Directed Degree Sequences

Dissertation zur Erlangung des akademischen Grades
doctor rerum naturalium (Dr. rer. nat.)

Martin-Luther-Universität Halle-Wittenberg
Naturwissenschaftliche Fakultät III
Agrar- und Ernährungswissenschaften, Geowissenschaften und Informatik
(Institut für Informatik)

eingereicht von M. Sc. Annabell Berger
geb. am 27. Mai 1974 in Jena

Halle (Saale), 10. Oktober 2011

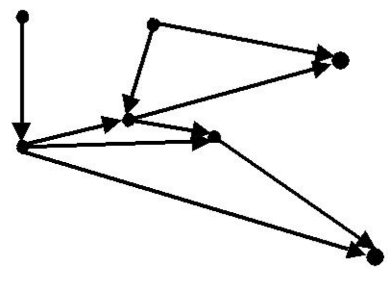
1. Reviewer: Prof. Dr. Matthias Müller-Hannemann
2. Reviewer: Prof. Dr. Dieter Rautenbach

Day of the defense: December, 12th 2011

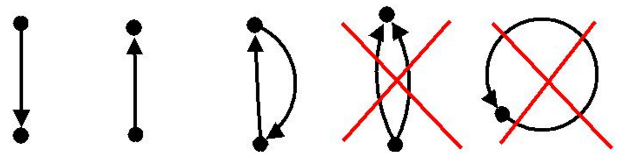
Ein kleiner Einblick in diese Doktorarbeit für interessierte Laien

"Wohin Du auch gehst, geh mit deinem ganzen Herzen." (Konfuzius)

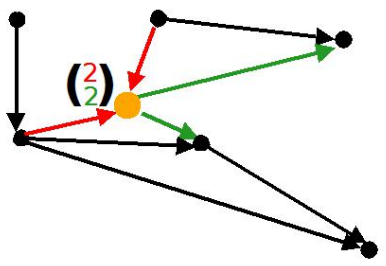
Meine Arbeit beschäftigt sich mit mathematischen Objekten, die man *Digraphen* nennt. Hier habe ich einen Digraphen gemalt.



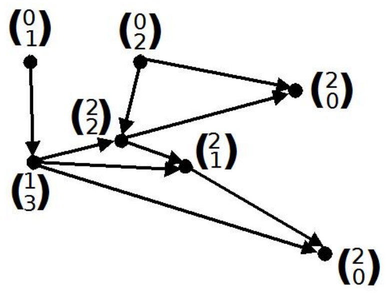
Er besteht aus sogenannten *Knoten* und *Pfeilen.* Die Knoten sind die kleinen schwarzen Kreise, die durch Pfeile miteinander verbunden werden können. Dabei müssen bestimmte Regeln eingehalten werden. Zwei Knoten dürfen mit höchstens einem Pfeil in einer Richtung miteinander verbunden werden. Somit ist es zwar erlaubt, dass zwei entgegengesetzt gerichtete Pfeile zwischen zwei Knoten existieren. Jedoch sind in die gleiche Richtung zeigende Pfeile verboten. Hier sehen Sie ein Bild von allen erlaubten und nicht erlaubten Möglichkeiten, Pfeile zu zeichnen.



Zählen Sie nun die Anzahl der eingehenden Pfeile und die Anzahl der ausgehenden Pfeile an einem bestimmten Knoten in unserem Digraphen.

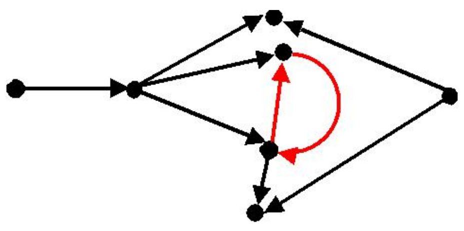


In unserer Zeichnung enden zwei Pfeile in dem gelb markierten Knoten mit ihrer Pfeilspitze und zwei Pfeile beginnen dort. Wir ordnen diesem Knoten ein Zahlenpaar $\binom{2}{2}$ zu. Nun bestimmen wir das jeweils passende Zahlenpaar für jeden anderen Knoten in unserem Digraphen.



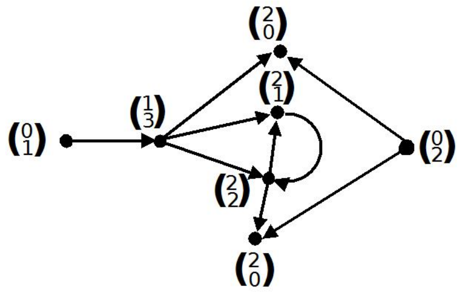
Wir erhalten eine Folge von Zahlenpaaren. Für unseren Digraphen ist das die Folge $\binom{2}{2}, \binom{0}{1}, \binom{1}{3}, \binom{0}{2}, \binom{2}{0}, \binom{2}{1}, \binom{2}{0}$. Damit wir uns besser über solche Folgen unterhalten können, geben wir ihnen einen besonderen Namen. Wir bezeichnen sie als *Digraph-Folgen*. In meiner Arbeit beschäftige ich mich mit Digraph-Folgen, aber interessanter Weise muss ich das Problem genau umgekehrt betrachten. Damit meine ich, man gibt mir eine beliebige Digraph-Folge und ich muss herausfinden, ob es einen passenden

Digraphen gibt. Wenn wir also die Digraph-Folge $\binom{2}{2}, \binom{0}{1}, \binom{1}{3}, \binom{0}{4}, \binom{2}{0}, \binom{2}{1}, \binom{2}{0}$ betrachten, können wir schon einen passenden Digraphen zeichnen. Die Frage ist aber, gibt es ein Rezept, mit dessen Hilfe für jede Digraph-Folge ein passender Digraph konstruiert werden kann? Die Antwort kann ich bejahen und das ist schon seit 30 Jahren bekannt. Etwas schwieriger wird es, wenn weitere Anforderungen gestellt werden, die in dem Digraphen erfüllt sein sollen. Man könnte beispielsweise fragen, ob es für eine Digraph-Folge einen Digraphen gibt, der keine Kreise enthält. Dazu sollte ich erklären, was das Wort *Kreis* in einem Digraphen überhaupt bedeutet. Vielleicht sind sie schon einmal bei einer Wanderung "im Kreis gelaufen" und zu ihrem Ausgangspunkt zurückgekehrt, weil sie sich verlaufen haben? Ganz ähnlich verhält es sich mit Kreisen auf einem Digraphen. Stellen Sie sich vor, sie könnten auf einem Digraphen von Knoten zu Knoten spazieren gehen. Dabei dürfen sie immer auf den Pfeilen in ihrer natürlichen Richtung laufen. Die Pfeile sind sozusagen Einbahnstraßen. Wenn es möglich ist, von einem Knoten wegzulaufen, etwas herum zu spazieren und am Ende wieder am Startknoten zu landen, dann sind Sie im Kreis gelaufen. In unserem Digraphen oben gibt es keinen Kreis, der gelaufen werden könnte. Deswegen zeichne ich Ihnen ein Beispiel, wo ein solcher Kreis (rot hervorgehoben) vorhanden ist.



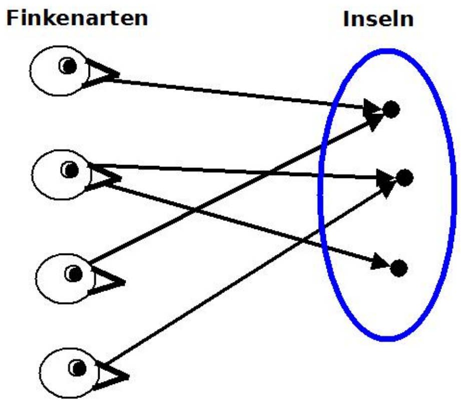
In meiner Arbeit habe ich versucht, eine Lösung zu finden für die folgende Aufgabe: Gegeben sei eine Digraph-Folge. Kann ich ein Rezept finden, was mir einen passenden Digraphen **ohne Kreise** konstruiert oder mir mit hundertprozentiger Sicherheit sagt, dass es keinen Digraphen ohne Kreise für diese Folge gibt? Ich konnte diese Aufgabe zwar für jede beliebige Digraph-Folge lösen, allerdings war ich lange Zeit nicht vollständig zufrieden mit dieser Lösung. Der Grund besteht darin, dass es Digraph-Folgen gibt, bei der meine vorgeschlagene Konstruktion für den Bau eines passenden Digraphen in einigen Fällen zu lange dauert. Bis kurz vor Abgabe meiner Arbeit wusste ich nicht, ob eine bessere Lösung existiert und ich diese nur nicht finden kann.

André Nichterlein (Nic11), ein Mathematiker aus Berlin, hat diese Frage vor kurzem beantworten können. Er zeigte, dass es sich um ein “schweres” Problem handelt. Viele Mathematiker und Informatiker gehen heute davon aus, dass so geartete Probleme keine effiziente Lösung für alle auftretenden Fälle besitzen. Für eine bestimmte Art von Digraph-Folgen konnte ich jedoch eine sehr elegante, schöne Lösung finden, die sehr leicht umgesetzt werden kann. Kehren wir zu unserem Beispiel zurück. Bestimmen wir die Digraph-Folge des zuletzt gezeichneten Digraphen.



Es handelt sich um exakt die gleiche Digraph-Folge wie für den Digraphen in unserem ersten Beispiel. Klar ist aber auch, dass es sich hierbei um zwei verschiedene Digraphen handeln muss, denn im ersten konnten wir keinen Kreis finden. An dieser Stelle möchte ich kurz den Gedanken unterbrechen und erklären, wie ich überhaupt zu solchen Fragen kam, die zugegebenermassen zuerst einmal wie eine sehr verspielte Knobelei aussehen.
Die Verbindung zur Wirklichkeit findet sich in dem Objekt des Digraphen. Stellen Sie sich die Knoten als Bahnhöfe vor und die Pfeile als Bahnstrecken zwischen zwei Bahnhöfen, dann erhalten wir ein Bahnnetz. Oder denken Sie sich die Knoten als Menschen und einen Pfeil zwischen zwei Menschen, wenn einer der beiden die Meinung vertritt, er sei mit dem anderen befreundet. Natürlich ist in einem solchen Digraphen der Pfeil meistens auch in der anderen Richtung vorhanden. Weiterhin könnte man auch die verschiedenen im Körper ablaufenden chemischen Reaktionen, die teilweise voneinander abhängen, in einem Digraphen darstellen. Betrachten wir mal ein ganz anderes, durch das Darwinjahr wieder mehr ins Bewusstsein getretene Thema. Verschiedene Finkenarten auf den Galapagosinseln. Genau genommen gibt es 17 Galapagosinseln und 13 verschiedene Finkenarten, die auf diesen Inseln leben. Wir ordnen jeder Finkenart genau einen Knoten und jeder Insel genau einen Knoten zu und verbinden eine Fin-

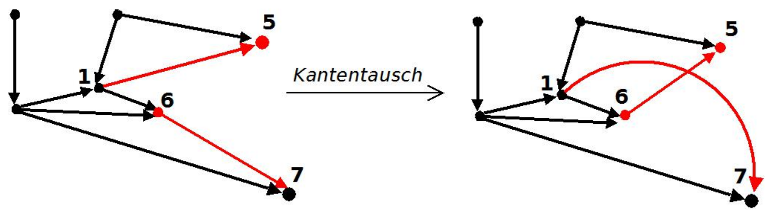
kenart durch einen Pfeil genau dann mit einer Insel, wenn diese Finkenart auf dieser Insel lebt. Dann erhalten wir einen Digraphen mit einer zugehörigen Digraph-Folge. Hier zeichne ich nur ein schematisches Beispiel für den Digraphen, da das Original zu groß wäre.



Solche Digraphen aus der realen Welt werden heute zu verschiedenen Zwecken von verschiedenen Wissenschaftlern in Physik, Informatik, Biologie, Soziologie und Chemie untersucht. Man möchte mehr über die Struktur dieser Digraphen erfahren, um komplexe Zusammenhänge besser verstehen zu lernen. Der Wunsch dieser Wissenschaftler besteht darin, ganz besondere Auffälligkeiten in den Digraphen zu finden. Es stellt sich die Frage, wie identifiziert man eigentlich "Extravaganzen". Das ist im Allgemeinen nur möglich, wenn man Kenntnis von einer Art *Normalität* besitzt, mit der man vergleichen und zu der man eine besondere Abweichung feststellen kann. Solche Normalitäten könnten bei Digraphen andere, sozusagen zufällig gewürfelte Digraphen, zu exakt der gleichen Digraph-Folge sein. Stellen Sie sich beispielsweise vor, Sie stellen fest, dass der Digraph der Darwinfinken keinen Kreis enthält. Ein ungenauer Betrachter könnte das für eine Auffälligkeit halten. Beschäftigt man sich jedoch näher mit der zugehörigen Digraph-Folge, dann kann man leicht feststellen, dass kein passender Digraph einen Kreis enthalten kann. Die Erklärung dieses Phänomens liegt in der Tatsache, dass jedes Zahlenpaar der zugehörigen Digraph-Folge $\binom{0}{1}, \binom{0}{2}, \binom{0}{1}, \binom{0}{1}, \binom{2}{0}, \binom{2}{0}, \binom{1}{0}$ eine Null enthält. Das bedeutet, ein passender Digraph enthält entweder nur Knoten mit eingehenden Pfeilen oder nur Knoten mit ausgehenden Pfeilen. Knoten in einem Kreis müssen jedoch mindestens einen eingehenden Pfeil und einen ausgehenden Pfeil enthalten, denn sonst kann man den Knoten nicht wieder verlassen. Die Kreisfreiheit liegt sozusagen in der “Natur” der Digraph-Folge der Darwinfinken. Würde man diese Eigenschaft nicht so einfach herausfinden können, wäre aber in der Lage 100 weitere Digraphen der Digraph-Folge der Darwinfinken möglichst zufällig zu erzeugen, dann würde man bei jedem dieser Digraphen feststellen, dass er keinen Kreis enthält und schlussfolgern, dass es sich nicht um eine Besonderheit beim Digraphen der Darwinfinken handeln kann. Genau mit dieser Frage:

“Wie erzeugt man weitere Digraphen für eine Digraph-Folge – und zwar so, dass jeder Digraph mit der gleichen Wahrscheinlichkeit konstruiert wird?”

beschäftigt sich meine Arbeit auch. Sie werden vielleicht einsehen, dass es nicht immer möglich ist, alle Digraphen zu malen und einen zufällig auszuwählen. Häufig gibt es einfach zu viele und selbst “die beste Rechentechnik der Welt” kann das ab einer bestimmten Größe der Digraphen nicht mehr leisten. Digraphen aus der realen Welt haben oft Tausende, manchmal gar Millionen von Knoten. Man bedient sich hier einer sehr schönen, einfachen Idee. Man nimmt in einem Digraphen sehr kleine Veränderungen vor und achtet dabei darauf, dass die Digraph-Folge erhalten bleibt. Man kann z.B. die Pfeilenden zweier nicht benachbarter Pfeile miteinander vertauschen, falls das nicht unseren Regeln – wie man Pfeile malen darf – widerspricht. In unserem Beipielbild können Sie die Enden der Pfeile von Knoten 1 zu 5 und 6 zu 7 miteinander vertauschen. Wir erhalten die neuen Pfeile von Knoten 1 zu 7 und 6 zu 5.



Das besondere an solch einem Tausch ist, dass die Anzahl der eingehenden Pfeile und der ausgehenden Pfeile an keinem Knoten verändert werden. Somit erhalten wir einen neuen Digraphen zur gleichen Digraph-Folge. Die Pfeilenden der Pfeile von Knoten 3 nach 7 und von Knoten 1 nach 6 dürfen wir nicht tauschen, da ein neuer Pfeil von 3 nach 6 entstehen würde, welcher aber schon vorhanden ist. Nun könnte man einen solchen Tausch der Pfeilenden sehr oft wiederholen. Es stellt sich heraus, dass man –

unter Beachtung einiger, kleiner Regeln – bei sehr vielen Digraph-Folgen auf diese Art alle Digraphen erzeugen kann. Allerdings – und das ist der Haken an der Sache – müsste man diese Pfeilendenvertauschungen unendlich oft wiederholen, wenn man sicher sein möchte, dass jeder Digraph mit der gleichen Wahrscheinlichkeit erzeugt wird, was ja die eigentliche Aufgabe gewesen ist. Ich habe in meiner Arbeit bewiesen, für welche Typen von Digraph-Folgen dieses Ergebnis stimmt und ich habe auch herausgefunden, welche weiteren kleinen Veränderungen in Digraphen, für die das Pfeilendentauschen nicht ausreicht, vorgenommen werden müssen. Was ich leider nicht beweisen konnte, ist, ob es möglich ist, nach relativ kurzer Anzahl der Vertauschungen aufzuhören, weil man weiß, dass man nur winzige Abweichungen vom gewünschten Ergebnis messen würde. Manchmal kann man das bei anderen Problemen tatsächlich zeigen. Eines dieser Probleme kennt jeder. Es handelt sich um das Kartenmischen beim Kartenspiel. Vor einiger Zeit haben Wissenschaftler ausgerechnet, wie of man "Riffeln" muss, um einen wirklich gut durchmischten Kartenstapel von 52 Karten zu erhalten. Riffeln bedeutet, man hebt einen Stapel ab und lässt die Karten beider Stapel einigermassen abwechselnd zu einem neuen Stapel fließen. Man sagt, 7 Mal sollte man nacheinander mindestens riffeln, mehr als 12 Mal braucht man nicht und es wurde gezeigt, dass sechs Mal nicht ausreichen (AZ04). Die Analogie zu unserem Problem besteht darin: Einmal Riffeln entspricht einmal Pfeilendentauschen in unserem Digraphen. Wir verändern jedes Mal den aktuellen Zustand des Kartenstapels bzw. des Digraphen ein wenig. Beim Kartenmischen kann man beweisen, dass man nicht unendlich lange Riffeln muss, um gut zu mischen. Beim unserem Problem ist es ein offene Frage, die noch beantwortet werden muss. In der praktischen Anwendung geht es beim zufälligen Erzeugen eines Digraphen häufig ähnlich zu, wie zu Hause beim Kartenspiel. Man riffelt einige Male und wenn man den Eindruck hat, der Stapel ist gut gemischt, hört man auf. Überprüfen wird es kaum Einer.

Acknowledgements

Meinem Doktorvater Matthias: Unsere Diskussionen sind zu einem erschreckend wichtigen Ritual geworden. Danke für diesen produktiven Spaß!
Ivo Hedtke: Du hast mir den $\LaTeX$-Himmel gezeigt. Auch wenn ich nicht weiß, ob ich dort jemals hinkomme. Danke!
Jessica Jacobs: Many thanks for your helpful “native speaker comments”.

Contents

List of Figures

1

Degree Sequences – an introduction to a classic topic in theory as well as in practice

The realization of graphs and digraphs with prescribed degree sequences has attracted researchers for several decades tracing back to Havel, Hakimi, Erdős, Gallai, Gale, Ryser, Fulkerson, Chen, Kleitman and Wang (Hav55, Hak62, EG60, Gal57, Rys57, Ful60, Che66, KW73). Clearly, this problem is a classical graph theoretic problem. On a second view it is much more than a problem in graph theory because it has been considered in several different scientific communities with completely different notions such that many results were reinvented repeatedly until today. This shows the relevance of these problems in theory as well as in practice. We started to deal with several arising new problems in this context as a physicist asked for a solution of a "little problem". He works in the field of network analysis and wanted a random solution for the following problem:

Problem 1.1 (dag realization problem)**.** *Given is a finite sequence $S := \binom{a_1}{b_1}, \ldots, \binom{a_n}{b_n}$ with $a_i, b_i \in \mathbb{Z}_0^+$. Does there exist an acyclic digraph (without parallel arcs) $G = (V, A)$ with the labeled vertex set $V := \{v_1, \ldots, v_n\}$ such that we have indegree $d_G^-(v_i) = a_i$ and outdegree $d_G^+(v_i) = b_i$ for all $v_i \in V$?*

If the answer is "yes", we call sequence S *dag sequence* and the acyclic digraph G (a so-called "dag") a *dag realization.* Consider for example the dag realization G in Figure 1.1 for the given dag sequence $\binom{0}{2}, \binom{0}{1}, \binom{1}{3}, \binom{2}{2}, \binom{2}{1}, \binom{2}{0}, \binom{2}{0}$.

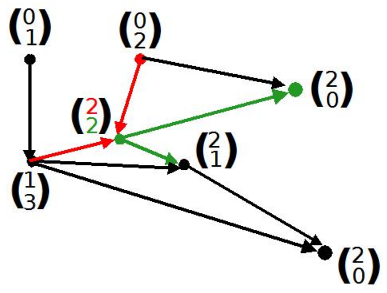


Figure 1.1: A possible dag realization G for sequence S.

Unless explicitly stated, we assume that a sequence does not contain any *zero tuples* $\binom{0}{0}$. Moreover, we will tacitly assume that $\sum_{i=1}^{n} a_i = \sum_{i=1}^{n} b_i$, as this is obviously a necessary condition for any realization to exist, since the number of ingoing arcs must equal the number of outgoing arcs. Furthermore, we denote tuples $\binom{a_i}{b_i}$ with $a_i > 0$ and $b_i = 0$ as *sink tuples*, those with $a_i = 0$ and $b_i > 0$ as *source tuples*, and the remaining ones with $a_i > 0$ and $b_i > 0$ as *stream tuples.*

The physicist did not only want one arbitrary solution of the dag realization problem. Instead, he additionally demanded a randomly picked solution from the set of **all** dag realizations taken with the same probability. To the best of our knowledge there is no previous work dealing with this problem. Therefore, we started to study the problem in two different ways:

1. Is a given dag sequence realizable? How can we get a dag realization?

2. What is known about the so-called *uniform sampling of graph or digraph realizations*?

Let us first consider question 1.

1.1 Dag Sequences in Brief

Very recently Nichterlein (Nic11) proved the NP-completeness of the dag realization problem. Let us focus on related problems, i. e. , a relaxation which is solvable in polynomial running time and an NP-complete generalization. The *digraph realization problem* is a relaxed version of the dag realization problem without the condition of acyclicity. The corresponding sequence of a "yes"-instance we denote as *digraph sequence* and the digraph G is called a *digraph realization.* This problem can either be solved in polynomial running time using a recursive algorithm (KW73) or by a complete characterization (Gal57, Rys57, Ful60, Che66) leading to n inequalities which have to be checked. Now, we consider a generalization of our problem.

Problem 1.2 (f-factor dag problem)**.** *Given is a sequence* $S := \binom{a_1}{b_1}, \ldots, \binom{a_n}{b_n}$ *and a digraph* $G' = (V, A)$ *with a labeled vertex set* $V := \{v_1, \ldots, v_n\}$*. Does there exist an acyclic subdigraph* $G = (V, A')$ *of* G' *with* $d^-_G(v_i) = a_i$ *and* $d^+_G(v_i) = b_i$ *for all* $v_i \in V$*?*

The difference to our dag realization problem is, that there the digraph G' is always a complete digraph. The hardness of this generalization can easily be seen by considering sequence $S := \binom{0}{1}, \binom{1}{1}, \ldots, \binom{1}{1}, \binom{1}{0}$. It possesses only one unique dag realization (up to isomorphy), namely a Hamiltonian path. Hence, a polynomial time algorithm solving our f-factor dag realization problem, could find in the same way a directed Hamiltonian path on digraph G'.

Theorem 1.1 (f-factor dag problem)**.** *The f-factor dag problem is NP-complete.*

Again, a relaxation of the f-factor dag problem to the *f-factor problem* – by omitting the property of acyclicity – leads to a polynomial time algorithm (Tut81). We solved the dag realization problem for special classes of sequences, made several experiments for unsolved classes and now as a result we can give important structural insights. In the following, we give an overview about our scientific results for question 1.

a) We show how to solve the dag realization problem for an important class of sequences – so called *opposed sequences* – in linear time. For that, we give a recursive algorithm (Section 4.4).

b) We give a complete characterization for a special subclass of opposed sequences –*strongly opposed sequences.* This means we can show that it is sufficient to check n inequalities, where n is the number of tuples of a given sequence S. It turns out that we can apply in the case of strongly opposed sequence exactly the same classical characterizations as for general digraph sequences. Hence, a strongly opposed sequence (with some further necessary conditions) is a dag sequence if and only if it can also be realized as a digraph.

c) We have a general result for all sequences which significantly improves upon a straightforward exponential time algorithm but which may still run in exponential time (Subsection 4.1.2).

d) We had a conjecture (when the complexity of the dag realization problem was still open) that a combination of the general algorithm in c) and a certain strategy leads to a polynomial time algorithm for all sequences. We disproved our conjecture by constructing a counter-example. On the other hand, we show in experiments that a large fraction of systematically constructed dag sequences can be efficiently solved by this strategy. Another striking observation is that this simple disproved linear-time algorithm solves a set of real-world instances from different domains, namely ordered binary decision diagrams (OBDDs), train and flight schedules, as well as instances derived from food-web networks **without any exception**. The vertex degrees of OBBDs always correspond to opposed sequences. Likewise, we observe that the dag sequences corresponding to the train and flight schedules are opposed as well. Hence, we have a linear-time algorithm for their corresponding sequences with a). To explore possible reasons for the observation that so many non-opposed sequences are solved successfully by our strategy we started several experiments leading to interesting insights. This motivates us to develop characteristics like dag density and “distance to provably easy sequences” which can indicate how easy or difficult a given sequence can be realized.

e) We propose a randomized algorithm which exploits our structural insight on topological sortings and uses a number of reduction rules. We compare this algorithm with other straightforward randomized algorithms in extensive experiments. We observe that it clearly outperforms all other variants and behaves surprisingly well for almost all instances.

f) We summarize our insights of d) and e) in a “strategy for practitioners”. In systematic experiments we show that a combination of the lexmax strategy and our randomized algorithm can efficiently solve the dag realization problem for almost all sequences.

1.2 Uniformly Sampling Digraph Realizations in Brief

Let us now consider the problem of uniformly sampling graph or digraph realizations. It also remains open in a practical sense. In a break-through paper (Fulkerson Prize 2006), Jerrum, Sinclair, and Vigoda (JSV04) presented a polynomial-time approximation scheme for a famous problem, i.e., determining the permanent (the number of perfect matchings in bipartite graphs) which is known to be $\sharp P$-hard (Val79). A part of their solution is to sample perfect matchings uniformly. The connection to our digraph realization problem can be seen by applying the following steps.

1. Reduce it to a directed f-factor problem on an underlying complete digraph.
2. Transform it into an undirected f-factor problem on an undirected bipartite graph.
3. Modify it to a perfect matching problem on a bipartite graph via a construction by Tutte (replace each vertex by a certain bipartite graph), see (Tut52).

Hence, their approach can be used to sample arbitrary bipartite graphs and arbitrary digraphs with a given (di)graph sequence in $O(n^{14}\log^4 n)$ via the above-mentioned reduction which is far from practical. Bezakova et al. (BBV07) improve these upper bounds to $O(n^{11}\log^5 n)$. The reduction to a perfect matching problem is different to all other natural approaches which are extensively used by scientists in physics, biology or chemistry. The reason may be that this reduction is not only too slow for practical use but also quite complicated. It is a very impressive work to have proven the complexity status of this problem, but it is not something that one would implement. However, it is the only known approach with provable polynomial running time. Further popular variants are the *configuration model (also called pairing model)* and the *switching algorithm* – mostly formulated for undirected graphs – for which we give more details and discuss recent work in the following chapters. Here, we focus on the switching algorithm which is known for the uniform sampling problem of graph realizations. The idea is as follows: It starts with a given graph realization and replaces two non-adjacent edges $\{a,b\},\{c,d\}$ either by $\{a,c\},\{b,d\}$ or by $\{a,d\},\{b,c\}$, provided that both new edges have not been contained in the current graph realization before this so-called *swap operation.* In the latter case one has to maintain the actual graph realization. Note, that the degrees of vertices a, b, c and d remain unchanged during this procedure. Hence, the graph sequence does not change. It was proven (via Markov chains, more details follow in the next chapters) that there exists a sequence of swap operations between each pair of different graph realizations for one given sequence. If one numbers the steps of swap operations from 1 to t, one can prove that the distribution of all graph realizations tends for $t \to \infty$ to the uniform distribution. To get this result, it is very important to choose each pair of non-adjacent edges with the same probability and also to count the swap operations which were rejected because at least one edge was already contained in the actual graph realization. It is easy to give counter-examples if these conditions are not satisfied. The question for us was to find out if we get an analogous result for digraph realizations. At the beginning of our work we were confronted with several popular claims arising from the community of complex network analysis, which were as follows:

- It is sufficient to perform only 2-swaps in the directed case, i. e., replace two arcs $(a, b), (c, d)$ by $(a, d), (c, b)$ provided that both arcs have not been contained before.
- This method samples a digraph realization uniformly at random and quickly.

We conclude this section with a summary of our scientific results, answering some of the above formulated claims, by “yes – sometimes”, “no – in general” and “we don’t know how fast this process converges”.

