}\right)^{n-\sqrt{n}} \;\leq\; ne^{-\sqrt{n}+1} \;=\; O(n^{-\gamma})$$

for any fixed $\gamma > 0$, if n is sufficiently large. □

Theorem 6.4 *Any algorithm for the single-source shortest-paths problem has complexity $\Omega(n \log n)$ with high probability on random simple cost functions.*

Proof. Let i denote the first central stage of the labeling process, and let $S^{(i)}$ denote the set of vertices that have already been labeled up to and in this stage. By definition, $|S^{(i)}| \geq n/e$. By Lemma 6.2, with high probability, the process has $\Omega(\log n)$ central layers. Lemma 6.3 tells us that all central layers are non-empty with high probability. With the notation introduced in the discussion of the labeling process, this reads

$$\sum_{u \in S^{(i)}} \Big(\ell(\delta(u)) - 1\Big) \;=\; \Omega(n \log n) \quad \text{with high probability.}$$

By Lemma 6.1, the left-hand side term is a lower bound on the complexity of any single-source shortest-paths algorithm. □

Any algorithm can gain complete knowledge of the cost function by $O(n^2)$ queries to the oracle. Since our input model does not restrict the computational power of an algorithm, a non-trivial lower bound on the complexity of the all-pairs shortest-paths problem (such as, say, $\Omega(n^2 \log n)$) can "certainly not" [50] be obtained under the general assumptions of Theorem 6.4.

Appendix A

Lemma 4.2(b) from [31]

For the sake of completeness, we include a proof of the following lemma, though it appeared almost verbatim in [31]. Our version is different in that it does not necessarily assume that $\kappa = \Theta(\ln \ell)$. This is important for the use of this lemma in Section 3.3.2.

Lemma A.1 ([31, Lemma 4.2(b)]) *Let $U_{(k:\ell)}$ denote a random variable distributed as the k-th order statistic of a sample of ℓ independent random variables that are uniformly distributed on $[0,1]$. Suppose that $Y_1, Y_2, \ldots, Y_m$ are independent random variables with Y_i distributed as $U_{(k_i:\ell)}$, for $i = 1, 2, \ldots, m$, and let κ be fixed with $k_1 + k_2 + \cdots + k_m \leq \kappa$. If $\mu > 1$, then*

$$\mathbf{Pr}\left(Y_1 + Y_2 + \cdots + Y_m \geq \tfrac{\mu\kappa}{\ell+1}\right) \;\leq\; e^{-\mu\kappa}(e\mu)^{\kappa} \;. \tag{A.1}$$

Proof. The random variable $U_{(k:\ell)}$ has density

$$f_k(x) \;=\; \binom{\ell}{k} k x^{k-1}(1-x)^{\ell-k} \;, \quad 0 \leq x \leq 1 \;;$$

see, for example, [38, 39, Ex. 4.11.22(b)]. Hence, for any $i \geq 0$,

$$\begin{aligned}\mathbf{E}[(U_{(k:\ell)})^i] \;&=\; \int_0^1 x^i f_k(x)\,\mathrm{d}x \;=\; \binom{\ell}{k} k \int_0^1 x^{i+k-1}(1-x)^{\ell-k}\,\mathrm{d}x \\ &=\; \binom{\ell}{k} k \frac{(i+k-1)!(\ell-k)!}{(\ell+i)!} \;\leq\; \frac{k(k+1)\cdots(k+i-1)}{(\ell+1)^i} \;,\end{aligned}$$

from which we conclude that, for any $t > 0$,

$$\mathbf{E}[e^{tU_{(k:\ell)}}] \;\leq\; \sum_{i\geq 0} \frac{t^i}{i!} \frac{k(k+1)\cdots(k+i-1)}{(\ell+1)^i} \;=\; \sum_{i\geq 0} \left(\frac{t}{\ell+1}\right)^i \binom{k-1+i}{i} \;=\; \left(1 - \frac{t}{\ell+1}\right)^{-k} \;;$$

see [36, eq. (5.56)]. The (elementary) Bernstein inequality implies that for any real y and any $t > 0$,