- For the case of digraph realizations we explain that a popular switching algorithm fails in general, because it is not possible to find a sequence of swap operations for each pair of digraph realizations of a digraph sequence.
- We prove that a sequence of swap operations and a further operation 3-*cycle reorientation*, i. e., simply reorient the arcs of a directed induced 3-cycle, can be constructed for each pair of different digraph realizations. Applying classical results from Markov chains (random walks on graphs) we show that an infinite number of such steps lead to a uniform distribution for all digraph realizations of a given digraph sequence. We were not able to show the rapidity of the convergence. After proving these results, we realized that Rao et al. (RJB96) have

already considered the problem in the context of $0, 1$-matrices. They introduced structurally equivalent operations (3-cycles correspond to so-called hexagons), but work on a different Markov chain.

- We study under which conditions it is possible to apply only swap operations leading to a provable uniform sampling. We denote this class of sequences by *arc swap sequences*. Such sequences can be identified in $O(m^2)$ time using matching techniques.
- We prove that a certain *state digraph* for digraph sequences which are not arc swap sequences – (in a state digraph each vertex corresponds to a digraph realization for one given digraph sequence) – decomposes into a number of isomorphic strongly connected components. We can also efficiently determine the number of these components. These results give a theoretical foundation to compute certain network characteristics for example the motif content (MSOI$^+$02) on unlabeled digraphs only using swap operations. However, for other network characteristics, for example betweenness centrality on arcs (KLP$^+$05), this generally leads to incorrect estimations.

In our work we concentrate on two representations of our problem — the graph-theoretic approach and matrices with entries “one” and “zero”. We do not want to prevent other views on our problem. The following short overview about further formulations of the digraph realization problem or its sampling variant shows the naturalness of these problems. At different times, they arose in several communities and have attracted many researchers.

Matrices of zeros and ones with row and column sum vectors. Let A be a matrix of m rows and n columns and let the entries of A be the integers 0 and 1. Furthermore, we denote by r_i the ith row sum and by c_i the ith column sum. We consider the *row sum vector* $R := (r_1, \dots, r_m)$ and the *column sum vector* $C := (c_1, \dots, c_n)$. Clearly, we have $\sum_{i=1}^{m} r_i = \sum_{i=1}^{n} c_i$. The search for a matrix with given row sum vector R and column sum vector C, where $m = n$, corresponds to the digraph realization problem with at most one loop for each vertex if we consider matrix A as an adjacency matrix of a digraph realization. Forbidding entries ‘one’ on the diagonal leads exactly to the digraph realization problem. On the other hand, matrix A can be seen as the adjacency matrix of a bipartite graph where row indices correspond to vertices of one independent vertex set and column indices to vertices of the second one. Let us consider the powers A^k of the adjacency matrix. An entry A^k_{ij} of matrix A^k maps to the number of directed walks from vertex v_i to vertex v_j of length k. For acyclic digraphs (dags) the length of a walk is upper bounded by $n-1$. In other words, each directed walk is already a directed path. Otherwise, we would find directed cycles. Hence, we have $A^k = 0$ for all $k \geq n$. This is one possible definition of *nilpotent matrices.* We conclude that the adjacency matrix of a dag is a nilpotent matrix. Hence, the dag realization problem is the problem of finding a nilpotent matrix for given row and column sums. Sometimes, these matrices are also called *Binary Contingency Tables* corresponding to a bipartite graph realization problem.

Integer linear programming. Given is the row sum vector $R := (r_1, \dots, r_n) \in \mathbb{Z}^n$ and the column sum vector $C := (c_1, \dots, c_n) \in \mathbb{Z}^n$.

We define the variables $x_{ij} \in \{0,1\}$ for all $i, j \in \{1, \dots, n\}$ with $i \neq j$ and demand

$$\sum_{j=1}^{n} x_{ij} = a_i \text{ for all } i \in \{1, \dots, n\} \text{ and}$$

$$\sum_{i=1}^{n} x_{ij} = b_j \text{ for all } j \in \{1, \dots, n\}.$$

This is a formulation for the digraph realization problem. We add further variables $y_1, \dots, y_n \in \{1, \dots, n\}$ and demand that these variables are *pairwise different* (to express these conditions further inequalities are required) and

$$y_i - y_k \leq n(1 - x_{ik})$$

for all $i \neq k$.

This can be interpreted as the condition that there has to be a topological ordering of the vertices. Hence, the increasing ordering of all variables corresponds to a topological sorting. Clearly, the digraph has to be acyclic. Hence, all these constraints give a formulation for the dag realization problem.

2

Fundamental Notions and Notation

In this section, we recall basic notions on graphs and digraphs and fix our notation. We define $\mathbb{N}_n := \{1, \dots, n\}$.

Graphs, forests and trees. A graph G is a tuple $G = (V, E)$ consisting of a vertex set V and the set E of *edges*. Particularly, the edge set E is a subset of all 2-subsets $M := \{\{u, v\} |\ u, v \in V\}$ from V. A graph $G' = (V', E')$ is called *subgraph* of graph G if we have $V' \subseteq V$ and $E' \subseteq E$. We denote graph G as *simple graph*, if there does not exist an edge $\{v, v\} \in E$ with $v \in V$. Note, that we here exclude the case of more than one edge between a vertex u and a vertex v, because in our definition set E is not a multi-set. Let us consider two different types of so-called *walks*. We denote the alternating sequence $W = v_1, e_1, v_2, e_2, \dots, e_{k-1}, v_k$ of vertices and edges as

1. *undirected walk* if $e_i = \{v_i, v_{i+1}\} \in E$ is fulfilled for all $i \in \{1, \dots, k-1\}$;
2. *alternating walk* if we have $e_i \in E \wedge e_{i+1} \notin E$ for either all odd indices or all even indices $i \in \{1, \dots, k-1\}$.

We denote the number $k - 1$ of edges in a walk W as its length $l(W)$. The notion "$l(W)$-walk" means a walk of length $l(W)$. A walk W is called *path* P, if all vertices are pairwise different. Hence, we distinguish between *undirected paths* and *alternating paths*. A walk W is denoted as *cycle* C, if we have $v_1 = v_k$. Again, we get the notion of *undirected cycles* and *alternating cycles*. A graph G which does not possess a subgraph which is a cycle, is called *forest*. We define the following equivalence relation $\sim_{path} \subset V \times V$ with

$$u \sim_{path} v \Leftrightarrow \text{there exists an undirected path from vertex } u \text{ to vertex } v \ .$$

Clearly, this equivalence relation decomposes the set of vertices V in several equivalence classes $V_1, \dots, V_l$ with $V_i \subseteq V$. We call a subgraph $G_i = (V_i, E_i)$ consisting of vertex set V_i and its corresponding induced edge set E_i as *component of* G. A forest which consists of only one component is denoted as *tree*. A graph G is a tree if and only if it has only one component and exactly $|V|-1$ edges. We extend this characterization to forests: A graph is a forest if and only if each of its components is a tree. Obviously, a graph can only be a forest if the number of edges $|E_i|$ in each component $G_i = (V_i, E_i)$ is exactly $|E_i| - 1$. Hence, the number of edges in a forest is smaller or equals $\sum_{i=1}^{l} |E_i| \leq n - 1$.

Neighborhood sets, vertex degrees and f-factors in graphs. We define a *neighborhood set* $N_G(V')$ for a graph $G = (V, E)$ as the set of incident edges $\{u, v\} \in E$ for all vertices $v \in V'$ where $V' \subset V$. In particular, we denote by $N_G(v) := N_G(\{v\})$ the set of all incident edges of vertex $v \in V$. Furthermore, we define a *vertex degree function* $d_G : V \mapsto \{0, \dots, n-1\}$ which assigns to each vertex v the number $d_G(v) := |N_G(v)|$ of its incident edges. We denote $d_G(v)$ as the *vertex degree* of vertex v. Finally, we define a function $f : V \mapsto \{0, \dots, n-1\}$ which assigns to each vertex v an integer between 0 and $n-1$. We call a subgraph G' of G *simple f-factor of graph G* if the condition $d_G(v) = f(v)$ is fulfilled for all $v \in V$. The *f-factor problem* is the decision problem: Given a function f, does there exist a simple f-factor in graph G?

Digraphs, subdigraphs and special structures in digraphs. A *digraph* G is a tuple $G = (V, A)$ consisting of a vertex set V and the set A of *directed arcs*. Particularly, the arc set $A \subseteq V \times V$ is a subset of all ordered vertex pairs, but we exclude the existence of loops $(v, v) \in A$. In the case of loops, we call G *digraph with loops*. A *subdigraph* $G' = (V', A')$ of digraph G is a digraph with $V' \subseteq V$ and $A' \subseteq A$. Let H be a subdigraph of G. We say that $H = (V_H, A_H)$ is an *induced subdigraph* of G if every arc of A with both end-vertices in V_H is also in A_H. We write $H = G\langle V_H \rangle$.

Note, that we here exclude the case of more than one arc beginning at a vertex u and ending at a vertex v, because in our definition set A is not a multi-set.

Let us now consider three different types of a so-called *walk*, namely *directed walks*, *oriented walks* and *alternating oriented walks.* We denote an alternating sequence $W = v_1, a_1, v_2, a_2, \ldots, a_{k-1}, v_k$ of vertices and arcs

1. as *directed walk*, if $a_i = (v_i, v_{i+1}) \in A$ is fulfilled for all $i \in \{1, \ldots, k-1\}$,
2. as *oriented walk*, if we **either** have ($a_i = (v_i, v_{i+1})$ for odd indices $\wedge$ $a_{i+1} = (v_{i+2}, v_{i+1})$ for even indices) **or** ($a_i = (v_{i+1}, v_i)$ for odd indices $\wedge$ $a_{i+1} = (v_{i+1}, v_{i+2})$ for even indices), and
3. as *alternating oriented walk*, if walk W is an oriented walk and we have $a_i \in A \wedge a_{i+1} \notin A$ for all either all odd indices $i \in \{1, \ldots, k-1\}$ or for all even indices $i \in \{1, \ldots, k-1\}$.

We denote the number $k-1$ of arcs in a walk W as its *length* $l(W)$. The notion "$l(W)$-walk" means a walk of length $l(W)$. A walk W is called *path* P, if all vertices are pairwise different. Hence, we distinguish between *directed paths*, *oriented paths* and *alternating oriented paths.* A walk W is denoted as *cycle* C, if we have $v_1 = v_k$. Analogously to the case of paths we get the notions for *directed cycles*, *oriented cycles* and *alternating oriented cycles.* For simplicity, we sometimes omit the arcs a_i in a cycle $C = v_1, a_1, v_2, a_2, \ldots, a_{k-1}, v_1$ and use the shorter form $C = v_1, v_2, \ldots, v_1$ if it is clear from the context which cycle is specified.

A digraph G which does not possess a subdigraph which is a directed cycle is called *directed acyclic graph (dag)* or *acyclic digraph.* We define the following equivalence relation $\sim_{upath} \subset V \times V$ with

$$u \sim_{upath} v \Leftrightarrow \text{there exists an underlying undirected path from vertex } u \text{ to vertex } v \ .$$

Clearly, this equivalence relation decomposes the set of vertices V in several equivalence classes $V_1, \ldots, V_l$ with $V_i \subseteq V$. We call a subdigraph $G_i = (V_i, A_i)$ consisting of vertex set V_i and its corresponding induced arc set A_i as *weak component of* G. A further equivalence relation $\sim_{dpath} \subset V \times V$ with

$$u \sim_{dpath} v \Leftrightarrow \text{there exists a directed path from vertex } u \text{ to vertex } v \text{ and from } v \text{ to vertex } u$$

decomposes a digraph in *strong components of* G.

A *Hamiltonian path* in digraph G is a directed path P containing all vertices $v \in V$ of G.

Neighborhood sets and vertex degrees. Let $N^-_G(V')$ be the set of all incoming arcs $(u,v) \in A$ for all vertices $v \in V'$ in a digraph G. We denote $N^-_G(V')$ as *incoming neighborhood set* of vertex subset $V' \subset V$. Analogously, we define the *outgoing neighborhood set* $N^+_G(V')$ for all outgoing arcs $(v,u) \in A$ for all vertices $v \in V'$. For a digraph G with loops we additionally define a *loop set* $L_G(V')$ containing all loops of a vertex subset $V' \subset V$. In particular, we denote with $L_G(v) := L_G(\{v\})$ the set of loops for vertex v, with $N^-_G(v) := N^-_G(\{v\})$ and $N^+_G(v) := N^+_G(\{v\})$ the sets of all incoming and outgoing arcs at vertex $v \in V$. Furthermore, we define a *vertex indegree function* $d^-_G : V \mapsto \{0,\dots,n-1\}$ which assigns to each vertex v the number $d^-_G(v) := |N^-_G(v)|$ of its incoming arcs. We denote $d^-_G(v)$ as the *indegree* of vertex v. Analogously, we define the *vertex outdegree function* $d^+_G : V \mapsto \{0,\dots,n-1\}$, which assigns to each vertex V the number $d^+_G(v) := |N^+_G(v)|$ of its outgoing arcs. We call $d^+_G(v)$ the *outdegree* of $v \in V$. For digraphs with loops we have $d^-_G := |N^-_G(v)| + |L_G(v)|$ and $d^+_G := |N^+_G(v)| + |L_G(v)|$. A vertex v is denoted as *sink* if the conditions $d^+_G(v) = 0$ and $d^-_G(v) > 0$ are fulfilled. Conversely, a vertex $v \in V$ is denoted as *source* if we have $d^+_G(v) > 0$ and $d^-_G(v) = 0$. A vertex $v \in V$ with $d^+_G(v) > 0$ and $d^-_G(v) > 0$ is called *stream vertex*.

Symmetric differences of graphs and digraphs. For two graph realizations G, G', the symmetric difference of their edge sets $E(G)$ and $E(G')$ is denoted as $G\Delta G' := (E(G) \setminus E(G')) \cup (E(G') \setminus E(G))$. A graph is called *Eulerian* if every vertex has even degree. Note that the symmetric difference $G\Delta G'$ of two graph realizations G, G' is Eulerian. Each component is nothing else but one alternating cycle.

The symmetric difference $G\Delta G'$ of two digraph realizations $G \neq G'$ is defined analogously to the undirected case. Consider for example the digraph realizations G and G' with $A(G) := \{a_1 = (v_1,v_2), a_2 = (v_3,v_4)\}$ and $A(G') := \{a_3 = (v_1,v_4), a_4 = (v_3,v_2)\}$ consisting of exactly two arcs. Then the symmetric difference is the alternating oriented 4-cycle $C := (v_1,a_1,v_2,a_4,v_3,a_2,v_4,a_3,v_1)$ where $(v_i,v_{i+1}) \in A(G)$ for $i \in \{1,3\}$ and $(v_{i+1},v_i) \in A(G')$ for $i \in \{2,4\}$ taking indices $i \bmod 4$. In contrast to the undirected case where each component of the symmetric difference is one alternating cycle, one weak component of the symmetric difference of two digraph realizations may decompose into more than one alternating oriented cycle, see Figures 2.1 and 2.2.

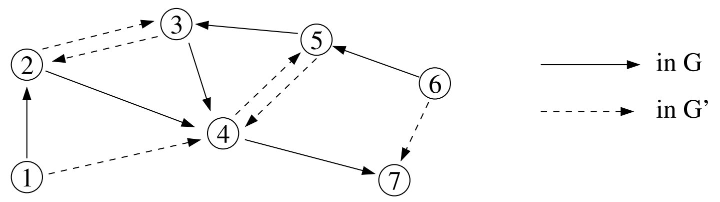


Figure 2.1: Example: Two digraph realizations G and G' with the same digraph sequence.

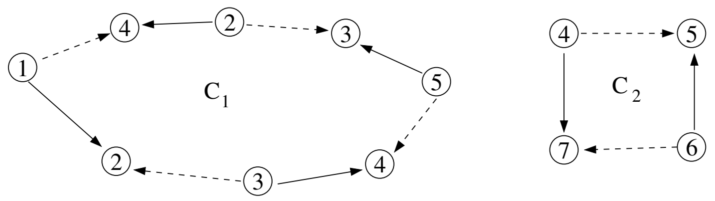


Figure 2.2: Decomposition of the symmetric difference $G \Delta G'$ of Figure 2.1 into the minimum number of alternating directed cycles.

Topological sorting. A topological ordering of the vertex set V from a dag $G = (V, A)$ is the enumeration of all vertices with respect to the relation $R_{top} \subseteq V \times V$. We define:

$$vR_{top}w \Leftrightarrow \text{there exists no directed path starting in } w \text{ and ending in } v.$$

Since our digraph is a dag — we have no directed cycle — we get for each pair of vertices v, w two possibilities. Either there exists at least one directed path from v to w, then it follows $vR_{top}w$ because the acyclicity implies that there does not exist a directed path from w to v. The second possibility is that there is no directed path from vertex w to v and vice versa. Hence, we get $vR_{top}w$ and $wR_{top}v$. In a dag we can compare all vertex pairs with respect to this relation. Hence, we can determine an enumeration for vertex set V such that we have $v_iR_{top}v_j$ if $i < j$. This enumeration is called *topological sorting*. Note, that it is possible to arrange all vertices in a chain, although relation R_{top} is not antisymmetric and not transitive. To see this, consider for example a dag consisting of vertex set $V := \{v_1, v_2, v_3\}$ and the single arc (v_1, v_3). Clearly, we have $v_3R_{top}v_2$ and $v_2R_{top}v_1$ but it does not follow $v_3R_{top}v_1$.

Types of lexicographical sortings. In our scenario we need two different notions for a lexicographical relation. The first we call *lexicographical relation with respect to the first component* $\leq_{lex1} \subset \mathbb{Z}^+ \times \mathbb{Z}^+$ with

$$\begin{pmatrix} a \\ b \end{pmatrix} <_{lex1} \begin{pmatrix} a' \\ b' \end{pmatrix} \Leftrightarrow a < a' \text{ or } (a = a' \;\wedge\; b < b').$$

By *lexicographical relation with respect to the second component* $\leq_{lex2} \subset \mathbb{Z}^+ \times \mathbb{Z}^+$ we denote the relation which is defined as

$$\begin{pmatrix} a \\ b \end{pmatrix} <_{lex2} \begin{pmatrix} a' \\ b' \end{pmatrix} \Leftrightarrow b < b' \text{ or } (b = b' \;\wedge\; a < a').$$

Note, that $\binom{a}{b} =_{lex} \binom{a'}{b'}$ if and only if $a = a'$ and $b = b'$. The lexicographical ordering (in both cases) is a total ordering, i. e., reflexive, antisymmetric and transitive. Hence, it is possible to number all tuples of a sequence S such that we have $\binom{a_i}{b_i} \leq_{lex1} \binom{a_j}{b_j}$ if and only if $i < j$ (the same is true for $\leq_{lex2}$). Such a labeling of a sequence we denote by *increasing lexicographical sorting with respect to the first/second component.* If we number the tuples of an increasing lexicographical sequence in the opposite direction we call this numbering *decreasing lexicographical sequence.*

Adjacency matrices and nilpotent matrices. The neighborhoods of vertices in a digraph can be represented by an $(n \times n)$-matrix, the so-called *adjacency matrix.* The matrix $A := (A_{ij})_{i,j \in \mathbb{N}_n}$ is defined by

$$A_{ij} \quad := \quad \begin{cases} 1 & \text{if } (v_i, v_j) \in A \\ 0 & \text{otherwise.} \end{cases}$$

Hence, the adjacency matrix of a digraph has “zero” entries on its diagonal. Let us consider the powers A^k of an adjacency matrix. An entry (A^k_{ij}) of matrix A^k maps to the number of directed walks from vertex v_i to vertex v_j of length k. For acyclic digraphs (dags) the length of a walk is upper bounded by $n-1$. In other words each directed walk is already a directed path. Otherwise, we would find directed cycles. Hence, we have $A^k = 0$ for all $k \geq n$. This is exactly the definition of *nilpotent matrices.* We conclude, that each adjacency matrix of a dag is a nilpotent matrix. In particular, there exists a vertex ordering $v_1, \ldots, v_n$ such that the corresponding adjacency matrix is an upper triangular matrix with zero entries on its diagonal. On the other hand, this ordering is also a topological sorting of the vertices because we have $v_i R_{top} v_j$ for $i < j$, i. e., there is no directed path from vertex v_j to vertex v_i.

3

Digraph Sequences – Realization and Characterization

“That is what learning is. You suddenly understand something you’ve understood all your life, but in a new way.” (Doris Lessing)

3.1 Digraph Realizations

In 1973, Kleitman and Wang (KW73) proposed two different algorithms to construct a digraph realization (defined in Chapter 1) for a given digraph sequence. These algorithms are extensions for the *graph realization problem* of the undirected version of Havel (Hav55) and Hakimi (Hak62) which is defined as follows. For a given finite sequence $S := (a_1), \dots, (a_n)$ with $a_i \in \mathbb{N}$ one has to decide if there exists a labeled graph $G = (V, E)$ such that each integer a_i possesses a corresponding vertex v_i with degree $d_G(v_i) = a_i$. In this case, we call graph G *graph realization* and we call the undirected sequence S *graph sequence*. The *Havel-Hakimi-Algorithm* works as follows. It greedily realizes a sequence (if possible) by determining the adjacent edge set of a selected vertex step by step. For an undirected sequence $S = (a_1), \dots, (a_n)$ one chooses an arbitrary start of this sequence, say a_i. Then one determines the a_i largest entries $a_{j_1}, \dots, a_{j_{a_i}}$ (without a_i) in S. If these a_i entries exist, one builds the edge set of a realization $G = (V, E)$ with labeled vertex set V by inserting the edges $\{v_i, v_{j_1}\}, \dots, \{v_i, v_{j_{a_i}}\}$. In sequence S each integer $a_{j_1}, \dots, a_{j_{a_i}}$ has to be reduced by one and a_i can be deleted afterwards. This step is repeated until sequence S is empty or one gets stuck. In the latter case, it is proven that S is not a graph sequence. Kleitman and Wang did not have their main focus on the digraph realization problem. Indeed, they gave the proof for an extension of the so-called *k-factor conjecture* of Kundu (Kun73). The k-factor conjecture is about the question whether there exists for a given undirected sequence $S = (a_1), \dots, (a_n)$ a graph realization with a given f-factor as a subgraph where $f(v) = k$ is a constant for all $v \in V$. Kundu proved the existence of a k-factor if and only if sequence $S' := (a'_1 = a_1 - k), \dots, (a'_n = a_n - k)$ is also a graph sequence. Kleitman and Wang give a constructive proof which results in an algorithm for an extension of the k-factor conjecture. They consider the question whether there exists for a given undirected sequence a graph realization with an f-factor as a subgraph such that we have $f(v_i) = k$ for a given vertex subset $v_i \in V' \subset V$ and $f(v_i) = k+1$ for the remaining vertices $v_i \in V \setminus V'$. The algorithm works in two steps. First they determine graph realizations G and G' from sequence S and sequence $S' := (a'_1 = a_1 - p_1), \dots, (a'_n = a_n - p_n)$ with $p_i = k$ (for $v_i \in V'$) or $p_i = k + 1$ (for $v_i \in V \setminus V'$) by using a slight modification of the algorithm by Havel and Hakimi. The difference to the general variant where we choose arbitrary a_i, is the necessity first to choose numbers with $a_i = a'_i$. Then they have to swap some arcs in the graph realization G of S as long as the graph realization G' of S' is a subgraph of G. Clearly, such a graph realization also contains the demanded f-factor. The directed variant of the problem is as follows. Does there exist for a given digraph sequence $S = \binom{a_1}{b_1}, \dots, \binom{a_n}{b_n}$ a subdigraph H with vertex outdegrees $d^+_H(v_i) = k$ for a subset $i \in I \subset \mathbb{N}_n$ and vertex outdegrees $d^+_H(v_j) = a_j$ for all $j \in \mathbb{N} \setminus I$? Analogously to the undirected case, Kleitman and Wang prove that this is true if and only if sequence $S' = \binom{a'_1}{b'_1}, \dots, \binom{a'_n}{b'_n}$ with $a'_i = a_i - k$ for $i \in I$, $a'_j = 0$ for $j \in \mathbb{N} \setminus I$ and $b_i \geq b'_i$ for all $i \in \mathbb{N}_n$ is a digraph sequence. The proofs are similar to the undirected case. Therefore, they developed two different algorithms for realizing digraph sequences (KW73). We summarize their results and formulate their theorems with a slight, but easy extension for the reverse implication.

Theorem 3.1 (digraph realization with arbitrary tuple choice (KW73))**.**
Let $S = \binom{a_1}{b_1}, \dots, \binom{a_n}{b_n}$ be a sequence and $\binom{a_i}{b_i}$ a tuple with $b_i > 0$. Let M_i be the set of all stream and sink tuples of S without tuple $\binom{a_i}{b_i}$. Furthermore, let $\binom{a_{l_1}}{b_{l_1}}, \dots, \binom{a_{l_{|M_i|}}}{b_{l_{|M_i|}}}$ be a decreasing lexicographical sorting by the first component of tuples in M_i. Sequence S is a digraph sequence if and only if the following conditions are true:

1. $|M_i| \geq b_i$ *and*

2. *sequence* $S^i := \binom{a_1^i}{b_1^i}, \ldots, \binom{a_n^i}{b_n^i}$ *with*

$$b_j^i := \begin{cases} b_j & \text{if } i \neq j \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad a_j^i := \begin{cases} a_j - 1 & \text{if } i \in \{l_1, \ldots, l_{b_i}\} \\ a_j & \text{otherwise} \end{cases}$$

is a digraph sequence.

This theorem is the analogous version of the algorithm of Havel and Hakimi. For a digraph sequence the algorithm constructs a digraph $G = (V, A)$ using a repeated application of this theorem on the current sequence S^i. Hence, it is possible to construct in each step the arcs (v_i, v_{l_j}) with $j \in \{1, \ldots, b_i\}$ or if condition 1. or 2. is not fulfilled, we can conclude that sequence S is not a digraph sequence. There exists an interesting insight of LaMar (LaM09). He developed a "parallel" version of the realization algorithm for sequences $S := \binom{a_1}{b_1}, \ldots, \binom{a_n}{b_n}$, where $a_i = b_i$ and $\sum_{i=1}^n a_i$ is even. The idea is to consider two sequences, namely sequence S and a further sequence $\overline{S}$, where the roles of a_i and b_i are swapped. LaMar proved that it is possible for an arbitrary tuple $\binom{a_i}{b_i}$ in sequence S and $\binom{b_i}{a_i}$ in sequence $\overline{S}$ to construct identical a_i-subsets $M' \subset M_i$. Hence, in each step one can simultaneously build all $a_i (= b_i)$ incoming and outgoing arcs of vertex v_i for a digraph realization G. Each directed cycle $C = (u, v, u)$ of length 2 can be replaced by the undirected edge $\{u, v\}$ which results in a graph realization of sequence $S^* := (a_1), \ldots, (a_n)$. In this sense the algorithm for such digraph sequences of LaMar is nothing else but the realization algorithm of Havel and Hakimi. Kleitman and Wang also give a second algorithm for constructing realizations. Here, the choice of a tuple $\binom{a_i}{b_i}$ is not arbitrary but one does not need a lexicographical sorting of the candidate set M.

Theorem 3.2 (digraph realization with a special tuple choice (KW73))**.**
Let $S = \binom{a_1}{b_1}, \ldots, \binom{a_n}{b_n}$ *be a sequence and* $\binom{a_i}{b_i}$ *a largest stream or sink tuple with respect to the lexicographical ordering by the second component. Furthermore, we define the set* M *consisting of all stream tuples and sink tuples of* S *without tuple* $\binom{a_i}{b_i}$*. Consider a decreasing sorting of tuples* $\binom{a_{l_1}}{b_{l_1}}, \ldots, \binom{a_{l_{|M|}}}{b_{l_{|M|}}}$ *in* M *such that we have* $i < j$ *if and only if* $a_{l_i} \geq a_{l_j}$*. Sequence* S *is a digraph sequence if and only if:*

1. $|M| \geq b_i$ *and*

2. *sequence* $S' := \binom{a'_1}{b'_1}, \ldots, \binom{a'_n}{b'_n}$ *with*

$$b'_j := \begin{cases} b_j & \textit{if } i \neq j \\ 0 & \textit{otherwise} \end{cases} \qquad \textit{and} \qquad a'_j := \begin{cases} a_j - 1 & \textit{if } j \in \{l_1, \ldots, l_{b_i}\} \\ a_j & \textit{otherwise} \end{cases}$$

is a digraph sequence.

Note that both theorems can be used to realize source-sink-sequences (defined in Chapter 1). We would like to point out that the algorithms of Kleitman and Wang are relatively unknown until today. So it is not surprising that these ideas have been reinvented several times. The proofs of Theorem 3.1 and Theorem 3.2 are elementary. Starting with a digraph realization G for sequence S, they show that there also exists a digraph realization where the vertex v_i is connected with vertices $v_{l_1}, \ldots, v_{l_{b_i}}$ by a directed arc. The deletion of all these arcs leads to a digraph realization of sequence S' (in Theorem 3.2) or S^i (in Theorem 3.1), respectively.

3.2 Digraph Characterizations

Four authors, namely David Gale (Gal57), Herbert J. Ryser (Rys57), Delbert Ray Fulkerson (Ful60) and Wai-Kai Chen (Che66) gave sufficient and necessary conditions which completely characterize digraph sequences. Actually, none of the mentioned authors has found the following theorem in its general form. Gale and Ryser dealt with other problems. However, their insights are fundamental for characterizing digraph sequences. Fulkerson gave a partial result and Chen formulated and proved the final details. Therefore, we cite the following theorem as the result of all four authors.