$$\mathbf{Pr}(Y_1 + Y_2 + \cdots + Y_m \geq y) \;\leq\; e^{-ty}\mathbf{E}\left[e^{t(Y_1+Y_2+\cdots+Y_m)}\right] \;;$$

and by the assumption that $Y_1, \ldots, Y_m$ are independent random variables, we get

$$\begin{aligned}\mathbf{Pr}(Y_1 + Y_2 + \cdots + Y_m \geq y) &\leq e^{-ty} \prod_{i=1}^{m} \mathbf{E}\left[e^{tY_i}\right] = e^{-ty} \prod_{i=1}^{m} \mathbf{E}\left[e^{tU_{(k_i:\ell)}}\right] \\ &\leq e^{-ty} \prod_{i=1}^{m} \left(1 - \frac{t}{\ell+1}\right)^{-k_i} = e^{-ty} \left(1 - \frac{t}{\ell+1}\right)^{-(k_1+\cdots+k_m)} ,\end{aligned}$$

which because of $\kappa \geq k_1 + \cdots + k_m$ results in

$$\mathbf{Pr}(Y_1 + Y_2 + \cdots + Y_m \geq y) \leq e^{-ty} \left(1 - \frac{t}{\ell+1}\right)^{-\kappa} .$$

The last term is minimal if we choose $t = \ell + 1 - \kappa/y$. We thus have

$$\mathbf{Pr}(Y_1 + Y_2 + \cdots + Y_m \geq y) \leq e^{-(\ell+1)y} \left(\frac{\kappa}{e(\ell+1)y}\right)^{-\kappa} ,$$

and if we choose $y = \mu\kappa/(\ell+1)$, then this reads

$$\mathbf{Pr}\left(Y_1 + Y_2 + \cdots + Y_m \geq \tfrac{\mu\kappa}{\ell+1}\right) \leq e^{-\mu\kappa}(e\mu)^{\kappa} = \exp(-\kappa(\mu - 1 - \log\mu)) .$$

□

Bibliography

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, Englewood Cliffs NJ, 1993

[2] R. K. Ahuja, K. Mehlhorn, J. B. Orlin, and R. E. Tarjan, Faster algorithms for the shortest path problem, *J. Assoc. Comput. Mach.* **37** (1990), pp. 213–223

[3] D. Aingworth, C. Chekuri, P. Indyk, and R. Motwani, Fast estimation of diameter and shortest paths (without matrix multiplication), *SIAM J. Comput.* **28** (1999), pp. 1167–1181

[4] K. Alam and K. M. L. Saxena, Positive dependence in multivariate distributions, *Comm. Statist. Theory Methods* **A10** (1981), pp. 1183–1196

[5] N. Alon and J. H. Spencer, *The Probabilistic Method*, 2nd ed., John Wiley & Sons, New York, 2000

[6] R. Bellman, *Dynamic Programming*, Princeton University Press, Princeton NJ, 1957

[7] R. Bellman, On a routing problem, *Quart. Appl. Math.* **16** (1958), pp. 87–90

[8] P. A. Bloniarz, A shortest-path algorithm with expected time $O(n^2 \log n \log^* n)$, *SIAM J. Comput.* **12** (1983), pp. 588–600

[9] G. S. Brodal, Worst-case efficient priority queues, in *Proc. 7th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1996, pp. 52–58

[10] J. S. Carson and A. M. Law, A note on Spira's algorithm for the all-pairs shortest-path problem, *SIAM J. Comput.* **6** (1977), pp. 696–699

[11] B. V. Cherkassky, A. V. Goldberg, and T. Radzik, Shortest path algorithms: Theory and experimental evaluation, *Math. Programming* **73** (1996), pp. 129–174

[12] B. V. Cherkassky, A. V. Goldberg, and C. Silverstein, Buckets, heaps, lists, and monotone priority queues, *SIAM J. Comput.* **28** (1999), pp. 1326–1346

[13] C. Cooper, A. Frieze, K. Mehlhorn, and V. Priebe, Average-case complexity of shortest-paths problems in the vertex-potential model, in J. Rolim (ed.), *Randomization and Approximation Techniques in Computer Science* (Lecture Notes in Comput. Sci., vol. 1269), Springer-Verlag, Berlin, 1997, pp. 15–26

[14] C. Cooper, A. Frieze, K. Mehlhorn, and V. Priebe, Average-case complexity of shortest-paths problems in the vertex-potential model, *Random Structures Algorithms* **16** (2000), pp. 33–46

[15] G. Dantzig, All shortest routes in a graph, in *Theory of Graphs. International Symposium*, Gordon and Breach, New York, 1967, pp. 91–92

[16] G. B. Dantzig, On the shortest route through a network, *Management Sci.* **6** (1960), pp. 187–190

[17] R. Davis and A. Prieditis, The expected length of a shortest path, *Inform. Process. Lett.* **46** (1993), pp. 135–141

[18] R. Dial, F. Glover, D. Karney, and D. Klingman, A computational analysis of alternative algorithms and labeling techniques for finding shortest path trees, *Networks* **9** (1979), pp. 215–248

[19] M. Dietzfelbinger and W. Maass, Two lower bound arguments with "inaccessible" numbers, *J. Comput. System Sci.* **36** (1988), pp. 313–335

[20] E. W. Dijkstra, A note on two problems in connexion with graphs, *Numer. Math.* **1** (1959), pp. 269–271

[21] J. R. Driscoll, H. N. Gabow, R. Shrairman, and R. E. Tarjan, Relaxed heaps: An alternative to Fibonacci heaps with applications to parallel computation, *Comm. ACM* **31** (1988), pp. 1343–1354

[22] D. Dubhashi, V. Priebe, and D. Ranjan, Negative dependence through the FKG inequality, Research Report MPI-I-96-1-020, Max-Planck-Institut für Informatik, Saarbrücken, August 1996

[23] J. Edmonds and R. M. Karp, Theoretical improvements in algorithmic efficiency for network flow problems, *J. Assoc. Comput. Mach.* **19** (1972), pp. 248–264

[24] U. Feige, D. Peleg, P. Raghavan, and E. Upfal, Randomized broadcast in networks, *Random Structures Algorithms* **1** (1990), pp. 447–460

[25] R. W. Floyd, Algorithm 97: Shortest path, *Comm. ACM* **5** (1962), p. 345

[26] L. R. Ford, Jr. and D. R. Fulkerson, *Flows in Networks*, Princeton University Press, Princeton, NJ, 1962

[27] M. L. Fredman, New bounds on the complexity of the shortest path problem, *SIAM J. Comput.* **5** (1976), pp. 83–89

[28] M. L. Fredman and R. E. Tarjan, Fibonacci heaps and their uses in improved network optimization algorithms, *J. Assoc. Comput. Mach.* **34** (1987), pp. 596–615

[29] A. Frieze, Minimum paths in directed graphs, *Oper. Res. Quart.* **28** (1977), pp. 339–346

[30] A. Frieze and C. McDiarmid, Algorithmic theory of random graphs, *Random Structures Algorithms* **10** (1997), pp. 5–42

[31] A. M. Frieze and G. R. Grimmett, The shortest-path problem for graphs with random arc-lengths, *Discrete Appl. Math.* **10** (1985), pp. 57–77

[32] A. M. Frieze and B. Reed, Probabilistic analysis of algorithms, in M. Habib, C. McDiarmid, J. Ramirez-Alfonsin, and B. Reed (eds.), *Probabilistic Methods for Algorithmic Discrete Mathematics* (Algorithms Combin., vol. 16), Springer-Verlag, Berlin, 1998, pp. 36–92

[33] A. V. Goldberg, Scaling algorithms for the shortest paths problem, *SIAM J. Comput.* **24** (1995), pp. 494–504

[34] A. V. Goldberg, A simple shortest path algorithm with linear average time, Technical Report STAR-TR-01-03, InterTrust Technologies Corp., Santa Clara, CA, March 2001, to be presented at *ESA '01*

[35] A. V. Goldberg and R. E. Tarjan, Expected performance of Dijkstra's shortest path algorithm, Technical Report 96-062, NEC Research Institute, Princeton, June 1996

[36] R. L. Graham, D. E. Knuth, and O. Patashnik, *Concrete Mathematics: A Foundation for Computer Science*, 2nd ed., Addison-Wesley, Reading MA, 1994