Theorem 3.3 (Characterization of digraph sequences (Ful60, Rys57, Che66, Gal57))**.** *Let $S := \binom{a_1}{b_1}, \ldots, \binom{a_n}{b_n}$ be a sequence in decreasing lexicographical order with respect to the first component. S is a digraph sequence if and only if the following conditions are fulfilled:*

$$\sum_{i=1}^{k} \min(b_i, k-1) + \sum_{i=k+1}^{n} \min(b_i, k) \geq \sum_{i=1}^{k} a_i$$

for all $k \in \{1, \ldots, n\}$.

Ryser's (Rys57) and Gale's (Gal57) view led to additional insights. In 1957, they independently published some results for a version of the digraph realization problem. They knew a famous result from 1935 by P. Hall (Hal35) for the following problem. Given are a finite set $X := \{x_1, \ldots, x_n\}$ and a collection of subsets $S_i \subset X$ with

$i \in \{1, \ldots, n\}$. The question is under which conditions one can select a distinct element x_i from each subset S_i. Hall formulated sufficient and necessary conditions for this case. More precisely, the existence of these elements is ensured if and only if the union of an arbitrary choice of k sets $S_{l_1}, \ldots, S_{l_k}$ contains a k-subset of X. Gale considered this problem as a kind of transportation model and extended it to a flow problem on a digraph, which can completely be characterized by exponentially many inequalities. The proof was given using the well-known "max flow and min cut" theorem of Ford and Fulkerson (FF56) which was published some months earlier in the same year. As a possible application he formulated the following version of the digraph problem. Given are two undirected sequences of non-negative numbers $S := (a_1, a_2, \ldots, a_n)$ and $S' := (b_1, \ldots, b_{n'})$ such that we have $a_i \geq a_{i+1}$ for all $i \in \{1, \ldots, n-1\}$ and the condition $\sum_{i=1}^{n} a_i = \sum_{i=1}^{n'} b_i$ holds. Does there exist an $(n \times n')$-matrix M with entries in $\{0, 1\}$ such that we have for the sum of the ith row at most value b_i and for the jth column sum at least value a_i. Such a matrix M can be interpreted as an adjacency matrix of a digraph *with at most one loop per vertex.* Clearly, the ith row sum and the jth column sum are exactly the vertex outdegree $d_G^+(v_i)$ and indegree $d_G^-(v_j)$ in the corresponding digraph. At the same time, Ryser investigated combinatorial properties of matrices with entries zero and one. Apart from the small difference that he considered the problem version demanding exact values b_i and a_j for the ith row sum and the jth column sum, he found the same connections. On the other hand, he came up with an inductive proof without using any result from flow theory. Furthermore, his view also allows other insights about the sets of all such sequence pairs with n tuples and $m := \sum_{i=1}^{n} a_i$ entries. To explain these results, let us give a definition for a special kind of matrix – "Ferrers diagram" – which was first given by Sylvester (SF82) in 1882. He refers to a solution method of Norman Macleod Ferrers (1829-1903) for a partition problem which was stated by Adams in a Tripos Paper of 1847. Ferrers has never published his idea but Sylvester wrote in his work "The above proof of the theorem of reciprocity is due to Dr. Ferrers,... . It was never been made public by its author, but first promulgated by myself ... in 1853." (VrAM04, Ven13, Kim99, Fer). Since we need different types of such matrices, we here change this notion into *canonical Ferrers matrix.*

Definition 3.1 (canonical Ferrers matrix for sequence S)**.** *Given is a sequence $S = \binom{a_1}{b_1}, \ldots, \binom{a_n}{b_n}$ with $a_i \geq a_{i+1}$ for all $i \in \{1, \ldots, n-1\}$. We construct an $(n \times n)$-matrix*

$F := (F_{ij})_{i,j \in \mathbb{N}_n}$ *with*

$$F_{ij} \;:=\; \begin{cases} 1 & \text{if } j \leq b_i \\ 0 & \text{if } j > b_i. \end{cases}$$

For a construction of F one has to fill b_i entries "one" in row i starting on the left side. The rest of this row is set to "zero". The resulting jth column sum we denote by a^*_j. On the other hand, F can be seen as the adjacency matrix of a digraph with at most one loop per vertex with vertex indegrees $a^*_1, \ldots, a^*_n$ and vertex outdegrees $b_1, \ldots, b_n$. We call sequence $S^* := \binom{a^*_1}{b_1}, \ldots, \binom{a^*_n}{b_n}$ the *corresponding sequence of the canonical Ferrers matrix F*. Note that sequence S^* may contain $\binom{0}{0}$ tuples.

Example 3.1 (canonical Ferrers matrix F)**.** *Given is sequence*

$$S := \binom{6}{2}, \binom{6}{3}, \binom{6}{4}, \binom{5}{3}, \binom{5}{2}, \binom{3}{7}, \binom{1}{4}, \binom{1}{4}, \binom{0}{4}.$$

The canonical Ferrers matrix F is

$$\begin{array}{c c}
\begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0
\end{bmatrix} &
\begin{array}{l} b_1 = 2 \\ b_2 = 3 \\ b_3 = 4 \\ b_4 = 3 \\ b_5 = 2 \\ b_6 = 7 \\ b_7 = 4 \\ b_8 = 4 \\ b_9 = 4 \end{array} \\
\begin{matrix} 9 & 9 & 7 & 5 & 1 & 1 & 1 & 0 & 0 \end{matrix} & a^*_i
\end{array}$$

Note, that the a^*_i are decreasingly sorted.

Ryser and Gale actually proved the existence of a matrix M with ith row sum b_i and jth column sum a_j if and only if we have $\sum_{i=1}^{k} a_i \leq \sum_{i=1}^{k} a^*_i$ for all $k \in \{1, \ldots, n\}$. The corresponding digraph of adjacency matrix F can be seen as one "largest" possible digraph for a given "outdegree sequence" $b_1, \ldots, b_n$ which is realizable. Therefore, the corresponding sequence S^* bounds the set of non-realizable sequences with outdegree sequence $b_1, \ldots, b_n$. It is therefore called *threshold sequence.* There exist very interesting equivalent characterizations of threshold sequences. An overview is given in the book of Mahadev and Peled (MP95).

Theorem 3.4 (characterization of digraph sequences with loops (Rys57, Gal57)**).**
Let $S := \binom{a_1}{b_1}, \ldots, \binom{a_n}{b_n}$ be a sequence such that $a_i \geq a_{i+1}$ for all $i \in \{1, \ldots, n-1\}$. Then sequence S is a digraph sequence with loops if and only if the following conditions are fulfilled:

(1.) There exists a canonical Ferrers matrix F with corresponding sequence $S^* = \binom{a_1^*}{b_1}, \ldots, \binom{a_n^*}{b_n}$ and

(2.) $\sum_{i=1}^{k} a_i^* \geq \sum_{i=1}^{k} a_i$ for all $k \in \{1, \ldots, n\}$.

The most important contribution of Ryser and Gale is a polynomial time algorithm to decide the realizability of a sequence because only the n inequalities of condition (2.) have to be checked. This is in fact the progress compared with the classical result of Hall allowing a complete characterization of a sequence but ignoring the costs to find one. This nice property has led to the natural question whether it is also possible to characterize digraph sequences where loops are forbidden in a digraph realization. Hence, the adjacency matrix of such a digraph realization must have only zeros on its diagonal. We modify the canonical Ferrers matrix and define a *diagonal-free Ferrers matrix.*

Definition 3.2 (diagonal-free Ferrers matrix for sequence S)**.**
Let $S := \binom{a_1}{b_1}, \ldots, \binom{a_n}{b_n}$ be a sequence in decreasing lexicographical order with respect to the first component. We construct an $(n \times n)$-matrix $F := (F_{ij})_{i,j \in \mathbb{N}_n}$ with

$$F_{ij} \quad := \quad \begin{cases} 1 & \textit{if } (j \leq b_i \wedge i < j) \vee (j \leq b_i + 1 \wedge i > j) \\ 0 & \textit{otherwise.} \end{cases}$$

Let again a_i^* denote the ith column sum of F. We call sequence $S^* = \binom{a_1^*}{b_1}, \ldots, \binom{a_n^*}{b_n}$ the corresponding sequence of the diagonal-free Ferrers matrix F.

Note again that sequence S^* may contain $\binom{0}{0}$ tuples. Let us consider the following example.

Example 3.2 (diagonal-free Ferrers matrix F)**.** *Given is the sequence*

$$S := \binom{6}{4}, \binom{6}{3}, \binom{6}{2}, \binom{5}{3}, \binom{5}{2}, \binom{3}{7}, \binom{1}{4}, \binom{1}{4}, \binom{0}{4}$$

in decreasing lexicographical order with respect to the first component. Then we get the diagonal-free Ferrers matrix F

$$\begin{array}{c} \begin{bmatrix} 0&1&1&1&1&0&0&0&0\\ 1&0&1&1&0&0&0&0&0\\ 1&1&0&0&0&0&0&0&0\\ 1&1&1&0&0&0&0&0&0\\ 1&1&0&0&0&0&0&0&0\\ 1&1&1&1&1&0&1&1&0\\ 1&1&1&1&0&0&0&0&0\\ 1&1&1&1&0&0&0&0&0\\ 1&1&1&1&0&0&0&0&0 \end{bmatrix} \\ \begin{matrix} 8&8&7&6&2&0&1&1&0 \end{matrix} \end{array} \begin{array}{l} b_1=4\\ b_2=3\\ b_3=2\\ b_4=3\\ b_5=2\\ b_6=7\\ b_7=4\\ b_8=4\\ b_9=4\\ a_i^* \end{array}$$

Note, that the column sums a_i^* do not necessarily build a decreasing sequence of integers in contrast to the column sums of the canonical Ferrers matrix. However, we find an analogous characterization as in Theorem 3.4.

Theorem 3.5 (characterization of digraph sequences without loops).
Let $S := \binom{a_1}{b_1}, \ldots, \binom{a_n}{b_n}$ be a sequence in decreasing lexicographical order. Then sequence S is a digraph sequence if and only if the following conditions are fulfilled:

(1.) There exists a diagonal-free Ferrers matrix F with corresponding sequence $S^* = \binom{a_1^*}{b_1}, \ldots, \binom{a_n^*}{b_n}$ and

(2.) $\sum_{i=1}^{k} a_i^* \geq \sum_{i=1}^{k} a_i$ for all $k \in \{1, \ldots, n\}$.

The difference to the version of Ryser and Gale can be found in the necessity to label sequence S in a *decreasing lexicographical order*. However, we want to show the connection between this characterization and Theorem 3.3. Determining the jth column sums of the diagonal-free Ferrers matrix directly gives us the following equivalent formulation:

$$\sum_{i=1}^{k} a_i^* = \sum_{i=1}^{k} \min(b_i, k-1) + \sum_{i=k+1}^{n} \min(b_i, k) \text{ for all } k \in \{1, \ldots, n\}.$$

Hence, the characterization of sequences using diagonal-free Ferrers matrices is an equivalent formulation of Theorem 3.3. However, none of the above mentioned authors formulated this complete result. Fulkerson (Ful60) gave the first characterization for digraph sequences in general but his formulation requires exponentially many inequalities. He overlooked the idea to sort a sequence in lexicographical order. Instead, he handled a special case of sequences which can be sorted such that we get $a_i \geq a_{i+1}$ and $b_i \geq b_{i+1}$ for all $i \in \{1, \dots, n\}$. It is remarkable that this sorting is already a lexicographical sorting. For this case, Fulkerson was able to give a characterization with polynomial many inequalities. To the best of our knowledge, Wai-Kai Chen detected the important detail to sort a sequence in lexicographical order in his article (Che66) of 1966. Interestingly, his work is relatively unknown until today and has been overlooked for many times. His proof is elementary and not difficult. It is worth to mention that he handled a more general case of the digraph realization problem. Namely, he parameterized the number ℓ of allowed loops and the number p of allowed parallel arcs between two vertices. Now he solved completely the characterization problem of digraphs with at most ℓ loops and p parallel arcs. On the other hand, the view of Ryser and Gale on matrices makes it possible to consider further variants of our digraph realization problem. A matrix M can also be interpreted as the adjacency matrix of an undirected bipartite graph, where the rows and columns correspond to the two independent vertex sets. Then the row and column sums are nothing else but the vertex degrees in this graph.

4

Dag Sequences

"Dass etwas schwer ist, muss ein Grund mehr sein, es zu tun." (Rainer Maria Rilke)

4.1 Dag Realizations

In Chapter 1, we introduced the dag realization problem and stated the complexity status as NP-complete (Nic11). Now, we want to introduce a special class of so-called *opposed sequences* for which we are able to solve the problem in polynomial time, see also our publications (BM11a, BM11b). The main difficulty for the dag realization problem is to find out a topological ordering of the sequence. In the case where we have one, our problem is nothing else but a directed f-factor problem on a complete digraph. The labeled vertices of this complete digraph are ordered in the given topological sorting of our dag problem. This problem can be reduced to a bipartite undirected f-factor problem which can be solved in polynomial time via a reduction by Tutte (Tut54) to a bipartite perfect matching problem. It turns out, that a certain ordering of *opposed sequences* always leads to a topological ordering of the tuples for at least one dag realization of a given dag sequence (Corollary 4.1). On the other hand, it is not necessary to apply the reduction via Tutte if we possess one possible topological ordering of a dag sequence. The solution is much easier. Next, we describe our approach. We denote a dag sequence $S := \binom{a_1}{b_1}, \ldots, \binom{a_n}{b_n}$ which possesses a dag realization with a topological numbering corresponding to the increasing numbering of its tuples by *dag sequence for a given topological order* and analogously the realization $G = (V, A)$ by *dag realization for a given topological order.* A realization algorithm works as follows.

Consider the first tuple $\binom{a_{q+1}}{b_{q+1}}$ from its increasing topological ordered sequence which is not a source tuple. Then there must exist at least a_{q+1} largest source tuples with a smaller number in the given dag sequence. Reduce the a_{q+1} largest source tuples by one and set the indegree of tuple $\binom{a_{q+1}}{b_{q+1}}$ to 0. That means, we reduce sequence $S := \binom{a_1}{b_1}, \dots, \binom{a_{q+1}}{b_{q+1}}, \dots, \binom{a_n}{b_n}$ to sequence S'. If we get zero tuples in S', then we delete them and denote the new sequence for simplicity by S'. Furthermore, we label this sequence with a new numbering starting from one to its length and consider this sorting as the given topological ordering of S'. We repeat this process until we get an empty sequence (corresponding to the realizability of S) or get stuck for the given order (corresponding to the non-realizability of S). The correctness of our algorithm is proven in Theorem 4.1.

Theorem 4.1. *S is a dag sequence for a given topological order $\Leftrightarrow$ S' is a dag sequence for its topological order.*

Proof. $\Leftarrow$: Trivial.
$\Rightarrow$: We consider a dag realization for the given topological ordering of dag sequence S. Clearly, we find at least a_{q+1} sources. This is true, because a first vertex with non-empty incoming neighborhood set in a topological sorting (a sink or a stream vertex) of a dag can only possess sources in its incoming neighborhood set. Otherwise, this numbering is not a topological sorting. Assume, there is no dag realization for the given topological order, such that a_{q+1} largest sources are connected with vertex v_{q+1}. In this case, we consider a dag realization G to this topological order such that the maximum possible number of largest sources is connected with vertex v_{q+1}. Then we have two sources v_i and v_j with $(v_i, v_{q+1}) \notin A$, $(v_j, v_{q+1} \in A)$, $b_i > b_j$ and $i, j < q + 1$. Since $b_i > 0$ and v_{q+1} is the first non-source tuple, there is a non-source vertex v_k $(k > q + 1)$ with $(v_i, v_k) \in A$ and $(v_j, v_k) \notin A$. We define a new digraph $G^* := (G \setminus \{v_i, v_k\} \cup \{v_j, v_{q+1}\}) \cup (\{v_i, v_{q+1}\} \cup \{v_j, v_k\})$. Obviously, G^* is a dag realization for the given topological order of sequence S. Contradiction to the assumption that G is a dag realization with the maximum possible number of largest sources connected with vertex v_{q+1}. Hence, there exists a dag realization G to the given topological order such that vertex v_{q+1} has in its incoming neighborhood set only the a_{q+1} largest sources from the set of all sources v_i with $i < q + 1$. We delete the incoming neighborhood set of vertex v_{q+1} and yield a dag realization for sequence S' for its given topological ordering. □

However, up to know we do not know how to determine a topological ordering for an arbitrary dag sequence. On the other hand, we are able to restrict the types of permutations to a certain class (see Corollary 4.2). We believe, this is the reason why our algorithmic experiments are quite successful for most sequences in our deterministic approaches as well as in our proposed randomized algorithm, see Section 4.2.

4.1.1 Opposed Sequences

We now turn towards a special class of sequences. To this end, we first define a new ordering $\leq_{opp} \subset \mathbb{Z}^2 \times \mathbb{Z}^2$.

Definition 4.1 (opposed relation). *Given are* $c_1 := \binom{a_1}{b_1} \in \mathbb{Z}^2$ *and* $c_2 := \binom{a_2}{b_2} \in \mathbb{Z}^2$. *We define:* $c_1 \leq_{opp} c_2 \Leftrightarrow (a_1 \leq a_2 \wedge b_1 \geq b_2)$.

Note, that a pair c_1 equals c_2 with respect to the opposed relation if and only if $a_1 = a_2$ and $b_1 = b_2$. The opposed relation is reflexive, transitive and antisymmetric and therefore a partial order. On the other hand it is not possible to compare all pairs of tuples c_1 and c_2. Hence, the opposed order is not a total order. In the following, we consider a special class of sequences which can easily be handled with respect to our dag realization problem. We call a sequence S *opposed sequence*, if it is possible to sort its stream tuples in such a way, that $a_i \leq a_{i+1}$ and $b_i \geq b_{i+1}$ is valid for stream tuples with indices i and $i+1$. In this case, we have the property $\binom{a_i}{b_i} \leq_{opp} \binom{a_{i+1}}{b_{i+1}}$ for all stream tuples. At the beginning of the sequence we insert all source tuples such that the b_i build a decreasing sequence and at the end of sequence S we put all sink tuples in increasing ordering with respect to the corresponding a_i. The notion *opposed sequence* describes a sequence, where it is possible to compare all stream tuples among each other and to put them in a "chain". Indeed, this is not always possible because the opposed order is not a total order. With such a labelling we call a sequence *increasing opposed sequence.* If we label an increasing opposed sequence in its opposite direction we call this sequence *decreasing opposed sequence.* Obviously, it is possible that a stream tuple is not comparable with a source tuple or a sink tuple. However, these sequences turn out to be "well-behaved". Next we follow the classic approach of solving this problem with an inductive construction of a realization. But it turns out that we need completely new ideas to save the classic method. We also obtain some interesting insights, for example we found that an increasing opposed sorting of an opposed dag sequence is a topological sorting of at least one dag realization, too.

The following theorem is the basis of a realization algorithm. It is similar to an inductive, greedy-like algorithm as in the classic approach by Havel and Hakimi (Hav55, Hak62). On the other hand, it differs completely in its details from this algorithm. In each step, we choose the smallest stream tuple with respect to the opposed relation and connect it with largest sources — namely with one arc from each source. After this step, this stream tuple will be a new source. If this step is not possible we are sure that sequence S is not a dag sequence.

Theorem 4.2 (opposed sequences)**.** *Let S be an increasing opposed sequence which is not a source-sink-sequence. Furthermore, let tuple $\binom{a_{i_{min}}}{b_{i_{min}}}$ be the smallest stream tuple with respect to the opposed ordering. Then sequence S is a dag sequence if and only if there exist at least $a_{i_{min}}$ source tuples in S and if*

$$S' := \begin{pmatrix} 0 \\ b_1 - 1 \end{pmatrix}, \dots, \begin{pmatrix} 0 \\ b_{a_{i_{min}}} - 1 \end{pmatrix}, \begin{pmatrix} 0 \\ b_{a_{i_{min}}+1} \end{pmatrix}, \dots, \begin{pmatrix} 0 \\ b_{i_{min}-1} \end{pmatrix}, \begin{pmatrix} 0 \\ b_{i_{min}} \end{pmatrix}, \begin{pmatrix} a_{i_{min}+1} \\ b_{i_{min}+1} \end{pmatrix}, \dots, \begin{pmatrix} a_n \\ b_n \end{pmatrix}$$

is a dag sequence.

Proof. $\Leftarrow$: Let S' be a dag sequence. By our assumption, there are at least $a_{i_{min}}$ source tuples in S. From a dag realization G' of S', we construct a new digraph G by inserting arcs $(v_1, v_{i_{min}}), \dots, (v_{a_{i_{min}}}, v_{i_{min}})$ into G'. Obviously, G is a dag, because new arcs outgoing from sources, cannot create a directed cycle. On the other hand, G is a digraph realization, because the conditions $d^+_G(v_i) = b_i$ and $d^-_G(v_i) = a_i$ are fulfilled for all $i \in \{1, \dots, n\}$.
$\Rightarrow$: Let S be a dag sequence and G a dag realization of S. There exists at least one source in G, because G is an acyclic digraph. We denote the number of sources in G by k.

Claim 1: There exists a stream vertex v_j with $d^-_G(v_j) \leq k$, such that the incoming neighborhood set $N^-_G(v_j)$ only contains sources. S contains at least $a_{i_{min}} \leq k$ source tuples.

Assume this is not the case. We delete all k sources and its outgoing arcs from G. By that we do not create a new source, because G does not contain a vertex which possesses only sources as incoming arcs — a contradiction to the assumption that G is a dag. For the stream vertex with smallest index it follows from $\binom{a_{i_{min}}}{b_{i_{min}}} <_{opp} \binom{a_j}{b_j}$ that $d^-_G(v_{i_{min}}) \leq d^-_G(v_j) \leq k$. Hence, there are at least $a_{i_{min}}$ source tuples in S.

Claim 2: There exists a dag realization $G := (V, A)$ such that a stream vertex v_j possesses as incoming neighborhood set $N^-_G(v_j)$ exactly the $d^-_G(v_j)$ largest sources.

Assume the opposite. By Claim 1, there exists a dag realization G, such that the incoming neighborhood set $N_G^-(v_j)$ of a stream vertex v_j only consists of sources. We choose such a digraph G with the largest possible number of maximum sources in $N_G^-(v_j)$. That means, we cannot replace a source $v_i \in N_G^-(v_j)$ by a source $v_{i^*} \notin N_G^-(v_j)$ with $d_G^+(v_i) < d_G^+(v_{i^*})$. By our assumptions there are sources v_i and v_{i^*} in G with these properties. Then there exists a vertex $v_{j'}$ fulfilling $(v_{i*}, v_{j'}) \in A$ and $(v_i, v_{j'}) \notin A$. We set $A' := A \setminus \{(v_{i*}, v_{j'}), (v_i, v_j)\} \cup \{(v_i, v_{j'}), (v_{i*}, v_j)\}$ and get a dag realization possessing a larger number of maximum sources as G. Contradiction!

Claim 3: There exists a dag realization $G := (V, A)$, such that the smallest stream vertex $v_{i_{min}} \in V$ with respect to the opposed ordering in sequence S possesses only $d_G^-(v_{i_{min}})$ maximum sources in its incoming neighborhood set $N_G^-(v_{i_{min}})$.

Assume there does not exist such a dag realization. We consider a dag realization G, where the incoming neighborhood set $N_G^-(v_j)$ of a vertex stream v_j exactly contains the a_j largest sources, see Figure 4.1. By Claim 1 and 2, we can ensure the existence of such a dag G. Then we can find a vertex $v_{i_{min}}$ with $\binom{d_G^-(v_{i_{min}})}{d_G^+(i_{min})} <_{opp} \binom{d_G^-(v_j)}{d_G^+(v_j)}$, such that the incoming neighborhood set $N_G^-(v_{i_{min}})$ does not only possess sources. Otherwise, we can construct our demanded dag realization by using Claim 2.
In the next steps we show, how to build a dag realization such that $N_G^-(v_{i_{min}})$ consists of $d_G^-(v_{i_{min}})$ maximum sources by interchanging arcs and non-arcs in a set of alternating, oriented 4-cycles of G.

Construction of a digraph realization G^* with directed cycles, such that the incoming neighborhood set of $v_{i_{min}}$ consists of maximum sources and possibly contains stream vertex v_j. Let

$$k' := |\{v^-_{i_{min_l}} \in N_G^-(v_{i_{min}}) \wedge v^-_{i_{min_l}} \text{ is not a source } \} \setminus \{(v_j, v_{i_{min}})\}|$$

be the cardinality of the incoming neighborhood set of $v_{i_{min}}$ without sources, where we also ignore the *possibly* existing arc $(v_i, v_{i_{min}})$. By our assumption $d_G^-(v_{i_{min}}) \le d_G^-(v_j)$, we can conclude that there exist k' sources which are not connected with $v_{i_{min}}$ but with v_j. We choose k' maximum sources $q_1, \ldots, q_{k'}$ of this kind. Now, we construct for each $l \in \{1, \ldots, k'\}$ the alternating oriented cycle

$$(q_l, v_j, v_{i^-_{min_l}}, v_{i_{min}}, q_l),$$

with $(q_l, v_j), (v_{i^-_{min_l}}, v_{i_{min}}) \in A$, $(v_{i^-_{min_l}}, v_j), (q_l, v_{i_{min}}) \notin A$ and $v_{i^-_{min_l}} \neq v_j$. We interchange the arcs and non-arcs in these cycles and get the digraph G^*, see Figure 4.2, with the following properties:

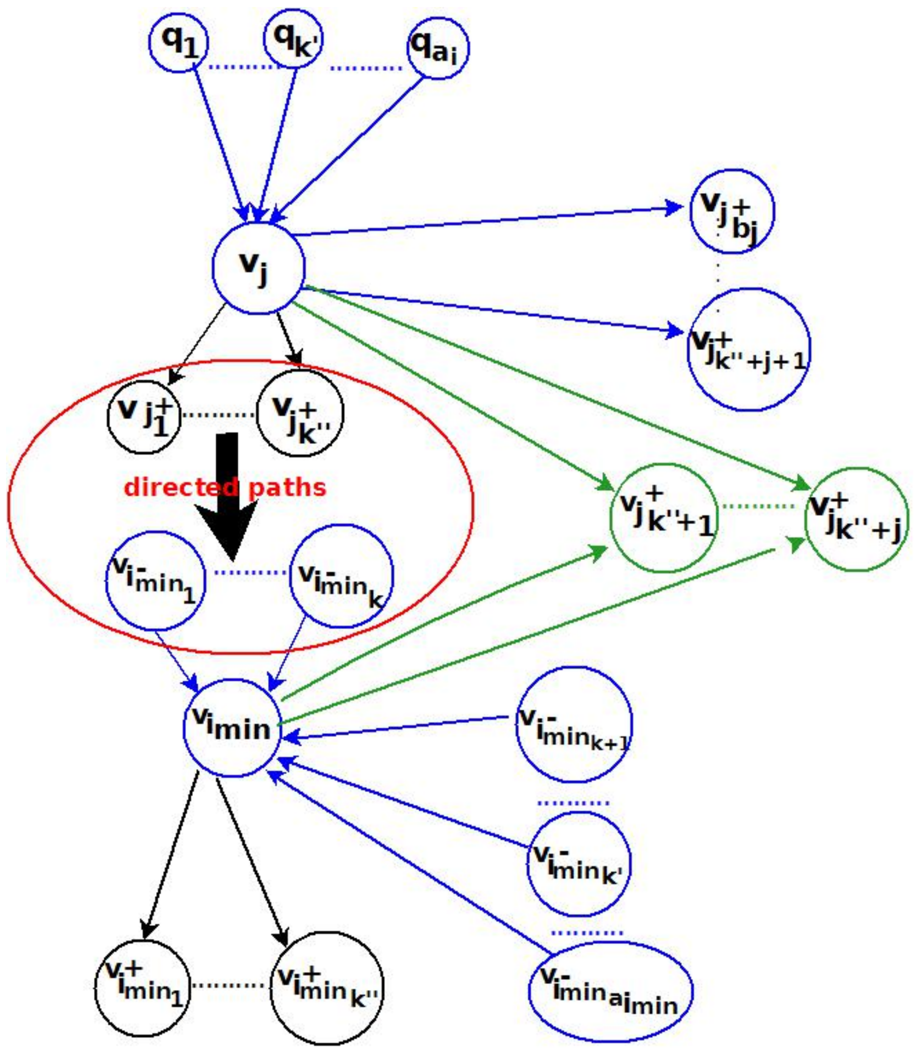


Figure 4.1: Proof of Theorem 4.2: Details of the directed acyclic digraph G.

1. There exists in G^* (except for the possibly existing arc $(v_j, v_{i_{min}})$) no directed path from v_j to $v_{i_{min}}$.

2. There may exist directed cycles $(v_{i^-_{min_l}}, v_j, v^+_{j_i}, \ldots, v_{i^-_{min_l}})$ consisting of directed paths from v_j to $v_{i^-_{min_l}}$ and arcs from $v_{i^-_{min_l}}$ to v_j, if there is in G at least one directed path from v_j to $v_{i_{min}}$, which is not the arc $(v_j, v_{i_{min}})$.