[37] R. L. Graham, A. C. Yao, and F. F. Yao, Information bounds are weak in the shortest distance problem, *J. Assoc. Comput. Mach.* **27** (1980), pp. 428–444

[38] G. R. Grimmett and D. R. Stirzaker, *Probability and Random Processes*, 2nd ed., Oxford University Press, Oxford, 1992

[39] G. R. Grimmett and D. R. Stirzaker, *Probability and Random Processes: Problems and Solutions*, Oxford University Press, Oxford, 1992

[40] T. Hagerup, Improved shortest paths on the word RAM, in U. Montanari, J. D. P. Rolim, and E. Welzl (eds.), *Automata, Languages and Programming* (Lecture Notes in Comput. Sci., vol. 1853), Springer-Verlag, Berlin, 2000, pp. 61–72

[41] R. Hassin, A computing scheme for network problems with random edge lengths, in G. Andreatta, F. Mason, and P. Serafini (eds.), *Advanced School on Stochastics in Combinatorial Optimization*, World Scientific, Singapore, 1987, pp. 228–232

[42] R. Hassin and E. Zemel, On shortest paths in graphs with random weights, *Math. Oper. Res.* **10** (1985), pp. 557–564

[43] H. Imai and M. Iri, Practical efficiencies of existing shortest-path algorithms and a new bucket algorithm, *J. Oper. Res. Soc. Japan* **27** (1984), pp. 43–57

[44] M. Iri, How to generate realistic sample problems for network optimization, in T. Ibaraki, Y. Inagaki, K. Iwama, T. Nishizeki, and M. Yamashita (eds.), *Algorithms and Computation* (Lecture Notes in Comput. Sci., vol. 650), Springer-Verlag, Berlin, 1992, pp. 342–350

[45] S. Janson, One, two and three times $\log n/n$ for paths in a complete graph with random weights, *Combin. Probab. Comput.* **8** (1999), pp. 347–361

[46] K. Joag-Dev and F. Proschan, Negative association of random variables, with applications, *Ann. Statist.* **11** (1983), pp. 286–295

[47] D. B. Johnson, A note on Dijkstra's shortest path algorithm, *J. Assoc. Comput. Mach.* **20** (1973), pp. 385–388

[48] D. B. Johnson, Efficient algorithms for shortest paths in sparse networks, *J. Assoc. Comput. Mach.* **24** (1977), pp. 1–13

[49] D. R. Karger, D. Koller, and S. J. Phillips, Finding the hidden path: Time bounds for all-pairs shortest paths, *SIAM J. Comput.* **22** (1993), pp. 1199–1217

[50] R. M. Karp, personal communication, May 1996

[51] S. G. Kolliopoulos and C. Stein, Finding real-valued single-source shortest paths in $o(n^3)$ expected time, *J. Algorithms* **28** (1998), pp. 125–141

[52] E. L. Lawler, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York, 1976

[53] M. Li and P. Vitányi, *An Introduction to Kolmogorov Complexity and Its Applications*, 2nd ed., Springer-Verlag, New York, 1997

[54] M. Li and P. M. B. Vitányi, Average case complexity under the universal distribution equals worst-case complexity, *Inform. Process. Lett.* **42** (1992), pp. 145–149

[55] M. Luby and P. Ragde, A bidirectional shortest-path algorithm with good average-case behavior, *Algorithmica* **4** (1989), pp. 551–567

[56] C. McDiarmid, Concentration, in M. Habib, C. McDiarmid, J. Ramirez-Alfonsin, and B. Reed (eds.), *Probabilistic Methods for Algorithmic Discrete Mathematics* (Algorithms Combin., vol. 16), Springer-Verlag, Berlin, 1998, pp. 195–248

[57] C. C. McGeoch, All-pairs shortest paths and the essential subgraph, *Algorithmica* **13** (1995), pp. 426–441

[58] K. Mehlhorn and S. Näher, *LEDA. A Platform for Combinatorial and Geometric Computing*, Cambridge University Press, Cambridge, 1999

[59] K. Mehlhorn and V. Priebe, On the all-pairs shortest path algorithm of Moffat and Takaoka, in P. Spirakis (ed.), *Algorithms — ESA '95* (Lecture Notes in Comput. Sci., vol. 979), Springer-Verlag, Berlin, 1995, pp. 185–198