3. There are no parallel arcs in G^*, because if we assume the existence of an arc $(v_{i^-_{min_l}}, v_j)$ in G, then v_j is connected with a vertex in G which is not a source in contradiction to our assumption. Now, suppose the existence of a parallel arc $(q_l, v_{i_{min}})$ in G^*. Then it follows $(q_l, v_{i_{min}}) \in G$ in contradiction to our choice of q_l as a source which is not connected with $v_{i_{min}}$ in G.

Hence, G^* is a digraph realization, where vertex $v_{i_{min}}$ is only adjacent to sources, with the only exception that arc $(v_j, v_{i_{min}})$ could exist. All directed cycles in G^* use (see 2.) arcs $(v_j, v^+_{j_i})$. In a further step, we delete such arcs.

Construction of a dag realization G^{**} by deleting outgoing arcs of vertex v_j, lying on a directed path to $v_{i_{min}}$ in G. We define the set

$$N^+_G(v_j)(v_{i_{min}}) := \{v^+_{j_l} \mid \text{ arc } (v_j, v^+_{i_l}) \text{ uses a directed path from } v_j \text{ to } v_{i_{min}} \text{in } G\}.$$

W. l. o. g. we assume $N^+_G(v_i)(v_{i_{min}}) \neq \emptyset$, otherwise digraph G^* is the desired dag realization from S in contradiction to our assumption. We define $k'' := |N^+_G(v_j)(v_{i_{min}}) \setminus \{v_{i_{min}}\}|$. We construct a further set of alternating, oriented 4-cycles to interchange their arcs and non-arcs. We build k'' of such cycles in the following way:

$$(v_j, v_{j^+_l}, v_{i_{min}}, v_{i^+_{min_l}}, v_j),$$

where $(v_j, v_{j^+_l}), (v_{i_{min}}, v_{i^+_{min_l}}) \in A$ and $(v_{i_{min}}, v_{j^+_l}), (v_j, v_{i^+_{min_l}}) \notin A$. Furthermore, we demand $v_{j^+_l} \in N^+_G(v_j)(v_{i_{min}}) \setminus \{v_{i_{min}}\}$.

In this construction we redirect all outgoing arcs from v_j which are contained in a directed path to $v_{i_{min}}$ in G (except the arc $(v_j, v_{i_{min}})$) to vertices of the outgoing neighborhood set of $v_{i_{min}}$. On the other hand we redirect k'' outgoing arcs from $v_{i_{min}}$ to vertices of the outgoing neighborhood set of v_j which were deleted above, see Figure 4.3. If the arc $(v_j, v_{i_{min}})$ is the unique, directed path from v_j to $v_{i_{min}}$, then $k'' = 0$ and we construct no alternating, oriented 4-cycle. We want to redirect k'' outgoing arcs from v_j to incident vertices of $v_{i_{min}}$ and k'' outgoing arcs from $v_{i_{min}}$ to incident vertices of v_j without constructing parallel arcs. The problematic vertices v_j for this case are

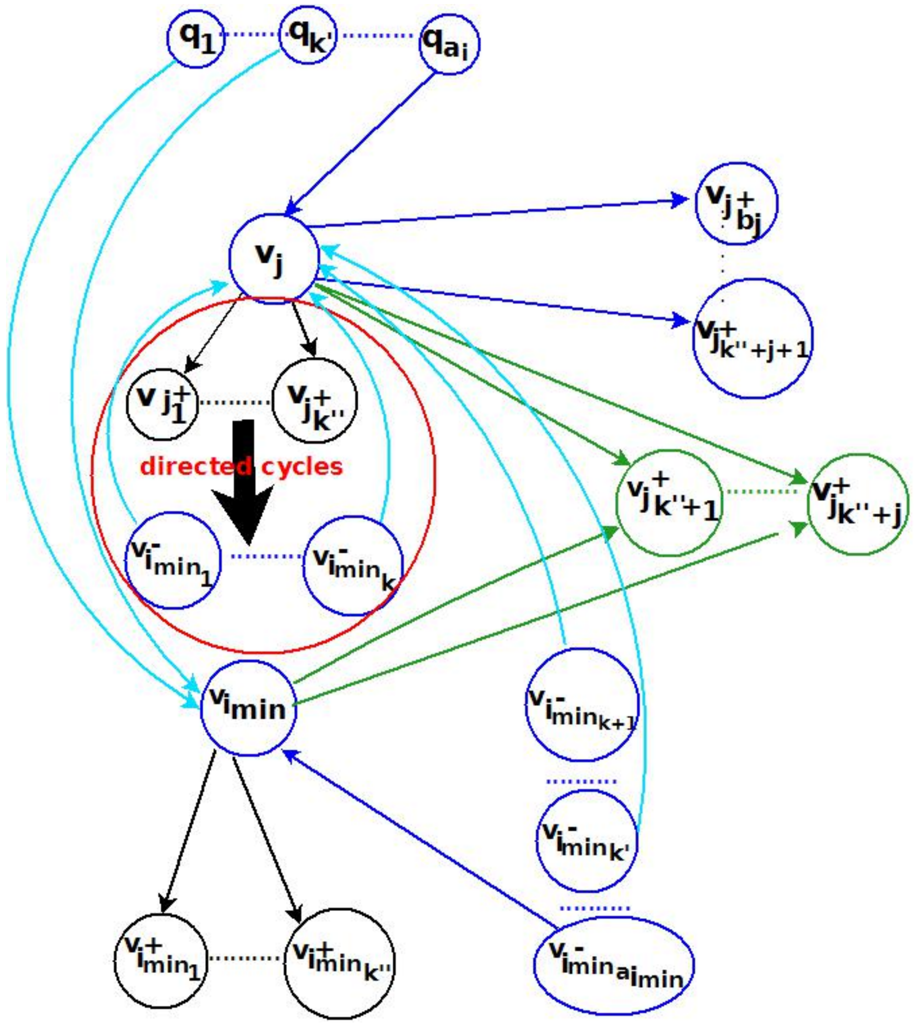


Figure 4.2: Proof of Theorem 4.2: G^* with directed cycles.

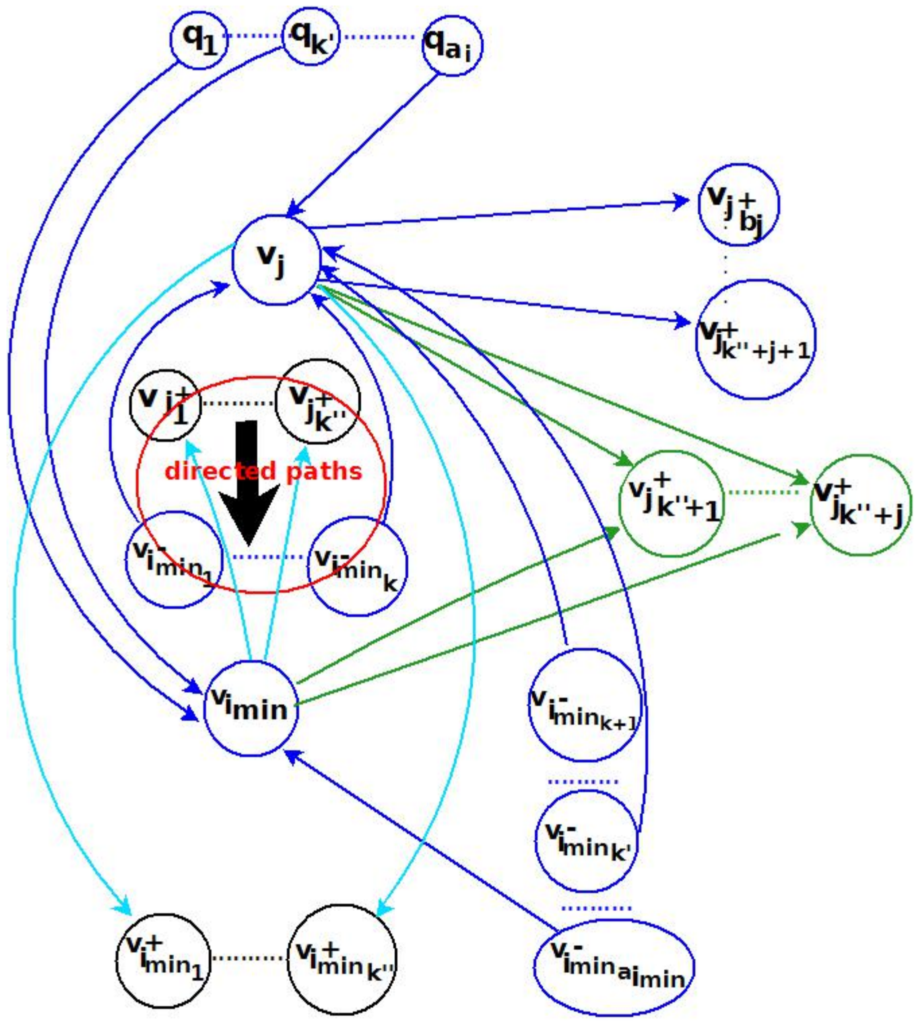


Figure 4.3: Proof of Theorem 4.2: Digraph G^{**} without directed cycles.

contained in the outgoing neighborhood set $N^+_{G^*}(v_j) := \{v_k|\ (v_j, v_k) \in G^*\}$ from v_j and the outgoing neighborhood set $N^+_{G^*}(v_{i_{min}}) := \{v_k|\ (v_{i_{min}}, v_k) \in G^*\}$ from $v_{i_{min}}$. We consider two possibilities. Either the set $N^+_{G^*}(v_{i_{min}}) \setminus N^+_{G^*}(v_j)$ consists of k'' elements — in the case $(v_j, v_{i_{min}}) \notin A$ — or it consists of $k''+1$ elements, if $(v_j, v_{i_{min}}) \in A$. We consider the set $N^+_{G^*}(v_j) \cap N^+_{G^*}(v_{i_{min}})$. A vertex v_k of this set is not contained in $N^+_G(v_j)(v_{i_{min}})$. Otherwise, there exists in G a directed cycle (v_j, v_k, v_j). We exploit $d^+_G(v_j) \leq d^+_G(v_{i_{min}})$ and get

$$\begin{aligned} |N^+_{G^*}(v_j) \cap N^+_{G^*}(v_{i_{min}})| &\leq |N^+_{G^*}(v_j)| - |N^+_{G^*}(v_j)(v_{i_{min}})| \\ &\leq |N^+_{G^*}(v_{i_{min}})| - |N^+_{G^*}(v_j)(v_{i_{min}})|. \end{aligned}$$

Then it follows

$$\begin{aligned} |N^+_{G^*}(v_{i_{min}}) \setminus N^+_{G^*}(v_j)| &= |N^+_{G^*}(v_{i_{min}})| - |N^+_{G^*}(v_j) \cap N^+_{G^*}(v_{i_{min}})| \\ &\geq |N^+_{G^*}(v_j)(v_{i_{min}})|. \end{aligned}$$

Now, we distinguish between two cases. For $(v_j, v_{i_{min}}) \notin A^*$, we get $k'' = |N^+_{G^*}(v_j)(v_{i_{min}})|$ and for $(v_j, v_{i_{min}}) \in A^*$, it follows $k''+1 = |N^+_{G^*}(v_j)(v_{i_{min}})|$. We interchange the arcs and non-arcs in all k'' oriented, alternating 4-cycles and get digraph G^{**} with the demanded vertex degree sequence.

Deleting the arc $(v_j, v_{i_{min}})$ with the aid of an alternating oriented 6-cycle. There exists at least one source q, which is not adjacent with $v_{i_{min}}$ but with v_j, because condition $d^-_G(v_{i_{min}}) \leq d^-_G(v_j)$ is valid. We construct the alternating, oriented 6-cycle $(q, v_j, v_{i_{min}}, v_{i^+_{min}}, v_j, v_{i_{min}}, q)$ with $(v_j, v_{i_{min}}), (v_{i^+_{min}}, v_j), (v_{i_{min}}, q) \notin A^{**}$, and $v_{i^+_{min}} \in N^+_{G^*}(v_{i_{min}}) \setminus N^+_{G^*}(v_j)$. Such a vertex exists, because we proved the existence of $k''+1$ vertices of this type. We interchange the arcs and non-arcs in this cycle and get a digraph with the demanded vertex degree sequence. For simplicity, we call this digraph still G^{**}. Then digraph G^{**} possesses the following properties.

1. There do not exist arcs $(v_j, v_{j^+_l})$ with $v_{j^+_l} \in N^+_{G^*}(v_j)(v_{i_{min}}) \setminus \{v_{i_{min}}\}$ in G^{**}. Hence, G^{**} does not contain a directed cycle $(v^-_{i_{min}}, v_j, v_{j^+_l}, \ldots, v^-_{i_{min}})$.

2. There are no parallel arcs $(v_{i_{min}}, v_{j^+_l})$ with $v_{j^+_l} \in N^+_{G^*}(v_j)(v_{i_{min}})$, as otherwise such an arc is already contained in G. By our assumption $v^+_{jl} \in N^+_{G^*}(v_j)(v_{i_{min}})$, we get a directed cycle in G. Contradiction!

3. In G^{**} there exists no directed path from $v_{j_l^+}$ to $v_{i_{min}}$, because in G we deleted all incoming arcs from $v_{i_{min}}$, which are contained in such a path. Hence, an arc $(v_{i_{min}}, v_{j_l^+})$ cannot close a directed cycle in G^{**}.

4. The special choice of arc $(v_j, v^+_{i_{min}})$ in G^* with $v^+_{i_{min}} \in N^+_{G^*}(v_{i_{min}}) \backslash N^+_{G^*}(v_j)$ avoids to construct parallel arcs.

5. Assume there are directed paths from $v^+_{i_{min}}$ to v_j in G^{**}. Then their existence in G is necessary, because we only added arcs $(v_{i_{min}}, v^+_{j_l})$ and $(v_{j^-_{min_l}}, v_j)$ in the direction of v_j. For the construction of such a directed path, we need the existence of the arcs $(v^+_{j_l}, v_j)$ or $(v_{i_{min}}, v^-_{i_{min_l}})$ in G^{**} which then are also contained in G. This is a contradiction to our assumption that G is an acyclic digraph. Hence, the demanded path already exists in G. We get $N^+_{G^*}(v_j)(v_{i_{min}}) = \emptyset$ in contradiction to our assumption.

We can conclude, that digraph G^{**} is a dag realization of S and the stream vertex with the smallest index with respect to an opposed ordering has an incoming neighborhood set which only contains sources. By Claim 2, there exists a dag realization G^{***} where the incoming neighborhood set only consists of maximum sources. We delete arcs connecting sources with vertex $v_{i_{min}}$ and get dag G' with vertex degree sequence S'. $\square$

This theorem reduces an opposed dag sequence until we get a source-sink-sequence S'. By applying Theorem 3.1 for digraph realizations, we can realize sequence S'. Algorithm 1 combines the insights of both theorems and constructs a dag realization for any opposed sequence which is realizable.

We give an example.

Example 4.1. *Given is the increasing opposed sequence* $S := \binom{a_1}{b_1} = \binom{0}{2}, \binom{a_2}{b_2} = \binom{0}{1}, \binom{a_3}{b_3} = \binom{2}{3}, \binom{a_4}{b_4} = \binom{2}{2}, \binom{a_5}{b_5} = \binom{2}{1}, \binom{a_6}{b_6} = \binom{1}{0}, \binom{a_7}{b_7} = \binom{2}{0}$. *We apply the while loop in line 3 of Algorithm 1 and get iteratively the following sequences:*
step 1: $\binom{a_1}{b_1} = \binom{0}{1}, \binom{a_3}{b_3} = \binom{0}{3}, \binom{a_4}{b_4} = \binom{2}{2}, \binom{a_5}{b_5} = \binom{2}{1}, \binom{a_6}{b_6} = \binom{1}{0}, \binom{a_7}{b_7} = \binom{2}{0}$
step 2: $\binom{a_3}{b_3} = \binom{0}{2}, \binom{a_4}{b_4} = \binom{0}{2}, \binom{a_5}{b_5} = \binom{2}{1}, \binom{a_6}{b_6} = \binom{1}{0}, \binom{a_7}{b_7} = \binom{2}{0}$
step 3: $\binom{a_3}{b_3} = \binom{0}{1}, \binom{a_4}{b_4} = \binom{0}{1}, \binom{a_5}{b_5} = \binom{0}{1}, \binom{a_6}{b_6} = \binom{1}{0}, \binom{a_7}{b_7} = \binom{2}{0}$
step 4: $\binom{a_3}{b_3} = \binom{0}{1}, \binom{a_4}{b_4} = \binom{0}{1}, \binom{a_5}{b_5} = \binom{0}{1}, \binom{a_6}{b_6} = \binom{1}{0}, \binom{a_7}{b_7} = \binom{2}{0}$
Now, we have a source-sink-sequence and jump to line 15. We get
$\binom{a_4}{b_4} = \binom{0}{1}, \binom{a_5}{b_5} = \binom{0}{1}, \binom{a_6}{b_6} = \binom{1}{0}, \binom{a_7}{b_7} = \binom{1}{0}$ $\binom{a_5}{b_5} = \binom{0}{1}, \binom{a_7}{b_7} = \binom{1}{0}$. *These steps lead to the arc set*
$A := \{(v_1, v_3), (v_2, v_3), (v_1, v_4), (v_3, v_4), (v_3, v_5), (v_4, v_5), (v_3, v_7), (v_4, v_6), (v_5, v_7)\}$ *of a dag realization* $G = (V, A)$.

Algorithm 1 Dag Realization Algorithm for Opposed Sequences

```
Require: An opposed sequence $S$ in increasing opposed order.
Ensure: A dag realization $G = (V, A)$ of $S$ or the answer that $S$ is not a dag sequence.
1: initialize $A \leftarrow \emptyset$;
2: while (the set of stream tuples in $S$ is not empty) do
3:   {application of Theorem 4.2}
4:   choose the stream tuple $\binom{a_j}{b_j}$ with smallest index $j$;
5:   if the number of sources in $S$ is smaller than $a_j$ then
6:     return FALSE;
7:   else
8:     set $b_i \leftarrow b_i - 1$ for the $a_j$ largest sources $\binom{0}{b_{l_1}}, \dots, \binom{0}{b_{l_{a_j}}}$;
9:     set $a_j \leftarrow 0$;
10:    set $A \leftarrow A \cup \{(v_{l_1}, v_j), (v_{l_2}, v_j), \dots, (v_{l_{a_j}}, v_j)\}$;
11:    delete $\binom{0}{0}$-tuples in $S$;
12:  end if
13: end while
14: while (the set of source tuples in $S$ is not empty) do
15:   {realization of a source-sink-sequence.}
16:   choose a largest source tuple $\binom{0}{b_j}$;
17:   if the number of sinks in $S$ is smaller than $b_j$ then
18:     return FALSE;
19:   else
20:     set $a_i \leftarrow a_i - 1$ for the $b_j$ largest sinks $\binom{a_{k_1}}{0}, \dots, \binom{a_{k_{b_j}}}{0}$;
21:     set $A \leftarrow A \cup \{(v_j, v_{k_1}), (v_j, v_{k_2}), \dots, (v_j, v_{k_{b_j}})\}$;
22:     delete $\binom{0}{0}$-tuples in $S$;
23:   end if
24: end while
25: return True;
```

Using bucket sort, the reordering of a given opposed sequence such that it is in increasing opposed order requires $O(m+n)$ time. Algorithm 1 can also be implemented to run in time $O(m+n)$ using a "bucket" technique. The idea is to maintain the source and sink tuples in buckets (realized as linked lists), one bucket for each outdegree and indegree, respectively, thus with at most $2n-2$ buckets. Non-empty buckets are ordered decreasingly, and each bucket has a pointer to its two neighbors in this order. Moreover, we maintain three pointers, one to the non-empty bucket corresponding to the source with the largest outdegree, one to the sink with largest indegree, and one to the source with the smallest outdegree. The stream tuples are kept in a list, sorted according to the increasing opposed order. Thus line 4 can be done in $O(1)$. With our data structure, it is easy to execute line 8 in $O(a_j)$ time, that is, to select the a_j largest source tuples, to decrease their outdegree by one and to update our data structure (which means in particular to shift tuples from their current bucket to the next lower bucket or to delete them). In line 9, the selected stream tuple with index j becomes a new source and is inserted into our bucketing data structure. This can also be done in $O(a_j)$ time. Likewise, in line 16 choosing the largest source tuple can be done in $O(1)$ and line 20 in $O(b_j)$. In total, this yields $O\left(\sum_{i=1}^{n}(a_i+b_i)\right) = O(m+n)$ time.

We point out, that there is the possibility to generalize Theorem 4.2 and to formulate an algorithm handling the realization of all sequences. The problem is, that the opposed order is not a total order. Therefore, there does not exist a unique minimal stream tuple in each sequence. Hence, we get a realization algorithm but destroy the polynomial running time in general. More details are given in Section 4.1.2. Theorem 4.2 leads to a further property connected with topological orderings. Let us consider a well-known algorithm determining a topological ordering of a dag. The iterative deletion of one current source results in a labeling of vertices representing a feasible topological sorting for a dag. Applying our theorem we can conclude the following: Each stream tuple $\binom{a_{i+i_{min}}}{b_{i+i_{min}}}$ with $i \in \{0, \dots, n_s-1\}$ (n_s denotes the number of stream tuples) is a minimum stream tuple in the ith iteration of the while statement in line 2. Clearly, tuple $\binom{a_{i+i_{min}}}{b_{i+i_{min}}}$ is in the $(i+1)$th iteration of this loop a new source $\binom{0}{b_{i+i_{min}}}$ (under the restriction that the sequence is not yet a source-sink-sequence). That means, if we consider a dag realization G which was constructed by Algorithm 1, then we could remove sources following the increasing opposed sorting. This yields the following corollary.

Corollary 4.1 (topological sorting)**.** *An increasing, opposed sorting of an opposed dag sequence S is a topological sorting of at least one dag realization of S.*

Back to our Example 4.1 we can indeed observe that the labeling $v_1, \dots, v_7$ is a topological sorting of our dag realization. This new view enables us to see the dag realization problem in the case of opposed sequences as *dag realization problem for a given topological order*. Hence, our algorithm works similar as in Theorem 4.1. On the other hand, we have a necessary criterion for determining the realizability of opposed sequences.

4.1.2 General Dag Realization

In Section 4.4, we introduced a realization algorithm for opposed sequences. Its underlying ideas can be extended to the general case – the realization of arbitrary dag sequences. This new algorithm does not necessarily possess a polynomial running time. On the other hand many dag sequences are realizable in polynomial running time using the following Theorem 4.3 and a further "strategy". We call a sequence $S = \binom{a_1}{b_1}, \dots, \binom{a_n}{b_n}$ with q source tuples and s sink tuples *canonically sorted*, if and only if the first q tuples in this labelling are decreasingly sorted source tuples (with respect to the b_i) and the last s tuples are increasingly sorted sink tuples (with respect to the a_i.)

Theorem 4.3 (Characterization of dag realizability)**.**
Let S be a canonically sorted sequence containing $k > 0$ source tuples. Furthermore, we assume that S is not a source-sink-sequence. We define the set

$$V_{min} := \left\{ \begin{pmatrix} a_i \\ b_i \end{pmatrix} \mid \begin{pmatrix} a_i \\ b_i \end{pmatrix} \text{ is a stream tuple}, a_i \leq k \text{ and there is no stream tuple } \begin{pmatrix} a_j \\ b_j \end{pmatrix} <_{opp} \begin{pmatrix} a_i \\ b_i \end{pmatrix} \right\}.$$

S is a dag sequence if and only if $V_{min} \neq \emptyset$ and there exists an element $\binom{a_{i_{min}}}{b_{i_{min}}} \in V_{min}$ such that

$$S' := \binom{0}{b_1 - 1}, \dots, \binom{0}{b_{a_{i_{min}}} - 1}, \binom{0}{b_{a_{i_{min}}+1}}, \dots, \binom{0}{b_k}, \dots, \binom{a_{i_{min}-1}}{b_{i_{min}-1}}, \binom{0}{b_{i_{min}}}, \binom{a_{i_{min}+1}}{b_{i_{min}+1}}, \dots, \binom{a_n}{b_n}$$

is a dag sequence.

Proof. $\Leftarrow$: Let S, S' be a dag sequences with a numbering corresponding to our definitions and let $V_{min} \neq \emptyset$. By construction of set V_{min} we conclude the existence of at least $a_{i_{min}}$ source tuples. We consider a dag realization $G' = (V, A')$ of S' and insert further arcs $(v_1, v_{i_{min}}), \dots, (v_{a_{i_{min}}}, v_{i_{min}})$. We call the new constructed digraph G. Obviously, digraph G is a dag, because all inserted arcs starting at a source cannot build

a directed cycle. On the other hand, G is a digraph realization of S, because conditions $d^+_G(v_i) = b_i$ and $d^-_G(v_i) = a_i$ are fulfilled for all $i \in \{1, \dots, n\}$.
$\Rightarrow$: Let S be a dag sequence with $k \geq 1$ source tuples. Furthermore, we consider a dag realization G of S. The following steps of our proof are analogously to that of Theorem 4.2. For this reason we refer to its details in most cases. First, we show $V_{min} \neq \emptyset$.

Claim 1: There exists a stream vertex v_j with $d^-_G(v_j) \leq k$, such that the incoming neighborhood set $N^-_G(v_j)$ only contains sources. It follows $V_{min} \neq \emptyset$.

Assume this is not the case. We delete all k sources and its adjacent arcs in dag G. By our assumption, we get a digraph without sources — a contradiction to the acyclicity of G. Hence, there is a stream vertex v_j with an incoming neighborhood set $N^-_G(v_j)$ which only contains sources. Then it follows $d^-_G(v_j) = a_j \leq k$. We distinguish between two possibilities. Either, we have $\binom{a_j}{b_j} \in V_{min}$ or there is a minimum stream tuple $\binom{a_l}{b_l}$ with $\binom{a_l}{b_l} <_{opp} \binom{a_j}{b_j}$ with respect to the opposed ordering. Then we get $\binom{a_l}{b_l} \in V_{min}$. Hence, it follows $V_{min} \neq \emptyset$.

Claim 2: There exists a dag realization $G := (V, A)$ and a stream vertex $v_j \in V$, such that the incoming neighborhood set $N^-_G(v_j)$ only contains $d^-_G(v_j)$ maximum sources.

The proof can be done analogously to that one of Claim 2 in Theorem 4.2.

Claim 3: There exists a dag realization $G := (V, A)$ and a stream vertex $v_{i_{min}} \in V_{min}$ such that the incoming neighborhood set $N^-_G(v_{i_{min}})$ only contains $d^-_G(v_{i_{min}})$ maximum sources.

Assume there is no such dag realization. By Claim 2, we can conclude the existence of a stream vertex $v_j \notin V_{min}$ with an incoming neighborhood set $N^-_G(v_j)$ which only contains $d^-_G(v_j)$ maximum sources. Then there is a stream vertex $v_{i_{min}} \in V_{min}$ with $\binom{d^-_G(v_{i_{min}})}{d^+_G(i_{min})} <_{opp} \binom{d^-_G(v_j)}{d^+_G(v_j)}$ and an incoming neighborhood set *not* only containing sources. Otherwise, we construct by exploiting Claim 2 the desired dag realization G. Now, we jump in the proof of Claim 3 of Theorem 4.2 and get a contradiction to our assumption. Hence, it exists a dag realization G^{**} with a stream vertex $v_{i_{min}} \in V_{min}$ such that the incoming neighborhood set $N^-_G(v_j)$ only contains $d^-_G(v_{i_{min}})$ sources. By Claim 2, we conclude the existence of a dag realization G^{***} such that vertex $v_{i_{min}}$ only has maximum sources as incoming neighborhood set. We delete all incoming arcs of vertex $v_{i_{min}}$ and get a dag G' with vertex degree sequence S'. □

This theorem gives us further interesting insights. Consider the relation $R_{top} \subseteq V \times V$, where we define: $vR_{top}w$ if and only if there exists no directed path starting in w and ending in v. For each dag sequence we can find a dag realization with a topological ordering such that we have for each pair of tuples with $v_i R_{top} v_j$ either $\binom{a_i}{b_i} \leq_{opp} \binom{a_j}{b_j}$ or they are not comparable but it never follows $\binom{a_j}{b_j} <_{opp} \binom{a_i}{b_i}$. Note, that this is not a general property for acyclic digraphs. Consider for example digraph $G = (V, A)$ with $V := \{v_1, \dots, v_7\}$ and $A = \{(v_1, v_3), (v_2, v_3), (v_3, v_4), (v_4, v_5), (v_4, v_6), (v_4, v_7)\}$. The corresponding stream tuples of $v_3 R_{top} v_4$ are $\binom{a_3}{b_3} = \binom{2}{1}$ and $\binom{a_4}{b_4} = \binom{1}{3}$ with $\binom{a_4}{b_4} <_{opp} \binom{a_3}{b_3}$. A further interesting observation is that this property is symmetrical in the following sense. Note, that for two pairs $\binom{a_1}{b_1} <_{opp} \binom{a_2}{b_2}$ we either have $a_1 < a_2 \wedge b_1 \geq b_2$ or $a_1 \leq a_2 \wedge b_1 > b_2$. Hence, we can conclude $\binom{b_2}{a_2} <_{opp} \binom{b_1}{a_1}$. On the other hand it is clear that the non-comparability of two pairs with respect to the opposed relation is symmetrical. If we change in a sequence $S = \binom{a_1}{b_1}, \dots, \binom{a_n}{b_n}$ the roles of a_i and b_i, then we get a sequence $\overline{S} = \binom{b_1}{a_1}, \dots, \binom{b_n}{a_n}$ such that our property is fulfilled in the opposite direction. Hence, this ordering gives us a possibility to include the underlying "symmetry of sinks and sources".