[60] K. Mehlhorn and V. Priebe, On the all-pairs shortest-path algorithm of Moffat and Takaoka, *Random Structures Algorithms* **10** (1997), pp. 205–220

[61] K. Mehlhorn, V. Priebe, G. Schäfer, and N. Sivadasan, All-pairs shortest-paths computation in the presence of negative cycles, *Inform. Process. Lett.* (2001), to appear

[62] M. Menduno, Atlas shrugged, *Sci. Amer.* **283**:5 (November 2000), pp. 16–17

[63] U. Meyer, Single-source shortest-paths on arbitrary directed graphs in linear average-case time, in *Proc. 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2001, pp. 797–806

[64] A. Moffat and T. Takaoka, An all pairs shortest path algorithm with expected running time $O(n^2 \log n)$, in *Proc. 26th Annual Symposium on Foundations of Computer Science*, 1985, pp. 101–105

[65] A. Moffat and T. Takaoka, An all pairs shortest path algorithm with expected time $O(n^2 \log n)$, *SIAM J. Comput.* **16** (1987), pp. 1023–1031

[66] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, Cambridge, 1995

[67] G. L. Nemhauser, A generalized permanent label setting algorithm for the shortest path between specified nodes, *J. Math. Anal. Appl.* **38** (1972), pp. 328–334

[68] K. Noshita, A theorem on the expected complexity of Dijkstra's shortest path algorithm, *J. Algorithms* **6** (1985), pp. 400–408

[69] I. Palásti, On the strong connectedness of directed random graphs, *Studia Sci. Math. Hungar.* **1** (1966), pp. 205–214

[70] U. Pape, Implementation and efficiency of Moore-algorithms for the shortest route problem, *Math. Programming* **7** (1974), pp. 212–222

[71] U. Pape, Algorithm 562: Shortest path lengths, *ACM Trans. Math. Software* **6** (1980), pp. 450–455

[72] U. Pape, Remark on Algorithm 562, *ACM Trans. Math. Software* **9** (1983), p. 260

[73] B. Pittel, On spreading a rumor, *SIAM J. Appl. Math.* **47** (1987), pp. 213–223

[74] R. Raman, The power of collision: Randomized parallel algorithms for chaining and integer sorting, in K. V. Nori and C. E. Veni Madhavan (eds.), *Foundations of Software Technology and Theoretical Computer Science* (Lecture Notes in Comput. Sci., vol. 472), Springer-Verlag, Berlin, 1990, pp. 161–175

[75] R. Raman, Recent results on the single-source shortest paths problem, *ACM SIGACT News* **28**:2 (June 1997), pp. 81–87

[76] D. R. Shier and C. Witzgall, Properties of labeling methods for determining shortest path trees, *J. Res. Nat. Bur. Stand.* **86** (1981), pp. 317–330

[77] T. L. Snyder and J. M. Steele, Probabilistic networks and network algorithms, in M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser (eds.), *Network Models* (Handbooks Oper. Res. Management Sci., vol. 7), Elsevier, Amsterdam, 1995, pp. 401–424

[78] P. M. Spira, A new algorithm for finding all shortest paths in a graph of positive arcs in average time $O(n^2 \log^2 n)$, *SIAM J. Comput.* **2** (1973), pp. 28–32

[79] C. R. Subramanian, A generalization of Janson inequalities and its application to finding shortest paths, in *Proc. 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1999, pp. 795–804

[80] T. Takaoka, A new upper bound on the complexity of the all pairs shortest path problem, *Inform. Process. Lett.* **43** (1992), pp. 195–199

[81] T. Takaoka, Theory of 2-3 heaps, in T. Asano, H. Imai, D. T. Lee, S.-i. Nakano, and T. Tokuyama (eds.), *Computing and Combinatorics* (Lecture Notes in Comput. Sci., vol. 1627), Springer-Verlag, Berlin, 1999, pp. 41–50

[82] M. Thorup, Undirected single-source shortest paths with positive integer weights in linear time, *J. Assoc. Comput. Mach.* **46** (1999), pp. 362–394

[83] M. Thorup, Floats, integers, and single source shortest paths, *J. Algorithms* **35** (2000), pp. 189–201

[84] N. Tomizawa, On some techniques useful for solution of transportation network problems, *Networks* **1** (1971/72), pp. 173–194

[85] S. K. Walley and H. H. Tan, Shortest paths in random weighted graphs, in D.-Z. Du and M. Li (eds.), *Computing and Combinatorics* (Lecture Notes in Comput. Sci., vol. 959), Springer-Verlag, Berlin, 1995, pp. 213–222

[86] S. K. Walley, H. H. Tan, and A. M. Viterbi, Shortest path cost distribution in random graphs with positive integer edge costs, in *Proc. 12th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'93)*, 1993, pp. 1023–1032

[87] S. K. Walley, H. H. Tan, and A. M. Viterbi, Limit distribution for the diameter and the shortest path hop count in random graphs with positive integer edge cost, in *Proc. 13th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'94)*, 1994, pp. 618–627

[88] A. C. Yao, D. M. Avis, and R. L. Rivest, An $\Omega(n^2 \log n)$ lower bound to the shortest path problem, in *Proc. 9th Annual ACM Symposium on Theory of Computing*, 1977, pp. 11–17

[89] F. B. Zhan and C. E. Noon, Shortest path algorithms: An evaluation using real road networks, *Transp. Sci.* **32** (1998), pp. 65–73

[90] U. Zwick, All pairs shortest paths in weighted directed graphs—exact and almost exact algorithms, in *Proc. 39th Annual Symposium on Foundations of Computer Science*, 1998, pp. 310–319

[91] U. Zwick, All pairs lightest shortest paths, in *Proc. 31st Annual ACM Symposium on Theory of Computing*, 1999, pp. 61–69

[92] U. Zwick, All pairs shortest paths using bridging sets and rectangular matrix multiplication, Report TR00-060, Electronic Colloquium on Computational Complexity, `http://www.eccc.uni-trier.de/eccc/`, August 2000

Zusammenfassung

In dieser Arbeit wird die *average-case*-Komplexität von Kürzeste-Wege-Algorithmen untersucht. Eingaben für einen Algorithmus werden dabei gemäß einer Wahrscheinlichkeitsverteilung auf der Menge aller möglichen Eingaben erzeugt. Im Fall von Kürzeste-Wege-Algorithmen bedeutet dies: Wir betrachten Wahrscheinlichkeitsverteilungen auf der Menge der Netzwerke (mit einer beliebigen, aber festen Anzahl n von Knoten), also der Menge der gerichteten Graphen mit Kantenkosten.

Wir zeigen in dieser Arbeit, dass einige Kürzeste-Wege-Probleme auf dem überwiegenden Teil der so erzeugten Netzwerke schneller gelöst werden können als dies durch die besten bekannten *worst-case*-Laufzeiten vorhergesagt wird. Dabei nutzen wir in allen Fällen aus, dass uns die jeweiligen Eingabemodelle erlauben, die Kürzeste-Wege-Probleme korrekt auch dann zu lösen, wenn wir nur einen Teil der Kantenmenge der Netzwerke betrachten.

Zur Lösung von Kürzeste-Wege-Problemen sind all diejenigen Kanten ohne Bedeutung, deren Kosten größer als die Kosten des teuersten aller kürzesten Wege, das heißt, größer als der Durchmesser des Netzwerkes, sind. Diese Beobachtung kann zum Beispiel im *uniform model* ausgenutzt werden: Die Netzwerke sind in diesem Modell vollständige Graphen, deren nicht negative Kantenkosten gemäß einer fixen Verteilungsfunktion F in unabhängigen Zufallsexperimenten bestimmt werden. Das folgende Vorgehen wurde schon von früheren Autoren vorgeschlagen: Wird ein Netzwerk gemäß des *uniform model* erzeugt, so ist (unter schwachen zusätzlichen Voraussetzungen an F) mit hoher Wahrscheinlichkeit der Durchmesser $O((\log n)/n)$, und nur $O(n \log n)$ Kanten haben kleinere Kosten. Auf dem Netzwerk mit entsprechend reduzierter Kantenmenge können somit die Distanzen zwischen allen Paaren von Knoten (das *all-pairs shortest-paths problem*) mit hoher Wahrscheinlichkeit in Zeit $O(n^2 \log n)$ berechnet werden. Wir weisen nach, dass ein analoger Algorithmus das *all-pairs shortest-paths problem* in derselben Zeit auch dann lösen kann, wenn Eingaben gemäß des *extended uniform model* generiert werden, in dem jede Kante, unabhängig von allen anderen Kanten, mit Wahrscheinlichkeit p existiert und den existierenden Kanten Kosten wie im *uniform model* zugewiesen werden. Nur $O(n \log n)$ Kanten, deren Kosten kleiner als $O((\log n)/(np))$ sind, sind dann für die Berechnung der Distanzen von Bedeutung. Wir erweitern außerdem das Wissen um die typische Struktur der Netzwerke in beiden Modellen, indem wir zeigen, dass alle kürzesten