Corollary 4.2. *Let S be a dag sequence. Then there exists a dag realization $G = (V, A)$ with a topological ordering $v_{l_1}, \dots, v_{l_{n_s}}$ of all n_s vertices corresponding to stream tuples, such that for $l_i < l_j$ we cannot find $\binom{a_{l_j}}{b_{l_j}} <_{opp} \binom{a_{l_i}}{b_{l_i}}$.*

Proof. Assume, S is a dag sequence and for all dag realizations we have for each topological ordering $v_{l_1}, \dots, v_{l_{n_s}}$ of all vertices corresponding to stream tuples at least one pair $v_{l_i} R_{top} v_{l_j}$ with $\binom{a_{l_j}}{b_{l_j}} <_{opp} \binom{a_{l_i}}{b_{l_i}}$. Consider a dag realization $G = (V, A)$, which was constructed by repeated application of Theorem 4.3. Hence, we applied this Theorem exactly l_{n_s} times. On the other hand, we can conclude $\binom{a_{l_i}}{b_{l_i}} \in V_{min}$ in the i–th step. But then we get $\binom{a_{l_i}}{b_{l_i}} <_{opp} \binom{a_{l_j}}{b_{l_j}}$ with respect to the definition of V_{min}. Contradiction! $\square$

Theorem 4.3 proves the existence of a dag realization which contains a vertex corresponding to an element of V_{min} only possessing maximum sources as incoming neighborhood set. We define a largest possible subset $V'_{min} \subseteq V_{min}$ with the property that all elements in V'_{min} are pairwise distinct (w. l. o. g. we can restrict our candidate set to V'_{min}). For each $\binom{a_{i_j}}{b_{i_j}} \in V'_{min}$ we construct the reduced sequence S'_j, set $S_j := S'_j$ and apply our Theorem 4.3 repeatedly until S'_j is a source-sink-sequence or we find out that S'_j is not a dag sequence, because V_{min} is empty.

Hence, Theorem 4.3 ensures the possibility for reducing a dag sequence into a source-sink-sequence. The latter can be realized by using Theorem 3.1. The whole algorithm is summarized in Algorithm 2. We give an example in Figure 4.4.

The bottleneck of this approach is the size of set V'_{min}. We only find stream tuples in this set which are not comparable with respect to the opposed relation. In an opposed dag sequence we have $|V'_{min}| = 1$, if sequence S is not a source-sink-sequence, because in this case there exists a unique minimum stream tuple (up to isomorphic ones). Hence, Theorem 4.2 is a special case of our Theorem 4.3. However, there are many sequences which are not opposed but Theorem 4.3 still yields a polynomial time execution of Algorithm 2. Consider for example dag sequence $S := \binom{0}{3}, \binom{0}{3}, \binom{2}{2}, \binom{3}{3}, \binom{1}{0}, \binom{2}{0}, \binom{3}{0}$ which is not an opposed sequence, because stream tuples $\binom{2}{2}$ and $\binom{3}{3}$ are not comparable with respect to the opposed ordering. However, we have $|V'_{min}| = |\{\binom{2}{2}\}| = 1$ and so we reduce S to $S' = \binom{0}{2}, \binom{0}{2}, \binom{0}{2}, \binom{3}{3}, \binom{1}{0}, \binom{2}{0}, \binom{3}{0}$, leading to the realizable source-sink-sequence $\binom{0}{1}, \binom{0}{1}, \binom{0}{1}, \binom{0}{3}, \binom{1}{0}, \binom{2}{0}, \binom{3}{0}$.

At the beginning of our work (when the complexity of the dag realization problem was still open), we conjectured that the choice of the lexicographical largest tuple from V'_{min} in line (3) of Algorithm 2 would solve our problem in polynomial time. We call this approach *lexmax strategy* and a dag sequence which is realizable with this strategy *lexmax sequence*, otherwise we call it *non-lexmax sequence.* We conjectured the following.

Conjecture 4.1 (lexmax conjecture)**.** *Each dag sequence is a lexmax sequence.*

We disproved our own conjecture by a counter-example, see our example in Figure 4.4. The green marked path corresponds to the lexmax strategy, but is unsuccessful. More interesting is the insight, that there is *no general strategy* for choosing a correct element in V'_{min} *without the consideration of all sinks.* To see this, consider the following example.

Example 4.2. *We consider the two sequences*

$$S_1 := \begin{pmatrix}0\\5\end{pmatrix}, \begin{pmatrix}0\\5\end{pmatrix}, \begin{pmatrix}0\\5\end{pmatrix}, \begin{pmatrix}0\\2\end{pmatrix}, \begin{pmatrix}0\\2\end{pmatrix}, \begin{pmatrix}5\\5\end{pmatrix}, \begin{pmatrix}5\\5\end{pmatrix}, \begin{pmatrix}2\\2\end{pmatrix}, \begin{pmatrix}2\\2\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}2\\0\end{pmatrix}, \begin{pmatrix}6\\0\end{pmatrix}, \begin{pmatrix}9\\0\end{pmatrix}$$

and

$$S_2 := \begin{pmatrix}0\\5\end{pmatrix}, \begin{pmatrix}0\\5\end{pmatrix}, \begin{pmatrix}0\\5\end{pmatrix}, \begin{pmatrix}0\\2\end{pmatrix}, \begin{pmatrix}0\\2\end{pmatrix}, \begin{pmatrix}5\\5\end{pmatrix}, \begin{pmatrix}5\\5\end{pmatrix}, \begin{pmatrix}2\\2\end{pmatrix}, \begin{pmatrix}2\\2\end{pmatrix}, \begin{pmatrix}6\\0\end{pmatrix}, \begin{pmatrix}6\\0\end{pmatrix}, \begin{pmatrix}7\\0\end{pmatrix},$$

Algorithm 2 DagRealization(sequence S)

```
Require: A canonically sorted sequence $S$.
Ensure: A Boolean flag indicating whether $S$ is realizable.
 1: if $S$ is not a source-sink-sequence then
 2:    Count the number of sources in $S$ and determine set $V'_{min}$.
 3:    for all $\binom{a_j}{b_j} \in V'_{min}$ do
 4:       Create a working copy $S'$ of $S$ with tuples $\binom{a'_i}{b'_i} = \binom{a_i}{b_i}$;
 5:       Set $b'_i \leftarrow b'_i - 1$ for $a'_j$ largest sources $\binom{0}{b'_i}$.
 6:       Set $a'_j \leftarrow 0$.
 7:       Delete $\binom{0}{0}$-tuples.
 8:       if DagRealization($S'$) then
 9:          return TRUE;
10:       end if
11:    end for
12:    return FALSE;
13: else
14:    while the set of source tuples in $S$ is not empty do
15:       {Realization of a source-sink-sequence.}
16:       choose a largest source tuple $\binom{0}{b_j}$.
17:       if number of sinks in $S$ is smaller than $b_j$ then
18:          return FALSE;
19:       end if;
20:       Set $a_i \leftarrow a_i - 1$ for $b_j$ largest sinks $\binom{a_i}{0}$.
21:       Delete $\binom{0}{0}$-tuples.
22:    end while
23:    return TRUE;
24: end if
```

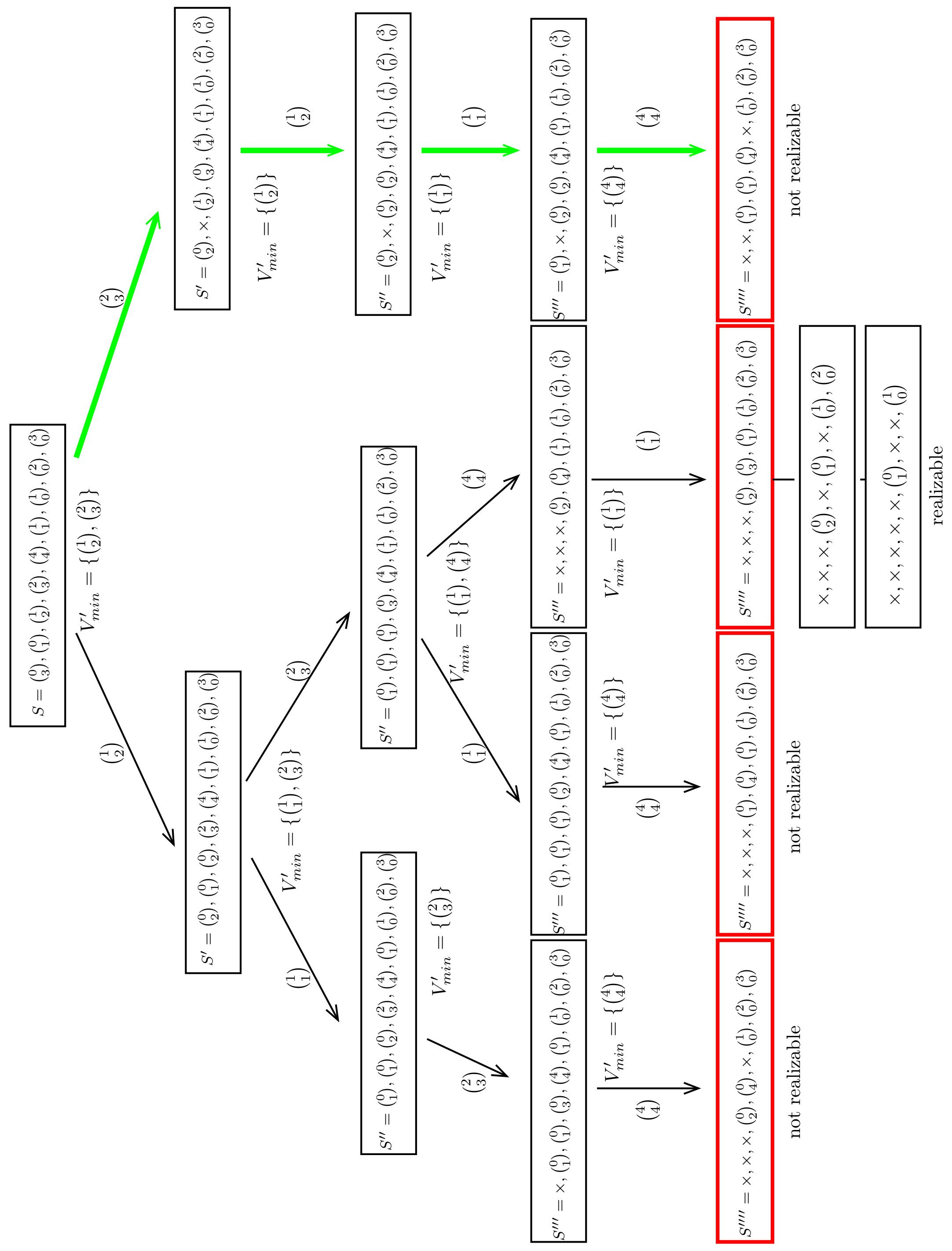


Figure 4.4: Recursion tree for Example 4.4. The symbol $\times$ here denotes tuples of S which have been deleted after being reduced to $\binom{0}{0}$. The forth tree level where the original sequence is reduced to a source-sink sequence is marked with red boxes.

only differing in their sink tuples. Dag sequence S_2 only possess one unique topological sorting of the stream tuples $\binom{5}{5}, \binom{5}{5}, \binom{2}{2}, \binom{2}{2}$ which is the lexmax strategy. In contrast, dag sequence S_1 has many topological possible sortings of the stream tuples but not the lexmax strategy. Hence, there does not exist one common strategy for these sequences.

Obviously, a more "efficient" algorithm has to consider the kind of sinks, otherwise it is not possible to choose the right element in V'_{min}. However, in a series of systematic experiments (see Section 4.2) for all dag sequences with $7, 8, 9$ tuples, we observed that our lexmax strategy determines a dag realization for a large fraction of all dag sequences.

Moreover, we can also prove that this strategy is possible for sequences with $\sum_{i=1}^{n} a_i \leq n - 1$. Such sequences are not necessarily opposed what can be seen for the dag sequence $S := \binom{0}{1}, \binom{0}{1}, \binom{2}{2}, \binom{1}{1}, \binom{1}{0}, \binom{1}{0}$. We denote these sequences by *forest sequences.* The existence of an efficient solution for forest sequences is not so surprising as there is a simple approach to construct a dag realization if there is one. First, one can apply a digraph realization algorithm. When we do not find a digraph realization, there also cannot be a dag realization. Assume, we have a digraph realization $G = (V, A)$. If G does not possess a directed cycle, then it is a dag realization and we are ready. Let us assume G has at least one directed cycle. In this case, there exist at least two weak components, because the underlying undirected graph is not a forest. Hence, we can choose an arc (v_1, v_2) of the directed cycle in the first component and a further arc (v_3, v_4) from the second weak component. We construct the new digraph $G' := (V, A')$ with $A' := A \setminus \{(v_1, v_2), (v_3, v_4)\} \cup \{(v_1, v_4), (v_3, v_2)\}$. We apply a sequence of such steps (at most n steps) until we get an acyclic dag realization. This is possible because as long as we can find a directed cycle we also have more than one weak component. (Note, that the underlying graph is not necessarily a simple graph. It can contain parallel edges corresponding to directed 2-cycles of the initial dag realization G.) Hence, we can conclude that each forest sequence which is a digraph sequence is also a dag sequence. Clearly, this can be decided in polynomial time. Indeed, we found out that each strategy to choose a tuple in V'_{min} is successful, see Corollary 4.3 below. First we give a more general result.

Theorem 4.4 (Realization of forest sequences).
Let $S := \binom{a_1}{b_1}, \ldots, \binom{a_n}{b_n}$ be a canonically sorted forest sequence containing $k > 0$ source tuples. Furthermore, we assume that S is not a source-sink-sequence. Consider an

arbitrary stream tuple $\binom{a_i}{b_i}$ with $a_i \leq k$.
S is a dag sequence if and only if

$$S' := \binom{0}{b_1 - 1}, \dots, \binom{0}{b_{a_i} - 1}, \binom{0}{b_{a_i+1}}, \dots, \binom{0}{b_k}, \dots, \binom{a_{i-1}}{b_{i-1}}, \binom{0}{b_i}, \binom{a_{i+1}}{b_{i+1}}, \dots, \binom{a_n}{b_n}$$

is a dag sequence.

Proof. $\Leftarrow$: Trivial.
$\Rightarrow$: Let S be a dag sequence with $k \geq 1$ source tuples. We consider a dag realization G such that we have a minimum number of weak components. Clearly, the underlying undirected graph is then a forest without undirected cycles. Furthermore, we consider a dag realization G as described where the incoming neighborhood set of vertex v_i consists of a maximum possible number of sources. Assume, there is a vertex $v_{i^-} \in N_G^-(v_i)$ which is not a source. Then we can conclude that there exists a source q and a vertex v_j with $(q, v_j) \in A$ but $(q, v_i) \notin A$, because we have $d_G^-(v_i) = a_i \leq k$ by our assumption. We distinguish between two cases.

case 1: There exists no underlying undirected path between vertices v_j and v_i.

case 2: There exists exactly one underlying undirected path P between vertices v_j and v_i.

Note, that there cannot be more than one underlying undirected path, because the underlying graph of G is by our assumption a forest. Let us start with case 1. There cannot be an underlying undirected path between vertices q and v_i, otherwise we would find the excluded undirected path between v_i and v_j, because q is adjacent to v_j. We construct the dag $G' = (V, A')$ with $A' := A \setminus \{(q, v_j), (v_{i^-}, v_i)\} \cup \{(q, v_i), (v_{i^-}, v_j)\}$. Consider Figure 4.5. G' is a dag, because G does not contain an underlying undirected path between v_i and q and not between v_{i^-} and v_j by our assumptions. Hence, we did not construct an underlying undirected cycle and clearly no directed cycle.

But then G' is a dag realization with a minimum number of weak components and a larger number of sources in the neighborhood set of v_i than in dag G. Contradiction! It remains to consider case 2. Since vertex v_i is a stream tuple we define the following dag $G' = (V, A')$ with $A' := A \setminus \{(q, v_j), (v_{i^-}, v_i), (v_i, v_{i^+})\} \cup \{(q, v_i), (v_{i^-}, v_{i^+}), (v_i, v_j)\}$ as can be seen in Figure 4.6.

Note, that v_j and v_{i^-} are not necessarily distinct vertices. In this case we replace in A' vertex v_{i^-} by v_j. Since we destroyed by our construction all underlying unique paths in G between v_i and q, between v_{i^+} and v_{i^-} and between v_j and v_i, digraph

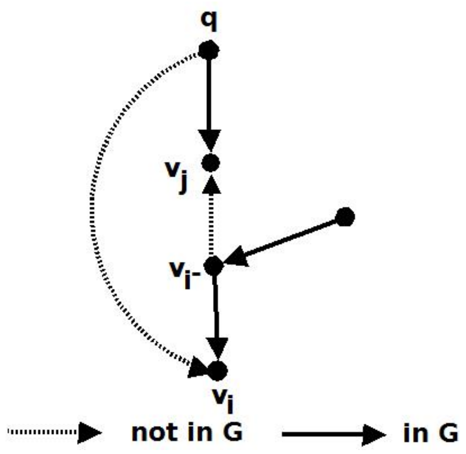


Figure 4.5: Case 1: no underlying undirected path between q and v_i in G.

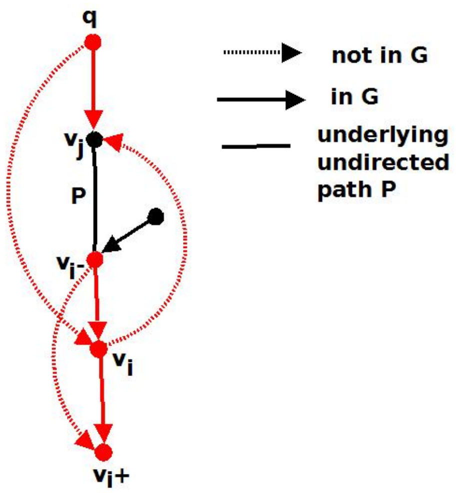


Figure 4.6: Case 2: one unique underlying undirected path P between q and v_i.

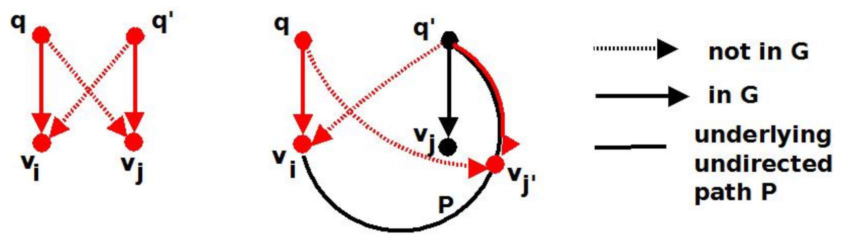


Figure 4.7: A larger source q' is not connected with vertex v_i.

G' is indeed acyclic and possesses a minimum number of weak components. On the other hand vertex v_i is connected with a larger number of sources as in G. Contradiction!

Hence, we can assume that there exists a dag realization $G = (V, A)$ with a minimum number of weak components such that the incoming neighborhood set of vertex v_i only contains sources. We consider a dag realization such that vertex v_i is connected with the maximum possible number of largest sources. Assume, there is a source $q' > q$ such that $(q, v_i) \in A$ and $(q', v_i) \notin A$. Then there exists a further vertex v_j with $(q', v_j) \in A$. We distinguish again between two cases. If there does not exist an underlying undirected path P between q and v_j or between q' and v_i, we define the new dag $G' := (V, A')$ with $A' := A \setminus \{(q, v_i), (q', v_j)\} \cup \{(q', v_i), (q, v_j)\}$ (see Figure 4.7) with a minimum number of weak components but with one larger source connected with v_i than in G. Contradiction!

Hence, we next assume that there is one underlying undirected path $P = q', v_{j'}, \ldots, v_i$ between q' and v_i. (A further path between q and v_j cannot exist, because in this case we would find an underlying cycle.) Note, that it is possible that we find $v_{j'} = v_j$. In this case we replace in the following steps $v_{j'}$ by v_j. We define the new dag realization $G' = (V, A')$ with $A' := A \setminus \{(q, v_i), (q', v_{j'})\} \cup \{(q', v_i), (q, v_{j'})\}$ with a minimum number of weak components, because we destroyed by our construction the underlying unique paths from q to $v_{j'}$ and from q' to v_i. Dag G' possesses one more large source connected with v_i than G. Contradiction! As a last case it remains, that there could exist an underlying undirected path $P = q, \ldots, v_j$ from q to v_j, see Figure 4.8.

Since q' is a larger source than q, there exists a further vertex $v_{j'}$ with $(q', v_{j'}) \in A$. Then we construct the dag realization $G' = (V, A')$ with $A' := (A \setminus \{(q, v_i), (q', v_{j'})\}) \cup \{(q', v_i), (q, v_{j'})\}$. Indeed, we destroyed in G the unique paths between q and $v_{j'}$ and between q' and v_i. Hence, G' is a dag with a minimum number of weak components but with one large source more connected to v_i than in G. Contradiction!
So, there exists a dag realization G such that vertex v_i has in its incoming neighborhood

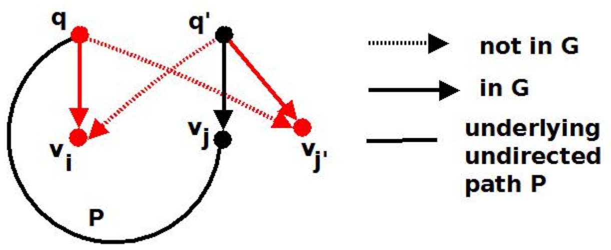


Figure 4.8: A larger source q' is not connected with vertex v_i and there exists an underlying path P between q to v_j.

set only largest sources. We delete the arcs from these sources to v_i in G and get a dag realization G' with dag sequence S'. $\square$

Note, that sequence S' may contain zero tuples. In this case, we delete these tuples and renumber the tuples from this new sequence $S' := \binom{a'_1}{b'_1}, \ldots, \binom{a'_{n'}}{b'_{n'}}$ from 1 to n'. Clearly, we have $\sum_{i=1}^{n'} a'_i \leq n - a_i - 1 \leq n' - 1$, because we deleted exactly the indegree of tuple $\binom{a_i}{b_i}$ in S and it is only possible to delete at most a_i occurring zero tuples in S'. Hence, Theorem 4.4 results in a recursive algorithm. At each step, one has to choose an arbitrary stream tuple $\binom{a_i}{b_i}$ with indegree of at least k and then to reduce a_i largest sources by one and to set the indegree a_i of this tuple to zero. On the other hand, all these possible tuples are also contained in the set V_{min} of Theorem 4.3. Hence, we get the following corollary.

Corollary 4.3 (arbitrary tuple choice in V_{min} for forest sequences)**.**
Let $S := \binom{a_1}{b_1}, \ldots, \binom{a_n}{b_n}$ be a canonically sorted forest sequence containing $k > 0$ source tuples. Furthermore, let S' be defined as in Theorem 4.3 where $\binom{a_{i_{min}}}{b_{i_{min}}}$ is an arbitrary tuple in V_{min}.
Then S is a dag sequence if and only if S' is a dag sequence.

In systematic experiments we found out that a large fraction of sequences can be solved by the lexmax strategy in polynomial time. We tell this story in the next Section 4.2. Moreover, we use the structural insights of our main theorem in restricting the number of possible topological sortings leading to a randomized algorithm which performs well in practice (Section 4.3).

4.2 How to Attack the NP-complete Dag Realization Problem in Practice

Why we became curious. Recall that the complexity status of the dag realization problem was open until a few weeks ago. To see whether our lexmax Conjecture 4.1 might be true, we generated a set of dag sequences denoted by *randomly generated sequences* by the following principle: Starting with a complete acyclic digraph, delete k of its arcs uniformly at random. We take the degree sequence from the resulting graph. Note that we only sample uniformly with respect to random dags but not uniformly degree sequences since degree sequences have different numbers of corresponding dag realizations.

In a first experiment we created with the described process one million dag sequences with 20 tuples each, and $m = \sum_{i=1}^{20} a_i = 114$. Likewise, we built up another million dag sequences with 25 tuples and $\sum_{i=1}^{25} a_i = 180$. The fact that the lexmax strategy realized all these test instances without a single failure was quite encouraging. The lexmax Conjecture 4.1 seemed to be true, only a correctness proof was missing. In Figure 4.4 we constructed a counter-example (the green path) and in Example 4.2 we found that no fixed strategy exists if we do not consider the sink tuples.

These observations give rise to several immediate questions: Why did we construct by our sampling method (for $n = 20$ and $n = 25$) only dag sequences which are lexmax sequences? How many dag sequences are not lexmax sequences?

Therefore, we started with systematic experiments. For small instances with $n \in \{7, 8, 9\}$ tuples we generated systematically the set of all dag sequences with all possible $\sum_{i=1}^{m} a_i =: m$, see for an example the case $n = 9$ in Figure 4.9. More precisely, we considered only *non-trivial sequences*, i.e. we eliminated all source-sink sequences and all sequences with only one stream tuple. We denote this set by *systematically generated sequences.* Note that the number of sequences grows so fast in n that a systematic construction of all sequences with a larger size is impossible.

We observed the following:

1. The fraction of lexmax sequences among the systematically generated sequences is quite high. For all m it is above 96.5%, see Figure 4.9 (blue squares).

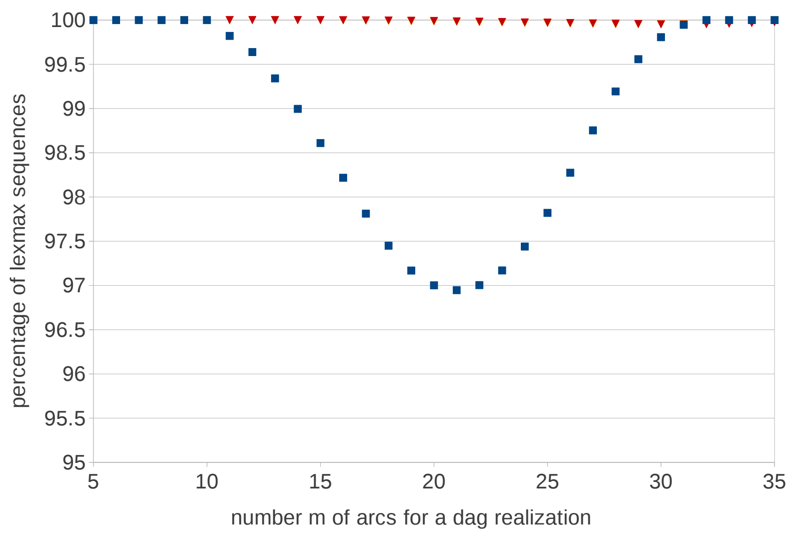


Figure 4.9: Percentage of (non-trivial) lexmax sequences for systematically generated (blue squares) and randomly generated sequences (red triangles) with 9 tuples and different $m \in \{5, \ldots, 35\}$.

2. The fraction of lexmax sequences strongly depends on m. It is largest for sparse and dense dags.

3. Lexmax sequences are overrepresented among one million randomly generated sequences (for each m), we observe more than 99% for all densities of dags, see Figure 4.9 (red triangles).

This leads to the following questions: Given a sequence for which we seek a dag realization. How should we proceed in practice? As we have seen, the huge majority of dag sequences are lexmax sequences. Is it possible to find out some characteristic properties for lexmax sequences or non-lexmax sequences, respectively?

Distance to opposed sequences. Let us exploit our characterization that opposed sequences are efficiently solvable. We propose the *distance to opposed* $d(S)$ for each dag sequence S. Consider for that the topological order of a dag realization G given by Algorithm 2, if in line 3 elements are chosen in decreasing lexicographical order. Thus, we obtain one unique dag realization G for S, if existing. Now, we renumber dag sequence S such that it follows the topological order induced by the execution by this

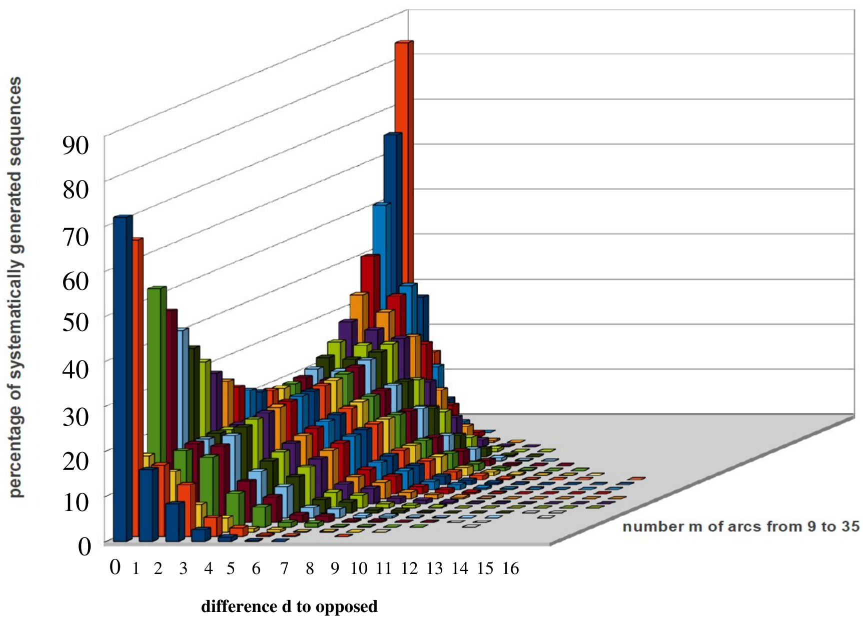


Figure 4.10: Percentage of systematic generated sequences S with their difference $d(S)$ to opposed with 9 tuples and $m \in \{9, \ldots, 35\}$.

algorithm, i. e. by the sequence of choices of elements from V'_{min}. Then the distance to opposed is defined as the number of pairwise incomparable stream tuples with respect to this order, more precisely,

$$d(S) := \left| \left\{ \left(\binom{a_i}{b_i}, \binom{a_j}{b_j} \right) \mid \binom{a_i}{b_i}, \binom{a_j}{b_j} \text{ incomparable stream tuples w. r .t. } \leq_{opp} \text{ and } i < j \right\} \right| .$$

We start this discussion by the following questions and experiments.