Wege mit hoher Wahrscheinlichkeit aus $O(\log n)$ Kanten bestehen.

Will man die *average-case*-Komplexität von Algorithmen untersuchen, die Kürzeste-Wege-Probleme auf Netzwerken mit beliebigen reellen Kantenkosten lösen, so ist eine sinnvolle Analyse mit den beiden bislang erwähnten Modellen nicht möglich. Denn wenn die Verteilungsfunktion F auch negative Kantenkosten erlaubt, so enthalten die erzeugten Netzwerke mit hoher Wahrscheinlichkeit negative Kreise. Wir analysieren das *vertex-potential model*, in dem eine Kante (v, w) Kosten $r(v, w) - \pi(v) + \pi(w)$ hat, wobei $r(v, w)$ wie im *uniform model* erzeugt wird, die Knotenpotentiale $\pi(v), \pi(w)$ jedoch beliebig gewählt werden können. Es folgt aus dieser Definition, dass Netzwerke, die gemäß des *vertex-potential model* erzeugt werden, überhaupt keine negativen Kreise haben. Wir können bei diesen Netzwerken die Kantenmenge so reduzieren, dass der Algorithmus von Bellman–Ford die Distanzen aller Knoten von einem fixen Ausgangsknoten (das *single-source shortest-paths problem*) mit hoher Wahrscheinlichkeit in Zeit $O(n^2)$ bestimmt; hier geht auch unser Resultat über die Anzahl von Kanten auf kürzesten Wegen ein. Diese Laufzeit impliziert, dass das *all-pairs shortest-paths problem* mit hoher Wahrscheinlichkeit in Zeit $O(n^2 \log n)$ lösbar ist.

Im *endpoint-independent model* wird eine Menge nicht negativer Kantenkosten beliebig gewählt; zufällig ist hier allein die Zuordnung der Kosten zu bestimmten Kanten. Das *endpoint-independent model* erlaubt demnach kein Reduktions-Argument vom in den beiden vorigen Abschnitten beschriebenen Typ. Ein schon auf Dantzig zurückgehender Algorithmus zur Lösung eines *single-source shortest-paths problem* mit nicht negativen Kantenkosten relaxiert bei Knoten, deren Distanz schon berechnet wurde, die ausgehenden Kanten inkrementell in der Ordnung steigender Kosten. Es zeigt sich, dass dieser Algorithmus unter den Annahmen des Modells mit hoher Wahrscheinlichkeit nach Relaxierung von nur $O(n \log n)$ vielen Kanten allen Knoten deren korrekte Distanz zugewiesen hat. Der nötige Sortieraufwand lässt sich jedoch erst in der Laufzeit $O(n^2 \log n)$ für das *all-pairs shortest-paths problem* subsumieren. Die Laufzeit dieses Algorithmus ist schon früher analysiert worden; wir glauben aber, dass unsere Analyse die erste korrekte ist.

Wir zeigen schließlich, dass eine weitere Reduktion der zur Lösung eines *single-source shortest-paths problem* betrachteten Kantenmenge nicht möglich ist. Wir beweisen, dass auf einer Klasse von Netzwerken mit ganzzahligen Kantenkosten jeder Algorithmus mit hoher Wahrscheinlichkeit $\Omega(n \log n)$ Kanten inspizieren muss, um die korrekten Distanzen zu berechnen.