Question 1: Do randomly generated sequences possess a preference to a "small" distance to opposed in comparison with systematically generated sequences? In Figure 4.10, we show the distribution of systematically generated sequences (in %) with their distance to opposed, depending on $m := \sum_{i=1}^{n} a_i$. We compare this scenario with the same situation for the percentage of randomly generated sequences in Figure 4.11.

Observations: Systematically generated sequences have a slightly larger range of the "distance to opposed" than randomly generated sequences. Moreover, when we generate dag sequences systematically, we obtain a significantly larger fraction of instances with a larger distance to opposed than for randomly generated sequences, and this phenomenon can be observed for all densities.

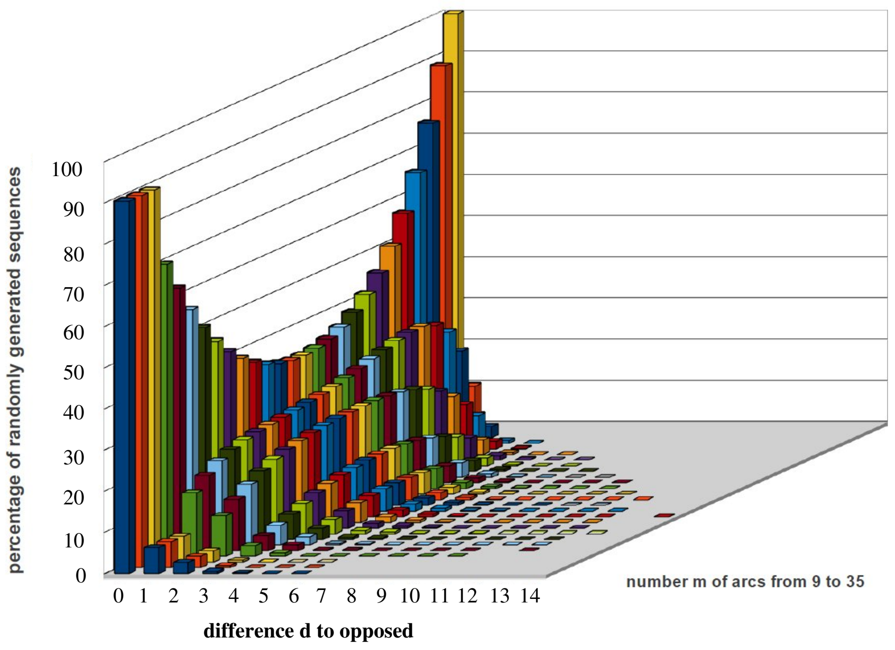


Figure 4.11: Percentage of randomized generated sequences S with their difference $d(S)$ to opposed with 9 tuples and $m \in \{9, \ldots, 35\}$.

Question 2: Do non-lexmax sequences possess a preference for large opposed distances? Since opposed sequences are easily solvable by Algorithm 1, we conjecture that sequences with a small distance to opposed might be easier solvable by the lexmax strategy than those with a large distance to opposed. If this conjecture were true, it would give us together with our findings for Question 1 one possible explanation for the observation that the randomly generated sequences have a larger fraction of efficiently solvable sequences by the lexmax strategy.

Observations: A separate analysis of non-lexmax sequences (that is, the subset of unsolved instances by the lexmax strategy), displayed in Figure 4.12, gives a clear picture: yes! For systematically generated sequences with $n = 9$, we observe in particular for instances with a middle density that the fraction of non-lexmax sequences becomes maximal for a relatively large distance to opposed.

Question 3: Can we solve real-world instances by the lexmax strategy? We consider real-world instances from different domains.

a): Ordered binary decision diagrams (OBDDs): In such networks the outdegree is two, that is constant. This immediately implies that the corresponding sequences

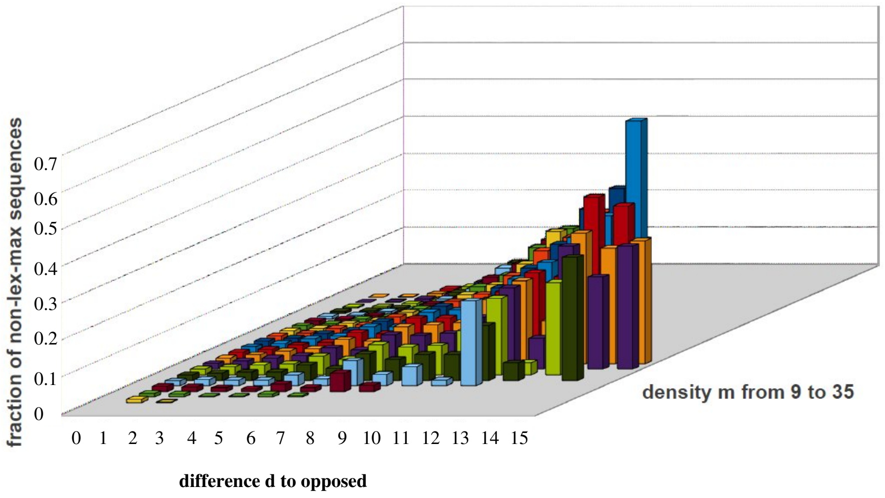


Figure 4.12: Fraction of systematic non lexmax sequences with 9 tuples and different densities $m \in \{9, \ldots, 35\}$ and varying difference to opposed $d(S)$.

are opposed sequences, and hence can provably be solved by the lexmax strategy.

b): Food Webs: Such networks are almost hierarchical and therefore have a strong tendency to be acyclic ("larger animals eat smaller animals"). In our experiments we analyzed food webs from the Pajek network library (Bat04).

c): Train timetable network: We use timetable data of German Railways from 2011 and form a time-expanded network. Its vertices correspond to departure and arrival events of trains, a departure vertex is connected by an arc with the arrival event corresponding to the very next train stop. Moreover, arrival and departure events at the same station are connected whenever a transfer between trains is possible or if the two events correspond to the very same train.

d): Flight timetable network: We use the European flight schedule of 2010 and form a time-expanded network as in c).

The characteristics of our real-world networks b) - d) are summarized in Table 4.1. Networks of type c) and d) are always opposed. The *dag density* ρ of a network is defined as $\rho = m/\binom{n}{2}$. To compare the distance to opposed for instances of different sizes, we normalize this value by the theoretical maximum $\binom{b}{2}$, where b denotes the number of stream tuples, and so obtain a *normalized distance to opposed.* Without any exception, all real-world instances have been realized by the lexmax strategy.

name and kind of network	n	m	b	ρ	normalized $d(S)$
burgess shale (b)	142	770	101	0.08	0.40
chengjiang shale (b)	85	559	54	0.16	0.50
florida bay dry (b)	128	2137	125	0.26	0.32
cyprus dry (b)	71	640	68	0.26	0.43
maspalomas (b)	24	82	21	0.30	0.30
rhode river (b)	20	53	17	0.28	0.42
train schedule 2011 (c)	19359	77201	18907	0.0004	0.00
flight schedule 2010 (d)	37800	1324556	32905	0.0019	0.00

Table 4.1: Characteristics of our real-world test instances.

4.3 Randomized Algorithms

The main idea for developing a randomized algorithm is the following. In each trial use a randomly chosen topological sorting (a random permutation of the tuples) for a given sequence and then apply the linear-time realization algorithm as described in Section 4.1 and justified by Theorem 4.1.

Clearly, it is not necessary to permute all tuples in a sequence. Instead we use a canonically sorted sequence and permute only the stream tuples. We denote this first naive version of a randomized algorithm by *stream tuple permutation algorithm* (Rand I). A random permutation of a sequence of length n can be chosen in $O(n)$ time, see for example (Dur64). Hence, one trial of the stream tuple permutation algorithm requires $O(m+n)$ time. This algorithm performs poorly since there are sequences with only a single realization among $(n-2)!$ many permutations of $n-2$ stream tuples. On the other hand, it is possible to restrict the number of possible topological sortings by the following theorem.

Theorem 4.5 (necessary criterion for the realizability of dag sequences).
Let S be a dag sequence. Denote the number of source tuples in S by q and the number of sink tuples by s. Then it follows $a_i \leq \min\{n-s, i-1\}$ and $b_i \leq \min\{n-q, n-i\}$ for all $i \in \mathbb{N}_n$ for each labeling of S corresponding to a topological order.

Proof. Let $S := \binom{a_1}{b_1}, \ldots, \binom{a_n}{b_n}$ be a labeling of S corresponding to a topological sorting of a dag realization G. Assume, there is a $j \in \mathbb{N}_n$ with $a_j > \min\{n-s, j-1\}$. (Case $b_j > \min\{n-q, n-j\}$ can be done analogously.) G is a subdigraph of a complete dag G^* with topological sorting $v_1, \ldots, v_n$. Clearly, we have $d^-_{G^*}(v_j) = j-1$. We distinguish

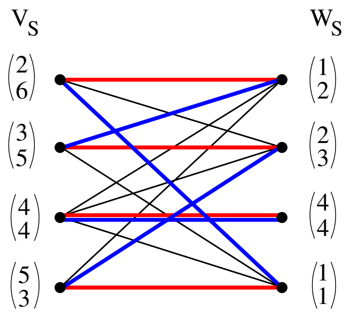


Figure 4.13: Bounding graph G_S for sequence $S := \binom{0}{3}, \binom{0}{1}, \binom{1}{2}, \binom{2}{3}, \binom{4}{4}, \binom{1}{1}, \binom{1}{0}, \binom{2}{0}, \binom{3}{0}$. One perfect matching (thick red edges) leads to the topological order $\binom{1}{2}, \binom{2}{3}, \binom{4}{4}, \binom{1}{1}$ which is realizable, whereas another perfect matching (thick blue edges) gives the topological order $\binom{1}{1}, \binom{1}{2}, \binom{4}{4}, \binom{2}{3}$ which is not realizable.

between two cases. If we have $\min\{n-s, j-1\} = n-s$, then it follows $a_j = d^-_G(v_j) > n-s$. Then the incoming neighborhood set $N^-_G(v_j)$ consists of more than $n-s$ vertices – in contradiction to the fact that $N^-_G(v_j)$ contains at most $n-s$ vertices. Let us now assume $\min\{n-s, j-1\} = j-1$. Then we get $d^-_{G^*}(v_j) = j-1 < a_j = d^-_G(v_j)$ – a contradiction to our assumption that G is a subdigraph of G^*. □

Hence, a stream tuple $\binom{a_i}{b_i}$ can only be at position j in a topological ordering if $a_j \leq \min\{n-s, i-1\}$ and $b_j \leq \min\{n-q, n-i\}$ is fulfilled. We define a bipartite *bounding graph* $B_S = (V_S \cup W_S, E_S)$ for a given canonically sorted sequence as follows. We define $|S| - q - s$ vertices $v_i \in V_S$ with $i \in \{q+1, \ldots, n-s\}$ where each vertex v_i corresponds to an "upper bound tuple" $\binom{\min\{n-s,i-1\}}{\min\{n-q,n-i\}}$ for a stream tuple in S. Furthermore, we define $|S| - q - s$ vertices w_i with $i \in \{q+1, \ldots, n-s\}$ each corresponding to a stream tuple $\binom{a_i}{b_i}$. The edge set E_S is built as follows. Two vertices v_i and w_j are adjacent if and only if we find for $\binom{a_j}{b_j}$ that $a_j \leq \min\{n-s, i-1\}$ and $b_j \leq \min\{n-q, n-i\}$. In Figure 4.13, we show an example of the bounding graph B_S where S is the dag sequence from the example in Figure 4.4.

A perfect matching in this bounding graph gives us a possible topological sorting with respect to Theorem 4.5. This means, we assign to each stream tuple $\binom{a_j}{b_j}$ in S the number i if and only if (v_i, w_j) is a matching edge in the chosen perfect matching. Clearly, there does not exist a dag realization of sequence S if B_S does not contain a perfect matching. Unfortunately, the computation of the number of perfect matchings in a bipartite graph is known to be $\sharp P$-hard (Val79). On the other hand, there exists a polynomial-time algorithm for the problem of uniform sampling a perfect matching within a bipartite graph by Jerrum, Sinclair and Vigoda (JSV04). They use a Markov chain based algorithm. The number of necessary steps in this algorithm is measured by the so-called *mixing time* τ_ϵ. The mixing time $\tau(\epsilon)$ is the minimum number of steps required by a random walk so that the distribution at this time has a variation distance $\delta(t) \leq \epsilon$. The *variation distance* $\delta(t)$ at time t with respect to the initial distribution P_0 measures how close the distribution of the Markov chain after t steps is to the stationary uniform distribution π. It is defined as $\delta(t) = \frac{1}{2}\sum_{V_G \in V_\Phi} |P_t(V_G) - \pi(V_G)|$. A random walk is said to be *rapidly mixing* if its mixing time can be bounded from above by a polynomial in the description length $n := |V|$ and $m := |E|$ of the bipartite graph $G = (V, E)$, on which a perfect matching has to be sampled. They proved a worst case mixing time of $O(n^8(n \log n + \log \frac{1}{\epsilon}) \log \frac{1}{\epsilon})$. Up to know, we do not know if we really need a uniform distribution, but we do not want to eliminate certain topological orderings. Our second version of a randomized algorithm – the *bounding permutation algorithm* (Rand II) – chooses in each trial a topological sorting by uniform sampling a perfect matching in B_S and then applies the realization algorithm for a given topological order (Theorem 4.1). For our experiments with very small instances, we sampled uniformly by enumerating all permutations of stream tuples.

Our third randomized algorithm – the *opposed permutation algorithm* (Rand III) – exploits the non-trivial result in Corollary 4.2 about opposed topological sortings. It uses for one trial, Algorithm 2 with a change in line 3. We replace line 3 by: "**Sample a $\boldsymbol{v_j \in V'_{min}}$ uniformly at random.**" If possible, we restrict the set of V'_{min} before line 3, i.e., we check for the largest $v_i \in V'_{min}$ whether the bounds of Lemma 4.4 are respected for later positions. Let k denote the number of recursive calls up to the current one. Expressed in terms of the original sequence, we have to choose the $(q+k)$–th tuple in the topological sorting in the current iteration. If $b_i = n-(q+k)$ for the lexicographical largest tuple $\binom{a_i}{b_i} \in V'_{min}$, then we set $V'_{min} := \{\binom{a_i}{b_i}\}$. The reason is that a larger position is not possible at all for this tuple, because the upper bound for b_i decreases strictly, as shown in Lemma 4.4. At first glance it is not clear whether the restriction to a subset of permutations within the randomized algorithm really increases the chance to draw a realizable topological sorting. This version of the algorithm only constructs dag realizations which possess an opposed topological sorting. Hence, we also exclude possible topological sortings which are not opposed topological sortings. However, empirically this idea pays off.

Our fourth randomized version combines the opposed permutation algorithm with several *reduction rules*.

1. *Exploit symmetric roles of in- and outdegrees.* If $|V'_{min}| = 1$, the reduction step in Algorithm 2 is safe (for any realizable sequence). Since the problem is symmetric

with respect to in- and outdegrees, we can exchange their roles. This suggests to check the size of V'_{min} from "both sides". If either of these sets has size one, the corresponding reduction step is safe and should be preferably applied.

2. *Degree dominance of some tuple.* Suppose that some b_i is so large that this number matches the number of available stream and sink tuples, then vertex v_i has to be connected with all current non-sources. Hence, sequence S can be reduced by deleting a source tuple $\binom{0}{b_i}$ or by updating a stream tuple $\binom{a_i}{b_i}$ to a new sink $\binom{a_i}{0}$, respectively, and by subtracting one from all $a_j > 0$ with $i \neq j$. The symmetric reduction rule can be stated for a dominating a_i-value.

3. *Dominating total degree of some stream tuple.* Suppose there is a stream tuple with $a_i + b_i = n - 1$. Then we can conclude that this tuple has to be connected with all other tuples. It is unclear which stream tuples come before and which after $\binom{a_i}{b_i}$ in some realization. However, we can be sure that it is connected with **all** sources and **all** sinks (in particular $a_i \leq q$ and $b_i \leq s$ must hold). In order to ensure that later recursive reduction steps do not introduce parallel arcs, we only apply a more conservative reduction. Namely, we connect the vertex v_i only with sources and sinks for which $a_i = 1$ or $b_i = 1$, respectively.

We additionally apply these rules whenever applicable and call the randomized algorithm *opposed permutation algorithm with reduction rules* (Rand IV).

Experiment 1: Which randomized algorithm possesses the best success probability for one trial? We define the *success probability* $p(m)$ as the probability that a given sequence $S := \binom{a_1}{b_1}, \ldots, \binom{a_n}{b_n}$ with $m := \sum_{i=1}^{n} a_i$ can be realized by a specified randomized algorithm in one single trial. In this experiment we test the four versions of our randomized algorithms with all non-trivial sequences (as defined in Section 4.2) of 9 tuples, see Figure 4.14. Moreover, we display the fraction of lexmax sequences to compare the deterministic lexmax strategy with our randomized strategy.

Observations: Randomized version 4 (opposed permutation algorithm with reduction rules) clearly outperforms all other strategies. We also observe that the success probability p depends on the density m of the dag realisations. Sparse and dense dags have the best success probability. The deterministic lexmax strategy has almost the same success probability as our best randomized version. Of course, we can repeat a randomized algorithm and thereby boost the success rate which is not possible for the deterministic variant. Nevertheless the good performance of the simple lexmax strategy

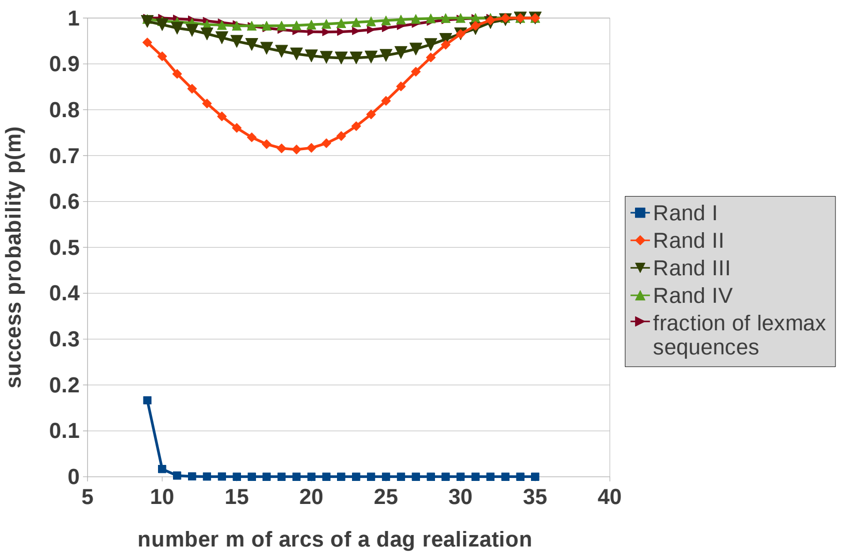


Figure 4.14: Success probability $p(m)$ for all non-trivial sequences with 9 tuples with four versions of randomized algorithms, and the fraction of lexmax sequences.

is quite remarkable, it clearly outperforms an arbitrary strategy to choose in line 3 of Algorithm 2 an element from V'_{min} (realized in randomized version Rand III).

Experiment 2: We consider the success probability for all randomized algorithms in the case of non-lexmax sequences which are not reducible by reduction rules 1-3. Noting that an impressively large fraction of sequences is efficiently solvable by the deterministic lexmax strategy combined with our reduction rules, we should ask: How well do our randomized algorithms perform for the remaining difficult cases, that is for *non-reducible non-lexmax sequences*? Actually, this is indeed the most interesting question, because the best approach for realizing a given sequence S would be: first to test, whether S is a reducible lexmax sequence. Only if this is not the case, one would take a randomized algorithm. Hence, we now determine the success probability $p(m)$ for all non-reducible non-lexmax sequences, see Figure 4.15.

Observations: As in the previous experiment, randomized version Rand IV has the overall best success probability p, but in sharp contrast we observe a completely different dependence on m. One possible explanation could be that for high densities our reduction rules have been applied more often. Note that the overall percentage of

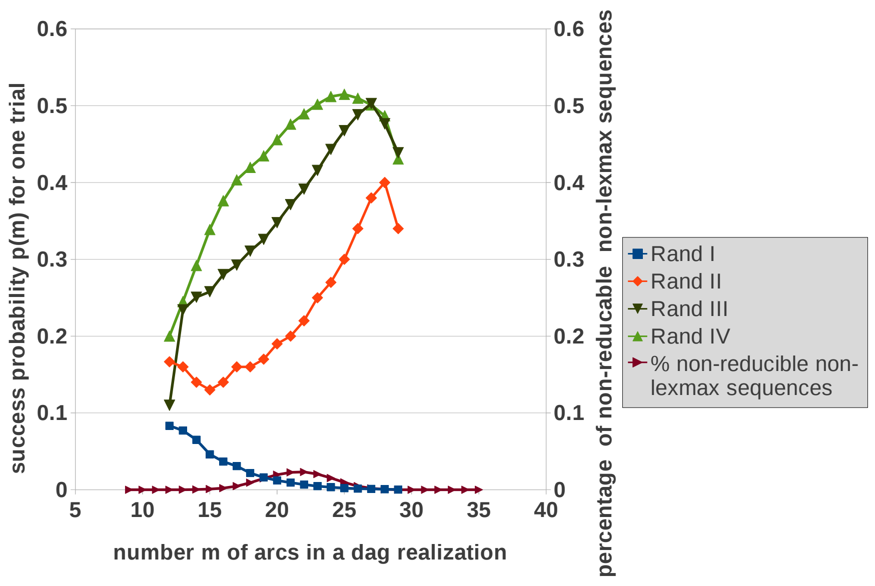


Figure 4.15: Success probability $p(m)$ for all non-reducible non-lexmax sequences of 9 tuples with four versions of randomized algorithms and percentage of non-reducible non-lexmax sequences in the set of all non-trivial sequences.

non-reducible non-lexmax sequences in the set of all non-trivial sequences with 9 tuples is so tiny (see the brown curve in Figure 4.15) — in particular for low densities — that we can realize after two or three trials almost all sequences.

For a given (real-world) sequence we propose the following recipe: Choose Algorithm 2 with lexmax strategy and apply the reduction rules 1-3. If this run is unsuccessful apply version 4 of our randomized algorithms, i. e. the opposed permutation algorithm with reduction rules. For most dag sequences in practice this will give us a pretty fair chance to find a realization.

The surprisingly broad success of the lexmax strategy suggests that there might be further subclasses of instances where it runs provably correct.

4.4 Characterization of Strongly Opposed Sequences

In this section, we want to focus on a special class of opposed sequences which we call *strongly opposed sequences.* Such sequences are characterized by the comparability of all pairs of tuples with respect to the opposed order. Hence, it is possible to sort the *whole* sequence S in increasing or decreasing order in the case of strongly opposed sequences.

We want to follow the classical results of a characterization of digraph sequences and here give a complete characterization of strongly opposed sequences. The conclusions from this result are quite interesting. We prove that each strongly opposed digraph sequence is also a dag sequence. This is not the case for opposed sequences in general.

An important point for understanding the ideas of the following Theorem 4.6 is the consideration of the canonical Ferrers matrix. A decreasing strongly opposed sequence fulfills the requirement $a_i \geq a_{i+1}$. Note, that such a sequence starts with the sinks and ends with the sources. In Theorem 4.5, we proved necessary conditions for dag sequences. Here, we apply this result for opposed sequences. By Theorem 4.1 an increasing, opposed sorting of an opposed dag sequence S is a topological sorting of at least one dag realization of S. Hence, we get the following result.

Corollary 4.4 (necessary criterion for the realizability of opposed sequences)**.**
Let S be an increasing opposed dag sequence. Denote the number of source tuples in S by q and the number of sink tuples by s. Then it follows $a_i \leq \min\{n-s, i-1\}$ and $b_i \leq \min\{n-q, n-i\}$ for all $i \in \mathbb{N}_n$.

This means in fact that the corresponding canonical Ferrers matrix F of a sequence fulfilling the conditions of Corollary 4.4 is an upper triangular matrix with zeros on its diagonal. In particular, the notions of the canonical Ferrers matrix and the diagonal-free Ferrers matrix are identical in this case. Moreover, F is the adjacency matrix of a dag realization, because it is a nilpotent matrix (for more details see Chapter 2). Clearly, if condition

$$\sum_{i=1}^{k} a_i \leq \sum_{i=1}^{k} a_i^*$$

is also fulfilled for all $k \in \mathbb{N}_n$ (where a_i^* are the column sums of F), then we can conclude that S is a digraph sequence by Theorem 3.5. In the next theorem we prove that this condition also characterizes strongly opposed dag sequences (and show later that it does not hold for general opposed sequences). Note, one important difference in our statement of the following theorem. We here consider *decreasing* opposed sequences in contrast to Corollary 4.4 which is stated with respect to *increasing* opposed sequences. The reason is that we decided to maintain the historical notion of Ferrers matrices. Hence, the indices are numbered in the opposite direction. Furthermore, we have to interchange the roles of a_i and b_i in Corollary 4.4.

Theorem 4.6 (characterization of strongly opposed dag sequences)**.**
Let S be a decreasing strongly opposed sequence. Sequence S is a dag sequence if and only if

(1.) $a_i \leq \min\{n-s, n-i\}$ *and* $b_i \leq \min\{n-q, i-1\}$ *is valid, and*

(2.) the corresponding sequence $S^{opp} := \binom{a_1^{opp}}{b_1^{opp}}, \dots, \binom{a_n^{opp}}{b_n^{opp}}$ *of the canonical Ferrers matrix F of S is a strongly opposed dag sequence such that the following property is valid for all $k \in \{1, \dots, n\}$:*

$$\sum_{i=1}^{k} a_i \leq \sum_{i=1}^{k} a_i^{opp}.$$

Proof. $\Rightarrow$: Let S be a strongly opposed dag sequence. Corollary 4.4 gives us property (1.), if we interchange the roles of a_i and b_i. Due to $b_i \leq \min\{n-q, i-1\}$ there exists a canonical $(n \times n)$-Ferrers matrix F for sequence S, which is a *lower* triangular matrix with zeros on its diagonal. For the column sums of the corresponding sequence S^{opp} we get $a_i^{opp} := |\{b_j | \ b_j \geq i, \ 1 \leq j \leq n\}|$. Clearly, $a_1^{opp} \geq a_2^{opp} \geq \dots \geq a_n^{opp}$ and $b_1^{opp} \leq \dots \leq b_n^{opp}$ with $b_i^{opp} := b_i$. Hence, S^{opp} is a decreasing strongly opposed sequence. F is the adjacency matrix of a dag realization of S^{opp} because F is nilpotent, see Section 2. Let us now consider the adjacency matrix A from a dag realization of S. In the ith row we find for the b_i entries $A_{ij_1}, \dots, A_{ij_{b_i}}$ which are "one", a "corresponding" sequence of "ones", namely $F_{il_1}, \dots, F_{il_{b_i}}$, such that the property $j_t \geq l_t$ is true for all $t \in \{1, \dots, b_i\}$. Hence, we get

$$\sum_{j=1}^{k} a_j = \sum_{i=1}^{n} \left(\sum_{j=1}^{k} A_{ij} \right) \leq \sum_{i=1}^{n} \left(\sum_{j=1}^{k} F_{ij} \right) = \sum_{j=1}^{k} a_j^{opp}$$

for all $k \in \mathbb{N}_n$ which is exactly condition (2.).

$\Leftarrow$: Let S be a strongly opposed sequence and S^{opp} the corresponding strongly opposed sequence (defined in condition (2.)) which is a dag sequence. We construct the canonical $(n \times n)$-Ferrers matrix F of S. F exists because we have $b_i \leq \min\{n - q, n-i\}$ by constraint (1.). Then matrix F is a lower triangular matrix with zeros on its diagonal. Hence, the canonical Ferrers matrix and the diagonal-free Ferrers matrix are here identical. For the ith column sums s_i of F it follows $s_i = |\{b_j | \ b_j \geq i, \ 1 \leq j \leq n\}| = a_i^{opp}$ for all $i \in \mathbb{N}_n$.

We show that S is a dag sequence if we have $\sum_{i=1}^{k} a_i \leq \sum_{i=1}^{k} a_i^{opp}$ for all $k \in \mathbb{N}_n$. We give a proof based on induction by the number of tuples n in sequence S.

- **Induction basis:** Let $n = 2$.

There is only one strongly opposed sequence with two tuples, namely $S = \binom{1}{0}, \binom{0}{1}$, which fulfills condition (1.). The associated canonical Ferrers matrix is $\left(\begin{smallmatrix} 0 & 0 \\ 1 & 0 \end{smallmatrix}\right)$ with $\sum_{i=1}^{k} a_i = \sum_{i=1}^{k} a_i^{opp}$ for $k \in \{1, 2\}$ and $S^{opp} = S$. Clearly, S is a dag sequence.

Now, we assume that S consists of exactly n tuples. By our assumption (2.), we have $a_1 \leq a_1^{opp}$.

- **Induction proof case 1:** We assume $a_1 = a_1^{opp}$.

Recall that we denote the number of sink tuples in S by s. By property (1.) we have $b_1 = 0$. We delete the first row and the first column in F and get the lower $(n-1) \times (n-1)$ triangular matrix F^* with zeros on its diagonal. This corresponds to the dag realization of the opposed sequence

$$S^* := \binom{a_2^{opp}}{b_2}, \dots, \binom{a_s^{opp}}{b_s}, \binom{a_{s+1}^{opp}}{b_{s+1}-1}, \dots, \binom{a_n^{opp}}{b_n - 1},$$

because all b_i are sorted in increasing ordering. For simplicity, we set $S^* := \binom{a_1^*}{b_1^*}, \dots, \binom{a_{n-1}^*}{b_{n-1}^*}$.

In a next step, we construct with respect to sequence S a sequence S' with

$$S' := \binom{a_2}{b_2}, \dots, \binom{a_s}{b_s}, \binom{a_{s+1}}{b_{s+1}-1}, \dots, \binom{a_n}{b_n - 1}$$

and set for simplicity $S' := \binom{a_1'}{b_1'}, \dots, \binom{a_{n-1}'}{b_{n-1}'}$. Note, that S' may contain zero tuples $\binom{0}{0}$ arising from source tuples $\binom{0}{1}$ in S. We distinguish between two cases.

case a: Assume, the number q' of sources in S' is smaller than the number q of sources in S.

In this case the entries of source tuples $\binom{0}{1}$ in S were decreased by one and we have at least one zero tuple $\binom{0}{0}$ in S'. On the other hand, the b_i are increasingly ordered by our assumption. It follows $b_i = 1$ for all stream tuples in S. Hence, sequence S' is a source-sink-sequence. Additionally, we have by our assumption $a_1 = a_1^{opp}$,

$$\sum_{i=1}^{k} a_i' = \sum_{i=2}^{k+1} a_i \leq \sum_{i=2}^{k+1} a_i^{opp} = \sum_{i=1}^{k} a_i^*$$

for all $k \in \{1, \dots, n-1\}$.

Obviously, matrix F^* is a lower triangular matrix with a zero diagonal which represents an adjacency matrix of a digraph realization. By application of Theorem 3.5 we identify S' as a digraph sequence. Each digraph realization of a source-sink-sequence is also a dag realization. Hence, there exists a dag realization G' with the labeled vertex set $V' := \{v_1', \dots, v_{n-1}'\}$. We construct a dag $G = (V, A)$ by inserting a new sink v_0' in $G' = (V', A')$ and connect the vertices $v_{s+1}', \dots, v_{n-1}' \in V'$ with v_0'. Then G is a dag realization of sequence S, if we define $v_i := v_{i-1}'$ with $v_i \in V$ and $i \in \{1, \dots, n\}$. Hence, S is a dag sequence.

case b: Consider the case $q' = q$.

Then we have no zero tuples in S'. We consider the ith column sum a^*_i of F^*. Under attention of $b_1 = 0$ we have for all $i \in \{1, \dots, n-1\}$,

$$\begin{aligned} a^*_i &= |\{b^*_j \mid b^*_j \geq i,\ 1 \leq j \leq n-1\}| \\ &= |\{b_j \mid b_j \geq i+1,\ 2 \leq j \leq n\}| \\ &= |\{b_j \mid b_j \geq i+1,\ 1 \leq j \leq n\}| \\ &= a^{opp}_{i+1}. \end{aligned}$$

Additionally, we get by our assumption $a_1 = a^{opp}_1$ for all $k \in \{1, \dots, n-1\}$,

$$\sum_{i=1}^{k} a'_i = \sum_{i=2}^{k+1} a_i \leq \sum_{i=2}^{k+1} a^{opp}_i = \sum_{i=1}^{k} a^*_i. \qquad (*)$$

In S^* all b^*_i are increasingly sorted and all a^*_i are decreasingly sorted, because S^{opp} is a decreasing opposed sequence and only one tuple $\binom{a^{opp}_1}{b_1}$ has been deleted. Hence, S^* is a strongly opposed dag sequence. On the other hand, F^* is a lower triangular matrix and the canonical Ferrers matrix of S'. We get $b'_i = b^*_i \leq \min\{n - q' - 1, i - 1\}$.

Obviously, we remove in S exactly one sink tuple, namely $\binom{a_1}{b_1}$. It is possible that σ new sink tuples $\binom{a_{s+1}}{b_{s+1}}, \dots, \binom{a_{s+\sigma}}{b_{s+\sigma}}$ are built in S'. Let $s' := s + \sigma - 1$ be the number of sink tuples in S'. We have $a^*_i \leq \min\{n - 1 - s', n - i - 1\}$, because F^* is a lower triangular matrix. Assume, there is an $i_0 \in \{1, \dots, n-1\}$ with $a'_{i_0} > n - 1 - s'$. Then we have $a_1 > n - 1 - s', \dots, a_{i_0} > n - 1 - s'$ using $a_1 \geq \dots \geq a_{i_0}$. We apply $(*)$ and get

$$(n - 1 - s') \cdot (i_0) < \sum_{i=1}^{i_0} a'_i \leq \sum_{i=1}^{i_0} a^*_i \leq (n - 1 - s') \cdot (i_0).$$

Contradiction! By our assumption $a_{i+1} \leq n - i - 1$ for all $i \in \{1, \dots, n-1\}$ we get $a'_i = a_{i+1} \leq \min\{n - 1 - s', n - i - 1\}$.

By the induction hypothesis, it follows that S' is a dag sequence. Hence, there is a dag realization $G' = (V', A')$ with the labelled vertex set $\{v'_1, \dots, v'_{n-1}\}$. We construct a dag $G = (V, A)$ by inserting a new sink v'_0 in $G' = (V', A')$ and connect the vertices $v'_{s+1}, \dots, v'_{n-1} \in V'$ with v'_0. Then G is a dag realization of sequence S, if we define $v_i := v'_{i-1}$ with $v_i \in V$ and $i \in \{1, \dots, n\}$. Hence, S is a dag sequence.

- **Induction proof case 2:** We assume $a_1 < a^{opp}_1$.

construction of an induction hypothesis: The goal of the following steps is the construction of a lower triangular matrix F^* such that the first column sum is a_1. This matrix shall be built from F by shifting entries "one" of the first column to a larger column under attention of maintaining a triangular matrix with zeros on its diagonal. We prove that it is possible to construct such a matrix F^*. We show all steps by an example.
Let us start with our example. Given is the decreasing strongly opposed sequence

$$S := \binom{3}{0}, \binom{3}{0}, \binom{3}{0}, \binom{3}{2}, \binom{3}{2}, \binom{3}{4}, \binom{2}{4}, \binom{0}{4}, \binom{0}{4}$$

and its canonical Ferrers-matrix F

$$\begin{array}{cc} \begin{bmatrix} 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 1&1&0&0&0&0&0&0&0\\ 1&1&0&0&0&0&0&0&0\\ 1&1&1&1&0&0&0&0&0\\ 1&1&1&1&0&0&0&0&0\\ 1&1&1&1&0&0&0&0&0\\ 1&1&1&1&0&0&0&0&0 \end{bmatrix} & \begin{array}{l} b_1=0\\ b_2=0\\ b_3=0\\ b_4=2\\ b_5=2\\ b_6=4\\ b_7=4\\ b_8=4\\ b_9=4 \end{array} \\ \begin{array}{ccccccccc} 6&6&4&4&0&0&0&0&0 \end{array} & a_i^{opp} \end{array}$$

with corresponding sequence

$$S^* := \binom{6}{0}, \binom{6}{0}, \binom{4}{0}, \binom{4}{2}, \binom{0}{2}, \binom{0}{4}, \binom{0}{4}, \binom{0}{4}, \binom{0}{4}.$$

We define with respect to matrix F the sets D and Z with

$$D := \{k| \ F_{k1} = 1 \wedge \exists l < k \text{ with } F_{kl} = 0\}$$

and

$$Z := \{(kl)| \ k \in D \wedge l \text{ is the smallest column index with } F_{kl} = 0\}.$$

Hence, we identify in F all entries of the first column which can be shifted to a column with a larger index l, where the definition of set D ensures that we maintain the triangular matrix form. In our example, we get the sets $D = \{4, 5, 6, 7, 8, 9\}$ and $Z = \{(43), (53), (65), (75), (85), (95)\}$. We build a new matrix F^* by interchanging entries $F_{k1} = 1$ with $k \in D$ and the corresponding $F_{kl} = 0$ with $(kl) \in Z$ for exactly $z := \min\{|Z|, a_1^{opp} - a_1\}$ such entries. For this, we sort the elements in Z with respect to their column indices l in increasing order and number them by $(k_1 l_1), \dots, (k_{|Z|} l_{|Z|})$. If two column indices l are identical, we prefer the pair of (kl) and $(k'l)$ with the larger row index $k > k'$. We distinguish between two different cases. First, let $|Z| \geq a_1^{opp} - a_1$. Then F^* is a lower triangular matrix with the first column sum a_1, because we are able to shift all supplemental $z = a_1^{opp} - a_1$ entries "one" and get a lower triangular matrix. In this case we get our requested construction. This is the case in our example where we get for the elements in Z the following sorting: $(k_1 l_1) = (53), (k_2 l_2) = (43), (k_3 l_3) = (95), (k_4 l_4) = (85), (k_5 l_5) = (75), (k_6 l_6) = (65)$. Moreover, we have $z = 3$ and get the matrix F^*

$$\begin{array}{cl}
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{0} & 1 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{0} & 1 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{0} & 1 & 1 & 1 & \mathbf{1} & 0 & 0 & 0 & 0
\end{bmatrix} &
\begin{array}{l}
b_1 = 0 \\ b_2 = 0 \\ b_3 = 0 \\ b_4 = 2 \\ b_5 = 2 \\ b_6 = 4 \\ b_7 = 4 \\ b_8 = 4 \\ b_9 = 4
\end{array} \\
\begin{array}{ccccccccc} 3 & 6 & 6 & 4 & 1 & 0 & 0 & 0 & 0 \end{array} & a_i^*
\end{array}$$

It remains to prove that the second case, $|Z| < a_1^{opp} - a_1$, is not possible. Assume the converse. Then, it follows $a_1^* > a_1$. Let us denote the number of rows corresponding to a non-sink and starting with an entry $F_{i1}^* = 0$ by k. Then we have

$$a_1^* := n - k - s \text{ where } 0 \leq k < n - s.$$

In the following we give an upper bound u for $\sum_{i=1}^n a_i$ and a lower bound l for $\sum_{i=1}^n a_i^*$ and show that we have $u < l$ in contradiction to the fact $\sum_{i=1}^n a_i = \sum_{i=1}^n a_i^*$. We start with some observations.

The upper bound u. With our assumption $a_1 \geq a_2 \geq \cdots \geq a_n$ and $a_1^* - a_1 \geq 1$ we get $a_1 \leq n - k - s - 1$. Hence, we have by assumption $a_i \leq \min\{n - s, n - i\}$ of condition (1.),

$$\begin{aligned}
\sum_{i=1}^{n} a_i &\leq \left(\sum_{i=1}^{k+s+1} (n-(k+s+1))\right) + (n-(k+s+2)) + (n-(k+s+3)) + \cdots + 1 + 0 \\
&= (n-k-s-1)(k+s+1) + (n-k-s-2) + (n-k-s-3) + \cdots + 1 \\
&=: u.
\end{aligned}$$

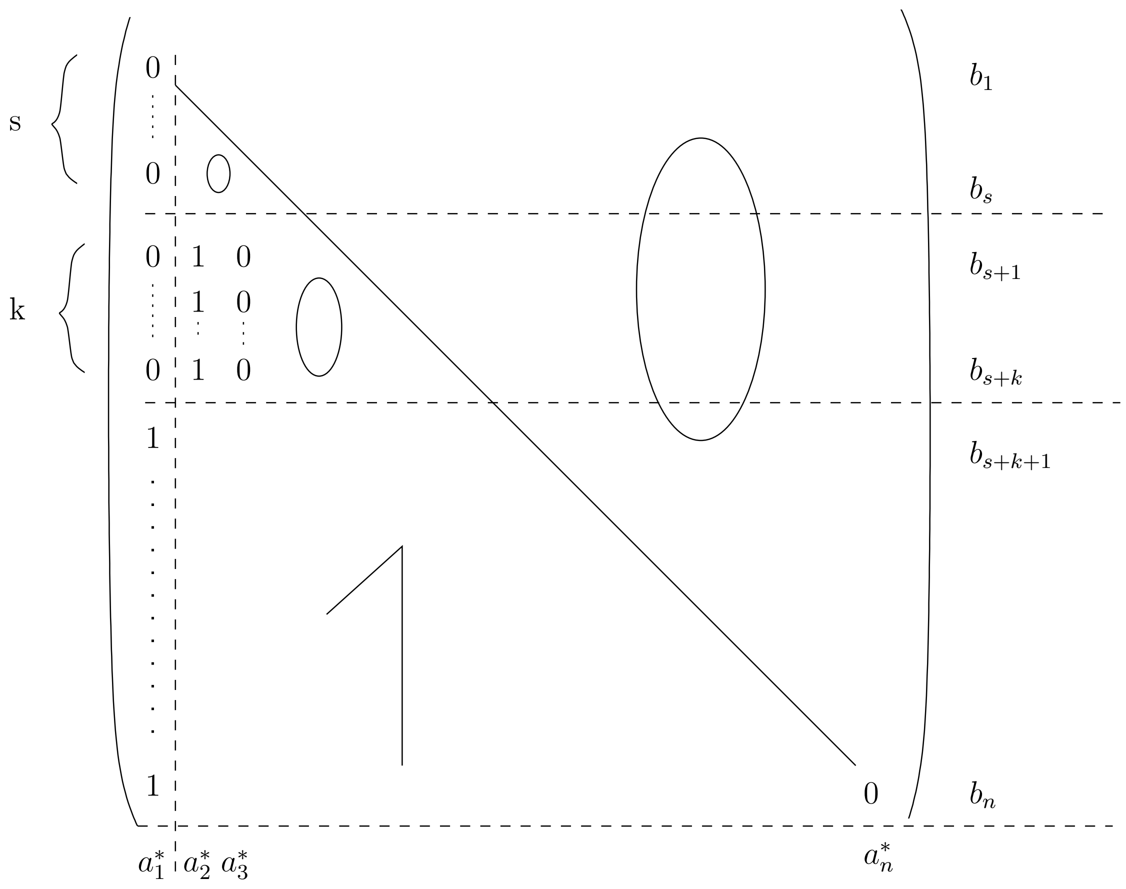
Three insights about F^*. The lower bound l.

1. For a row i starting with $F^*_{i1} = 1$ it follows $F^*_{il} = 1$ for all $l < i$. Otherwise, we had $i \in D$ and F^* is not constructed in the right way.

2. For a row i starting with $F^*_{i1} = 0$ it follows $F^*_{il} = 1$ for all $l < i$ if we have $F^*_{(i-1)1} = 1$. The reason is the condition $b_{i-1} \leq b_i$ and case 1.

3. For a row i starting with $F^*_{i1} = 0$ it follows $F^*_{il} = 1$ for an $l < i$ if we have $F^*_{(i-1)1} = 0$ and $F^*_{(i-1)l} = 1$. The reason is the condition $b_{i-1} \leq b_i$.

We construct a matrix F^* containing a minimum number l of entries "one" with respect to all three cases.



Clearly, each other construction would contain more entries “one”.

Note, that the rows corresponding to non-sinks are **not** full rows of a complete lower triangular matrix where we have $a_i = \min\{n-i, n-s\}$ entries for each $i \in \{1, \dots, n\}$. Instead in each of the rows $s+1, s+2, \dots, s+k$ we have $s-1, s, s+1, \dots, s+k-2$ less than expected entries “one”. Hence, we get

$$
\begin{aligned}
l &:= \sum_{i=1}^{n} a_i^* \\
&= \left(\sum_{i=1}^{n} \min\{n-i, n-s\} \right) - ((s-1) + s + (s+1) + \cdots + (s+k-2)) \\
&= (n-s) \cdot s + (n-(s+1)) + (n-(s+2)) + \cdots + 1 + 0 - k \cdot s - \sum_{i=1}^{k-2} i + 1
\end{aligned}
$$

We prove that $\sum_{i=1}^n a_i^* - \sum_{i=1}^n a_i > 0$.

$$\begin{aligned}
\sum_{i=1}^n a_i^* - \sum_{i=1}^n a_i &\geq l-u \\
&= \left[(n-s)\cdot s + (n-s-1) + (n-s-2) + \cdots + 1 - k\cdot s - \sum_{i=1}^{k-2} i + 1\right] \\
&- [(n-k-s-1)(k+s+1) + (n-k-s-2) + (n-k-s-3) + \cdots + 1] \\
&= \left[(n-s-k)\cdot s + (n-s-1) + (n-s-2) + \cdots + 1 - \sum_{i=1}^{k-2} i + 1\right] \\
&- [(n-s-k-1)\cdot s + (n-s-k-1)\cdot(k+1) + (n-k-s-2) + (n-k-s-3) + \cdots + 1] \\
&= \left[(n-s-1) + (n-s-2) + \cdots + (n-s-k-1) + (n-s-k-2) + \cdots + 1 - \sum_{i=1}^{k-2} i + 1\right] \\
&- [-s + k\cdot(n-k-s-1) + (n-k-s-1) + (n-k-s-2) + \ldots 1] \\
&= \left[(n-s-1) + (n-s-2) + \cdots + (n-s-k) - \sum_{i=1}^{k-2} i + 1\right] \\
&- [-s + k\cdot(n-k-s-1)] \\
&= (-1+1+k) + (-2+1+k) + (-3+1+k) + \cdots + (-k+1+k) - \sum_{i=1}^{k-2} i + 1 + s \\
&= \sum_{i=1}^{k} i - \sum_{i=1}^{k-2} i + 1 + s \\
&= 2k + s \\
&> 0.
\end{aligned}$$

Induction step, construction of the induction hypothesis. In this part we construct from sequence S a sequence S' and its corresponding threshold sequence S^{**} only consisting of $n-1$ tuples so that the induction hypothesis can be applied. We do this by deleting the first column and the first row of matrix F^*. First, we need some preparations.

For simplicity, we denote the set of indices of the *changed* columns in F in comparison to F^* (without column 1) by L. Let λ_i denote the number of changed entries in column $i \in L$. Obviously, we get $\sum_{i\in L} \lambda_i = a_1^{opp} - a_1$. Furthermore, we denote by R the set of indices of *unchanged* rows. For our example, we find $L = \{3, 5\}$ and $\lambda_3 = 2$ and $\lambda_5 = 1$. Moreover, we get $R = \{1, 2, 3, 6, 7, 8\}$.
For the corresponding sequence $S^* := \binom{a_1^*}{b_1^*}, \ldots, \binom{a_n^*}{b_n^*}$ of F^* we have

$$\begin{pmatrix} a_i^* \\ b_i^* \end{pmatrix} := \begin{cases} \binom{a_i}{b_i} & \text{if } i = 1 \\ \binom{a_i^{opp}+\lambda_i}{b_i^{opp}} & \text{if } i \in L \\ \binom{a_i^{opp}}{b_i^{opp}} & \text{otherwise.} \end{cases}$$

We delete the first column and the first line of F^* and get the $(n-1)\times(n-1)$-matrix F^{**} with corresponding sequence $S^{**} := \binom{a_1^{**}}{b_1^{**}}, \ldots, \binom{a_{n-1}^{**}}{b_{n-1}^{**}}$ where

$$\begin{pmatrix} a_i^{**} \\ b_i^{**} \end{pmatrix} := \begin{cases} \binom{a_{i+1}^*}{b_{i+1}-1} & \text{if } i+1 \in R \text{ and } b_{i+1} \neq 0 \\ \binom{a_{i+1}^*}{b_{i+1}^*} & \text{otherwise.} \end{cases}$$

Then we construct sequence $S' := \binom{a_1'}{b_1'}, \ldots, \binom{a_{n-1}'}{b_{n-1}'}$ by modifying sequence $S := \binom{a_1}{b_1}, \ldots, \binom{a_n}{b_n}$ in the following way.

$$\begin{pmatrix} a_i' \\ b_i' \end{pmatrix} := \begin{cases} \binom{a_{i+1}}{b_{i+1}-1} & \text{if } i+1 \in R \text{ and } b_{i+1} \neq 0 \\ \binom{a_{i+1}}{b_{i+1}} & \text{otherwise.} \end{cases}$$

That is, we set $b_i' := b_i^{**}$ for all $i \in \{1, \ldots, n-1\}$.

case 1: S' contains at least one zero tuple $\binom{0}{0}$.

In this case we only shifted entries from the first column to the second column to construct F^* from F. The reason is that zero tuples can only be built from sources $\binom{0}{1}$ in S. Clearly, we have $b_i = 1$ for all stream tuples in S, because all b_i are sorted in increasing order. If we observe $b_i' = 0$ for an index i not corresponding to a sink in S, then it is $b_{i+1} = 1$. Hence, the row $(i+1)$ is unchanged during the construction of F^*. It follows $(i+1) \in R$. From our construction rule for F^* –"shift (if necessary) entries into the columns with smallest indices" we conclude that entries of column 1 were only shifted to column 2. Otherwise, we had $b_i' = 1$. On the other hand, sequence S' may only contain sources and sinks, because if we have for one source $b_i' = 0$, then we get $b_j = 0$ for all stream tuples $j \in R$ under consideration of $j < i+1$ and our construction rule. Consider for example the canonical Ferrers matrix F (below) for sequence $S := \binom{4}{0}, \binom{3}{0}, \binom{2}{0}, \binom{1}{1}, \binom{0}{1}, \binom{0}{1}, \binom{0}{1}, \binom{0}{2}, \binom{0}{4}$ with

$$\begin{array}{l}\begin{bmatrix}0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0\\\mathbf{1}&0&0&0&0&0&0&0&0\\\mathbf{1}&0&0&0&0&0&0&0&0\\1&1&0&0&0&0&0&0&0\\1&1&1&1&0&0&0&0&0\end{bmatrix}\begin{array}{l}b_1=0\\b_2=0\\b_s=0\\b_4=1\\b_5=1\\b_6=1\\b_7=1\\b_8=2\\b_9=4\end{array}\\ \begin{array}{ccccccccc}6&2&1&1&0&0&0&0&0\end{array}\quad a_i^{opp}.\end{array}$$

The entries F_{61} and F_{71} are moved into column 2 by the construction of F^* with

$$\begin{array}{l}\begin{bmatrix}0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0\\0&\mathbf{1}&0&0&0&0&0&0&0\\0&\mathbf{1}&0&0&0&0&0&0&0\\1&1&0&0&0&0&0&0&0\\1&1&1&1&0&0&0&0&0\end{bmatrix}\begin{array}{l}b_1^*=0\\b_2^*=0\\b_3^*=0\\b_4^*=1\\b_5^*=1\\b_6^*=1\\b_7^*=1\\b_8^*=2\\b_9^*=4\end{array}\\ \begin{array}{ccccccccc}4&4&1&1&0&0&0&0&0\end{array}\quad a_i^*.\end{array}$$

Deleting row 1 and column 1 results in a matrix F^{**} and sequence $S^{**} := \binom{4}{0}, \binom{1}{0}, \binom{1}{0}, \binom{0}{0}, \binom{0}{1}, \binom{0}{1}, \binom{0}{1}, \binom{0}{3}$ and $S' := \binom{3}{0}, \binom{2}{0}, \binom{1}{0}, \binom{0}{0}, \binom{0}{1}, \binom{0}{1}, \binom{0}{1}, \binom{0}{3}$.
Note, that only the first two columns were modified. Hence, it follows $a_1^{opp} + a_2^{opp} = a_1^* + a_2^*$ and $a_i^{opp} = a_i^*$ for all $i \geq 3$. By our assumption it follows for all $k \geq 2$

$$\sum_{i=1}^{k} a_i^* = \sum_{i=1}^{k} a_i^{opp} \geq \sum_{i=1}^{k} a_i.$$

With $a_1^* = a_1$ we have for all $k \geq 1$

$$\sum_{i=1}^{k} a_i^{**} = \sum_{i=2}^{k+1} a_i^* \geq \sum_{i=2}^{k+1} a_i = \sum_{i=1}^{k} a_i'.$$

We delete all zero tuples of S' and the corresponding zero tuples with the same indices from S^{**}. (They exist in S^* due to the column sum relation.) Furthermore, the a'_i are decreasingly sorted, because we observe $a'_i = a_{i+1}$ for all $i \geq 1$. All relations for our (above proved) column sums of S^{**} and S' remain, because we only deleted tuples $a'_i = 0$ with largest indices. This is clearly the case because our sequence is labelled in a decreasing opposed order. It follows by Theorem 3.3, that S' is a digraph sequence with at most one loop per vertex. S' is a source-sink-sequence and therefore already a dag sequence. Then it exists a dag realization $G' = (V', A')$ with the labelled vertex set $\{v'_1, \dots, v'_{n-1}\}$. We build a dag $G = (V, A)$ by inserting a new sink v'_0 in $G' = (V', A')$ and connect all vertices $v'_i \in V'$ with $i + 1 \in R$ and $b_{i+1} \neq 0$ with v'_0. Then G is a dag realization of sequence S, if we define $v_i := v'_{i-1}$ for $v_i \in V$. Hence, S is a dag sequence.

case 2: S' does not contain zero tuples $\binom{0}{0}$.

Note, that the number q' of sources in S' equals q by our construction rule. We show that sequence S' is a decreasing opposed sequence and F^{**} the diagonal-free $(n-1) \times (n-1)$-Ferrers matrix with corresponding sequence S^{**} such that conditions (1.) and (2.) are fulfilled and therefore the induction hypothesis.

claim 1: Sequence S' is a decreasing opposed sequence.

claim 2: Matrix F^{**} is the diagonal-free Ferrers matrix for sequence S'.

claim 3: We have column sum relation (2.) for S' and S^{**}, if the number of sinks in S' is $s' = s - 1$. Especially, we have $a'_i \leq \min\{(n-1) - s', (n-1) - i\}$ for all $i \in \{1, \dots, n-1\}$.

claim 4: The column sum relation (2.) is fulfilled, if we have for the number s' of sinks in S', $s' > s - 1$. Furthermore, the condition $a'_i \leq \min\{(n-1) - s', (n-1) - i\}$ is fulfilled for all $i \in \{1, \dots, n-1\}$.

claim 5: $b'_i \leq \min\{n - 1 - q', i - 1\}$ for all $i \in \{1, \dots, n-1\}$.

Proof of claim 1. Assume, there exists $(i+1) \in R$ with $b_{i+1} > 0$ such that we have $b'_{i-1} > b'_i$. Then we have $i \notin R$, otherwise it follows from our assumption $b_{i-1} \leq b_i$, that $b'_{i-1} \leq b'_i$. We can conclude $b_i = b_{i+1}$ and $F_{(i+1)1} = 1$. Moreover, $i \in D$ and it exists a corresponding pair $(ij) \in Z$. On the other hand, we have $i + 1 \in D$ and $((i+1)j) \in Z$. But then we would have chosen $((i+1)j) \in Z$ before $(ij) \in Z$ by our construction rule. It follows $(i+1) \notin R$ in contradiction to our assumption.

Proof of claim 2. F^{**} is a lower triangular matrix with zeros on its diagonal, because we find this property for F^*. Assume, there is a $k > l$ with $k, l \in \{1, \ldots, n-1\}$ such that we find $F^{**}_{kl} = 1$ and $F^{**}_{k(l-1)} = 0$. Then we have in F, $F_{(k+1)(l+1)} = 0$ and $F_{(k+1)l} = 0$, because F is a diagonal-free Ferrers matrix. With $F^*_{(k+1)(l+1)} = F^{**}_{kl} = 1$ it follows $k+1 \in D$ and $F_{(k+1)(l+1)} \in Z$ is the corresponding entry in contradiction to our reconstruction rule of Z, where we have to choose the smallest possible index l for all columns. For the corresponding sequence S^{**} of F^{**} we find $b^{**}_i = b'_i$ for all $i \in \{1, \ldots, n-1\}$. Hence, F^{**} is the diagonal-free Ferrers matrix of S'.

Proof of claim 3. Let $s' := s - 1$. Then we have

$$a'_i = a_{i+1} \leq \min\{n-s, n-i-1\} = \min\{(n-1)-s', (n-1)-i\}$$

for all $i \in \{1, \ldots, n-1\}$. We denote the maximum index within the set L by t. For all $i < t-1$ we find by our construction rule of F^{**}, $a^{**}_i = \min\{n-1-s', n-i-1\} \geq a'_i$, because all columns with a smaller index as t (but not the first column) are completely filled by our reconstruction of F^* from F. For the i-th column sum of F^{**} and $k < t-1$ we have

$$\sum_{i=1}^{k} a^{**}_i \geq \sum_{i=1}^{k} a'_i.$$

For $n \geq k \geq t-1$ we get by our construction and assumption

$$\sum_{i=1}^{k} a^*_i = \sum_{i=1}^{k} a^{opp}_i \geq \sum_{i=1}^{k} a_i.$$

Due to $a^*_1 = a_1$, it follows

$$\sum_{i=1}^{k-1} a^{**}_i = \sum_{i=2}^{k} a^*_i \geq \sum_{i=2}^{k} a_i = \sum_{i=1}^{k-1} a'_i.$$

Proof of claim 4. Let $s' > s - 1$. Then we can conclude that all shifted entries of the first column of F are put into the second column because the second column is not complete. Furthermore, all stream tuples with $b_i = 1$ are now sink tuples in S'. But then we have $a^{opp}_1 + a^{opp}_2 = a^*_1 + a^*_2$ and $a^{opp}_i = a^*_i$ for all $i \geq 3$. Due to $a^*_1 = a_1$ and

$$\sum_{i=1}^{k} a_i \leq \sum_{i=1}^{k} a^{opp}_i = \sum_{i=1}^{k} a^*_i$$

for all $k \geq 2$ we find then

$$\sum_{i=1}^{k} a'_i \leq \sum_{i=1}^{k} a^{**}_i$$

for all $k \geq 1$. Assume, there exists an $i_0 \in \{1, \ldots, n-1\}$ with $a'_{i_0} > n-1-s'$. Then it follows $a'_1, \ldots, a'_{i_0} > n-1-s'$, because all a'_i are decreasingly sorted. Hence, we get

$$(n-1-s') \cdot (i_0) < \sum_{i=1}^{i_0} a'_i \leq \sum_{i=1}^{i_0} a^{**}_i \leq (n-1-s') \cdot (i_0),$$

because F^{**} is the diagonal-free Ferrers matrix of S by claim (2). Contradiction! On the other hand we have by our assumption $a'_i = a_{i+1} \leq n-i-1$ for all $i \in \{1, \ldots, n-1\}$. Therefore, we get $a'_i \leq \min\{n-1-s', n-i-1\}$.

Proof of claim 5. By our assumption, we get for all column indices $(i+1) \in R$

$$b'_i = b_{i+1} - 1 \leq \min\{n-q, i\} - 1 = \min\{n-q', i\} - 1 \leq \min\{n-1-q', i-1\}.$$

For a row $(i+1) \notin R$ there exists a corresponding matrix entry $F_{(i+1)j}$ with $((i+1)j) \in Z$. But then we have $b_{i+1} < \min\{n-q, i\}$ in S. Otherwise, we would get $F_{(i+1)j} = 1$ and therefore $((i+1)j) \notin Z$. Then it follows

$$b'_i = b_{i+1} \leq \min\{n-q, i\} - 1 \leq \min\{n-q-1, i-1\} = \min\{(n-1)-q', i-1\}.$$

Induction step. We apply our induction hypothesis to sequence S'. It follows that S' is a dag sequence of length $n-1$ by our claims (1)–(5). Let $G' = (V', A')$ be a dag realization with the labeled vertex set $\{v'_1, \ldots, v'_{n-1}\}$. We build from $G' = (V', A')$ a dag $G = (V, A)$, by adding a new sink v'_0 and connecting the vertices $v'_i \in V'$ with $(i+1) \in R$ to it. Then G is a dag realization of sequence S, if we define $v_i := v'_{i-1}$ for all $v_i \in V$. Hence, S is a dag sequence. $\square$

It is quite interesting to consider the canonical Ferrers matrix of a strongly opposed dag sequence more closely. The theorems for canonical and diagonal-free matrices are the same in this case because the diagonal consists of zeros and the triangular upper matrix only contains zeros. This leads to an easier characterization for strongly opposed dag sequences.

Corollary 4.5. *Let $S := \binom{a_1}{b_1}, \ldots, \binom{a_n}{b_n}$ be a decreasing strongly opposed sequence. Sequence S is a dag sequence if and only if S is a digraph sequence and we have $a_i \leq \min\{n-s, n-i\}$ and $b_i \leq \min\{n-q, i-1\}$ for all $i \in \mathbb{N}_n$.*

Proof. $\Rightarrow$: If S is a dag sequence then it also is a digraph sequence. Corollary 4.4 proves the second claim.

$\Leftarrow$: Let S be a digraph sequence with $a_i \leq \min\{n-s, n-i\}$ and $b_i \leq \min\{n-q, i-1\}$ for all $i \in \mathbb{N}_n$. Sequence $a_1, \ldots, a_n$ is decreasingly sorted because S is a strongly opposed sequence. We build the canonical Ferrers matrix F. It is a lower triangular matrix because we have $b_i \leq \min\{n-q, i-1\}$. Theorem 3.3 proves the column sum relation $\sum_{i=1}^{k} a_i \leq \sum_{i=1}^{k} a_i^* = \sum_{i=1}^{k} a_i^{opp}$ for all $k \in \{1, \ldots, n\}$, because S is a digraph sequence. Hence, all sufficient conditions for Theorem 4.6 are fulfilled and S is a dag sequence. $\square$

Corollary 4.5 cannot be extended to opposed sequences in general. The opposed digraph sequence $S := \binom{4}{0}, \binom{4}{0}, \binom{2}{0}, \binom{1}{2}, \binom{1}{4}, \binom{1}{5}, \binom{0}{2}$ is **not** a dag sequence which can easily proven with our realization algorithm for opposed sequences Algorithm 1. Moreover, we try to characterize an opposed sequence similar to Theorem 4.6. We use a possible topological order for a dag realization which is exactly the opposed order of an opposed sequence by Corollary 4.5. We start the attempt to construct a special Ferrers matrix for this sequence analogously to Theorem 4.6. As a result, we get the corresponding sequence $S^* := \binom{4}{0}, \binom{4}{0}, \binom{2}{0}, \binom{2}{2}, \binom{1}{4}, \binom{0}{5}, \binom{0}{2}$ for the lower triangular matrix F with

$$\begin{array}{c}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{array}{l} b_1 = 0 \\ b_2 = 0 \\ b_3 = 0 \\ b_4 = 2 \\ b_5 = 4 \\ b_6 = 5 \\ b_7 = 2. \end{array} \\ \begin{array}{cccccccl} 4 & 4 & 2 & 2 & 1 & 0 & 0 & a_i^* \end{array}\end{array}$$

The column sum relation $\sum_{i=1}^{k} a_i \leq \sum_{i=1}^{k} a_i^*$ is fulfilled for all $k \in \{1, \ldots, 7\}$. But, we know that sequence S is not a dag sequence. Hence, this kind of characterization is no longer possible for characterizations of general opposed sequences.

5

Uniform Sampling of Digraph Realizations

“Gott würfelt nicht.” (Albert Einstein)

5.1 Popular Variants for Sampling Graph and Digraph Realizations

We consider the problem of sampling uniformly at random from the set of all graph realizations of a graph sequence or from the set of all digraph realizations of a digraph sequence. A general method to sample random elements from some set of objects is via *rapidly mixing Markov chains* (Sin92, Sin93). Every Markov chain can be viewed as a *random walk* on a digraph $\Phi = (V_\Phi, A_\Phi)$, the so-called *state digraph.* In our context, its vertex set V_Φ (the states) correspond one-to-one to the set of all graph (or digraph) realizations of a given di(graph) sequence. In Section 5.2 we will define the arc set of our state digraph. In Figure 5.1 we give an example for a state digraph for digraph sequence $S = \binom{0}{1}, \binom{0}{1}, \binom{0}{1}, \binom{1}{0}, \binom{1}{0}, \binom{1}{0}$.

Let us briefly review the basic notions of random walks and their relation to Markov chains in our context. See (Lov96, JS96, Sin93) for more details. A random walk (Markov chain) on a digraph $\Phi = (V_\Phi, A_\Phi)$ **with loops** is a sequence of vertices $V_{G_0}, V_{G_1}, \dots, V_{G_t}, \dots$ where $(V_{G_i}, V_{G_{i+1}}) \in A_\Phi$. Vertex V_{G_0} represents the initial state. Denote by $d^+_\Phi(V)$ the outdegree of vertex $V \in V_\Phi$. At the tth step we move to an arbitrary neighbor of V_{G_t} with probability $\frac{|N^+_\Phi(V_{G_t})|}{d^+_\Phi(V_{G_t})}$ or stay at V_{G_t} with probability

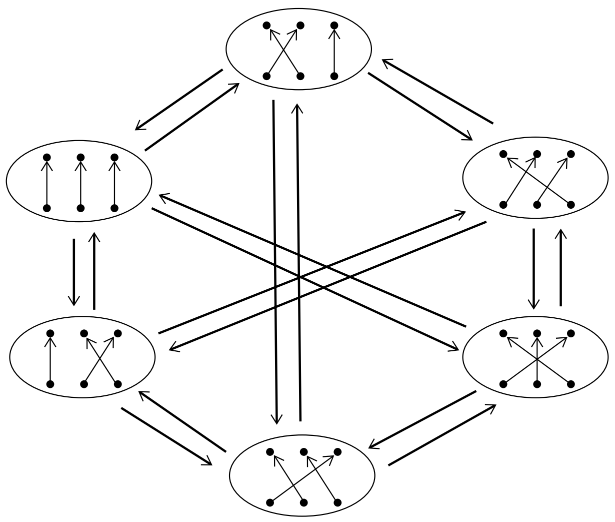


Figure 5.1: Example of a state digraph.

$\frac{|L_\Phi(V_{G_t})|}{d^+_\Phi(V_{G_t})}$, where $|N^+_\Phi(V_{G_t})|$ denotes the number of neighbors of V_{G_t} and $L_\Phi(V_{G_t})$ the set of loops (see Chapter 2). Furthermore, we define the distribution of V_Φ at time $t \in \mathbb{Z}^+$ as the function $P_t \in [0,1]^{|V_\Phi|}$ with

$$P_t(i) := Prob(\text{vertex } V_{G_t} \text{corresponds to digraph realization } G_i).$$

A well-known result (Lov96) is that P_t tends to the uniform stationary distribution for $t \to \infty$, if the digraph Φ is (1) non-bipartite (that means aperiodic), (2) strongly connected (i. e., irreducible), (3) symmetric, and (4) regular. A digraph Φ is *d_Φ-regular* if all vertices have the same in- and outdegrees d_Φ. Here, we will view Markov chains as random walks on symmetric d_Φ-regular digraphs $\Phi = (V_\Phi, A_\Phi)$ with loops whose vertices correspond to the state space V_Φ of all digraph realizations for a given digraph sequence. Note, that simply adding a certain number of loops to each vertex V_{G_i} constructs a regular state digraph. The transition probability on each arc $(V_G, V_{G'}) \in A_\Phi$ will be the constant $1/d_\Phi$.
The *variation distance* $\delta(t)$ at time t with respect to the initial distribution P_0 measures how close the distribution of the Markov chain after t steps is to the stationary distribution π. It is defined as $\delta(t) = \frac{1}{2}\sum_{V_G \in V_\Phi} |P_t(V_G) - \pi(V_G)|$. The mixing time $\tau(\epsilon)$ is the minimum number of steps required by a random walk so that the distribution at this time has a variation distance $\delta(t) \leq \epsilon$. A random walk is said to be *rapidly mixing* if its mixing time can be bounded from above by a polynomial in the description length $n := |V_G|$ and $m := \sum_{i=1}^n a_i$ of a single digraph realization of digraph sequence $S = \binom{a_1}{b_1}, \ldots, \binom{a_n}{b_n}$.

One popular variant of the Markov chain approach to sample among such (di)graph realizations is the so-called *switching-algorithm*. It starts with a given (di)graph realization, and then performs a sequence of 2-swaps. Hence, in a state digraph Φ two vertices are adjacent if and only if they are different in exactly four arcs, for an example consider Figure 5.1. This means, in the undirected case, a *2-swap* replaces two non-adjacent edges $\{a, b\}, \{c, d\}$ of a graph realization either by $\{a, c\}, \{b, d\}$ or by $\{a, d\}, \{b, c\}$, provided that both new edges have not been contained in the graph before the swap operation. Likewise, in the directed case, given two arcs $(a, b), (c, d)$ with all vertices a, b, c, d being distinct, a *2-swap* replaces these two arcs by $(a, d), (c, b)$ which are currently not included in the digraph realization (the latter is crucial to avoid parallel arcs). The switching algorithm is usually stopped heuristically after a certain number of iterations, and then outputs the resulting digraph realization as a "random element". The question whether this Markov chain is rapidly mixing is an open problem.

For undirected graphs, one can prove that this switching algorithm converges to a random stage. The directed case, however, turns out to be much more difficult. The following example demonstrates that the switching algorithm does not even converge to a uniformly random stage.

Example 5.1. *Consider the following class of digraphs $D = (V, A)$ with $3n$ vertices $V = \{v_1, v_2, \dots, v_{3n}\}$, see Figure 5.2. Roughly speaking, this class consists of induced directed 3-cycles C_i formed by triples $V_i = \{v_{3i+1}, v_{3i+2}, v_{3i+3}\}$ of vertices, and arcs $A_i = \{(v_{3i+1}, v_{3i+2}), (v_{3i+2}, v_{3i+3}), (v_{3i+3}, v_{3i+1})\}$ for $i \in \{0, \dots, n-1\}$. All vertices of cycle C_i are connected to all other vertices of cycles with an index larger than i. More formally, let $A' := \{(v, w)|\ v \in V_i, w \in V_j, i < j\}$. We set $A := A' \cup (\cup_{i=0}^{n-1} A_i)$.*

It is easy to check that no 2-swap can be applied to this digraph. However, we can independently reorient each of the n induced directed 3-cycles, leading to $2^{n/3}$ many (isomorphic) digraph realizations of the same digraph sequence. Thus, if we use only 2-swaps to define the possible transitions between digraph realizations, we will be stuck in a single digraph realization although exponentially many exist.

We give further non-trivial classes of digraphs where the switching algorithm will fail. Consider the following Figures 5.3 and 5.4. In both cases instances of these classes cannot be changed to another digraph realization which is only different in the orientation of the directed 3-cycle by a sequence of 2-swaps.

It is interesting to note that 2-swap operations suffice to sample digraphs *with* loops as has been proven by Ryser (Rys57). Kannan et al. (KTV99) showed how to sample bipartite graphs via Markov chains. They proved polynomial mixing time

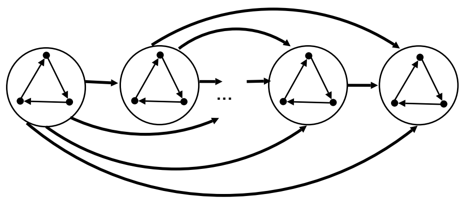


Figure 5.2: Example of digraphs where no 2-swap operation can be applied.

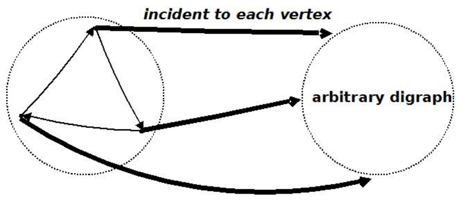


Figure 5.3: Digraph class where the switching algorithm fails.

for regular and near-regular graphs. Cooper et al. (CDG07) extended this work to non-bipartite, d-regular graphs and proved a polynomial mixing time for the switching algorithm. More precisely, they upper bounded the mixing time in these cases by $d^{15}n^8(dn\log(dn)+\log(\varepsilon^{-1}))$, for graphs with $|V|=n$. Erdős et al. (EMS10) extended the results from Kannan et al. to bipartite graphs which are semi-regular, i.e., bipartite graphs in which at least one independent vertex set is regular. They proved a rapidly mixing Markov chain for these graph types. In a break-through paper, Jerrum, Sinclair, and Vigoda (JSV04) presented a polynomial-time almost uniform sampling algorithm for perfect matchings in bipartite graphs. Their approach can be used to sample arbitrary bipartite graphs and arbitrary digraphs with a specified digraph sequence in $O(n^{14}\log^4 n)$ via a reduction by Tutte (Tut54). In the context of sampling binary contingency tables, Bezáková et al. (BBV07) managed to improve the running time for these sampling problems to $O(n^{11}\log^5 n)$, which is still far from practical.

A further popular version to sample a graph realization uniformly at random is the so-called *configuration model* (sometimes also called "pairing model"). Mostly, it is used for regular graphs with n vertices and a constant degree d, see for example Bender and Canfield (BC78) and Bollabás (Bol80). The idea of this approach is as follows.

1. Replace each of the n vertices by d vertices.

2. Choose a perfect matching of these $n \cdot d$ vertices uniformly.

3. Repeat this process until a *simple* graph (without parallel edges) is obtained.

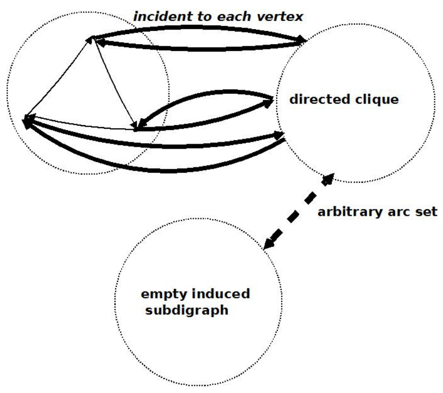


Figure 5.4: Further digraph class where the switching algorithm fails.

This algorithm looks very simple. The problem is an exponentially large running time for large degrees d.

Theorem 5.1 (Bender, Canfield (BC78)). *The probability that G is a simple graph tends to* $\exp \frac{1-d^2}{4}$ *for $n \to \infty$.*

There exist several modifications for the configuration model, a most popular one is to modify step 2., i. e. , choose only edges which do not correspond to loops or parallel edges. The problems maintain. The distribution only converges to a uniform distribution in the case of regular graph sequences. Provable polynomial running times are only known for small degrees d. We cite several results with respect to the configuration model. McKay and Wormald (MW90, MW91) use a configuration model and generate random graphs with degrees bounded by $o(n^{1/2})$ with uniform distribution in $O(m^2 d_{max})$ time, where d_{max} denotes the maximum degree, and m the number of edges. Steger and Wormald (SW99) introduced a modification of the configuration model that leads to a fast algorithm and samples asymptotically uniform for degrees up to $o(n^{1/28})$. Kim and Vu (KV03) improved the analysis of Steger and Wormald's algorithm, proving that the output is asymptotically uniform for degrees up to $O(n^{1/3-\epsilon})$, for any $\epsilon > 0$. Bayati et al. (BKS10) recently presented a nearly-linear time algorithm for counting and randomly generating almost uniformly graph realizations with a given graph sequence where the maximum degree is restricted to $d_{max} = O(m^{1/4-\tau})$, and τ is any positive constant.

Carefully looking at our example in Figure 5.2, we observe that the state digraph becomes strongly connected if we add a second type of operation to transform one digraph realization into another: Simply reorient the arcs of an induced directed 3-cycle. We call this operation *3-cycle reorientation.* We give a graph-theoretical proof that 2-

swaps and 3-cycle reorientations suffice not only in this example, but also in general for arbitrary digraph sequences. These observations allow us to define a Markov chain, very similar to the undirected case. The difference is that two digraph realizations are mutually connected by arcs if and only if their symmetric difference is either an alternating directed 4-cycle or a 6-cycle with exactly three distinct vertices. This state digraph becomes regular by adding additional loops. The transition probabilities are of order $O(1/m^2)$, and the diameter can be bounded by $O(m)$, where m denotes the number of arcs in the digraph sequence.

In the context of $(0,1)$-matrices with given marginals (i. e., digraph sequences in our terminology), Rao et al. (RJB96) similarly observed that switching operations on so-called "compact alternating hexagons" are necessary. A compact alternating hexagon is a (3×3)-submatrix, which can be interpreted as the adjacency matrix of a directed 3-cycle subdigraph. They define a random walk on a series of digraphs, starting with a non-regular state digraph which is iteratively updated towards regularity, i. e. their Markov chain converges asymptotically to the uniform distribution. However, it is unclear how fast this process converges and whether this is more efficient than starting directly with a single regular state digraph. Since Rao et al. work directly on matrices, their transition probabilities are of order $O(1/n^6)$, i. e., by several orders smaller than in our version.

Very recently, Erdős et al. (EMT09) proposed a similar Markov chain approach using 2-swaps and 3-swaps. The latter type of operation exchanges a simple directed 3-path or 3-cycle $(v_1, v_2), (v_2, v_3)$, (v_3, v_4) (the first and last vertex may be identical) by $(v_1, v_3), (v_3, v_2), (v_2, v_4)$, but is a much larger set of operations than ours.

Although in digraphs 2-swaps alone do not suffice to sample uniformly in general, the corresponding approach is still frequently used in network analysis. One reason for the popularity of this approach — in addition to its simplicity — might be that it empirically worked in many cases quite well (MKI$^+$04). In this thesis, we study under which conditions this approach can be applied and provably leads to correct uniform sampling. We call such digraph sequences *arc-swap sequences*, and give a graph-theoretical characterization which can be checked in polynomial time. More specifically, we can recognize arc-swap sequences in $O(m^2)$ time using matching techniques. Using a parallel Havel-Hakimi algorithm by LaMar (LaM09), originally developed to realize *Euler sequences* (for each tuple we have $a_i = b_i$) with an even number of arcs, the recognition problem can even be solved in linear time. This algorithm also allows us to determine the number of induced directed 3-cycles which appear in *every* digraph realization.

However, the simpler approach comes with a price: our bound on the diameter of the state digraph becomes mn and so is by one order of n worse in comparison with the alternative of using 2-swaps and 3-cycle reorientations. Since half of the diameter is a trivial lower bound on the mixing time and the diameter also appears as a factor in known upper bounds, we conjecture that the classical switching algorithm requires a mixing time τ_ε with an order of n more steps as the variant with 3-cycle reorientation. However, it remains as an open problem whether these Markov chains are rapidly mixing.

In those cases where 2-swaps do not suffice to sample uniformly, the state digraph decomposes into 2^k strongly connected components, where k is the number of induced directed 3-cycles which appear in every realization. We can also efficiently determine the number of strongly connected components of the state digraph (of course, without explicitly constructing this exponentially sized graph). However, all these components are isomorphic. This can be exploited as follows: For a non-arc-swap sequence, we first determine all those induced directed 3-cycles which appear in every digraph realization. By reducing the in- and outdegrees for all vertices of these 3-cycles by one, we then obtain a new sequence, now guaranteed to be an arc-swap sequence. On the latter we can either use the switching algorithm or our variant with additional 3-cycle reorientations on a smaller state digraph with a reduced diameter bound $n(m-3k)$ or $m-3k$, respectively, yielding an important practical advantage.

Our results (see also our publication (BM10)) give a theoretical foundation to compute certain network characteristics on unlabeled digraphs in a single component using 2-swaps only. For example, this includes the analysis of the motif content (MSOI$^+$02). Likewise we can still compute the average diameter among all digraph realizations if we work in a single component. However, for other network characteristics, for example betweenness centrality on arcs (KLP$^+$05), this leads in general to incorrect estimations.

5.2 Random Walk on the State Digraph of Digraph Realizations

We denote the state digraph (with loops) for our random walk by $\Phi = (V_\Phi, A_\Phi)$. Its underlying vertex set V_ϕ is the set of all digraph realizations of a given digraph sequence S. For a digraph realization G, we denote by V_G the corresponding vertex in the vertex set V_Φ. The arc multi-set A_Φ is defined as follows.

a) We connect two vertices $V_G, V_{G'} \in V_\Phi, G \neq G'$ with arcs $(V_G, V_{G'})$ and $(V_{G'}, V_G)$ if and only if one of the two following constraints is fulfilled

1. $|G\Delta G'| = 4$
2. $|G\Delta G'| = 6$ and $G\Delta G'$ contains exactly three different vertices.

b) We set a directed loop (V_G, V_G)

1. for each pair of non-adjacent arcs $(v_{i_1}, v_{i_2}), (v_{i_3}, v_{i_4}) \in A(G), i_j \in \{1, \ldots, n\}$ if and only if $(v_{i_1}, v_{i_4}) \in A(G) \vee (v_{i_3}, v_{i_2}) \in A(G)$ in the digraph realization G,
2. for each directed 2-path $(v_{i_1}, v_{i_2}), (v_{i_2}, v_{i_3}) \in A(G)$ if and only if one of the following constraints is true for the digraph realization G,
   i) $(v_{i_2}, v_{i_1}) \in A(G) \vee (v_{i_3}, v_{i_2}) \in A(G) \vee (v_{i_1}, v_{i_3}) \in A(G)$,
   ii) $(v_{i_3}, v_{i_1}) \notin A(G)$,
   iii) $i_3 < i_1 \vee i_3 < i_2$.
3. if G contains no directed 2-path.

Note, that multiple loops are allowed.

Lemma 5.1. *The state digraph* $\Phi := (V_\Phi, A_\Phi)$ *is non-bipartite, symmetric, and regular.*

Proof. In our setting we always connect two distinct vertices in both directions. Hence, Φ is symmetric. Furthermore, if some digraph realization G contains no directed 2-path, then each G is a digraph realization of a sequence S, only consisting of sinks and sources. With our setting Φ contains for each $V_G \in V_\Phi$ a directed loop and is therefore non-bipartite, by item $b)$3. in our construction. Let us now assume that a digraph realization G contains a directed 2-path. Either there exists a third arc which completes these two arcs to a directed 3-cycle or not. In all cases we can guarantee one directed loop at V_G : In the case of a directed 3-cycle C we distinguish two cases. Either $b)2.i)$ is fulfilled or in C there exists a 2-path with conditions as in $b)2.iii)$. If we have a 2-path which is not a subpath of a directed 3-cycle then we get condition $b)2.ii)$. Hence, Φ is not bipartite. For the proof of regularity, we consider at each vertex V_G the number of pairs of non-adjacent arcs in a digraph realization G. This is the number of all possible arc pairs minus the number of adjacent arcs $\binom{|A(G)|}{2} - \left(\sum_{i=1}^{n} \binom{a_i}{2} + \sum_{i=1}^{n} \binom{b_i}{2} + \sum_{i=1}^{n} a_i b_i\right)$ where $\sum_{i=1}^{n} \binom{a_i}{2}$ is the number of all incoming arc pairs at each vertex, $\sum_{i=1}^{n} \binom{b_i}{2}$ is the number of all outgoing arc pairs at each vertex and $\sum_{i=1}^{n} a_i b_i$ is the number of directed 2-paths in a digraph realization G. Hence, the number of non-adjacent arcs is a constant value for each digraph realization G. For each of these arc pairs we either set a directed loop or an incoming and an outgoing arc at each vertex $V_G \in V_\Phi$. For each 2-path in G we set a loop if it is not part of a directed 3-cycle $C = (v_{i_1}, v_{i_2}, v_{i_3}, v_{i_1})$ which is an induced subdigraph $C = G\langle\{v_{i_1}, v_{i_2}, v_{i_3}\}\rangle$. Otherwise, there exists a digraph realization G' with

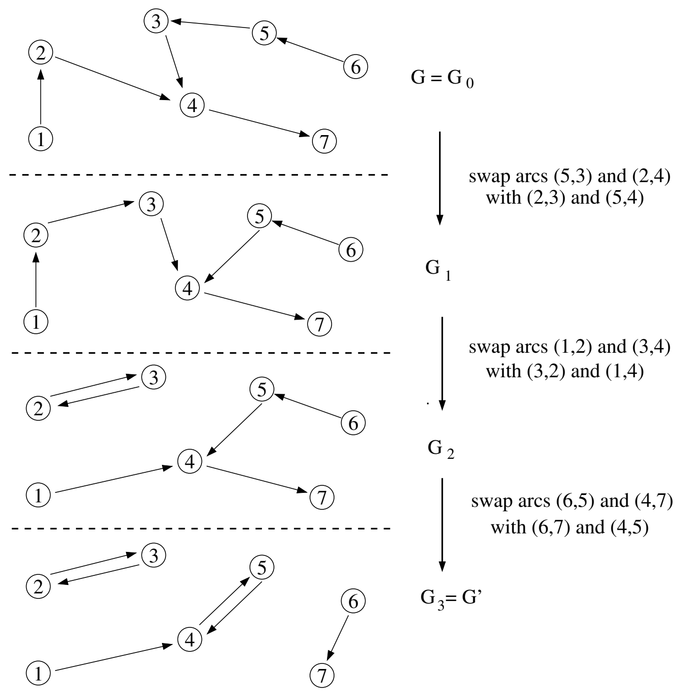

G = G0
swap arcs (5,3) and (2,4) with (2,3) and (5,4)
G1
swap arcs (1,2) and (3,4) with (3,2) and (1,4)
G2
swap arcs (6,5) and (4,7) with (6,7) and (4,5)
G3= G'

Figure 5.5: Transforming G from Figure 2.1 into G' by a sequence of swap operations.

$|G\Delta G'| = 6$ and $G\Delta G'$ contains exactly 3 different vertices. Hence, we set for the 2-path in C with $i_j < i_{j'}$ and $i_{j'} < i_{j''}$ with $j, j', j'' \in \{1, 2, 3\}$ the directed arcs $(V_G, V_{G'})$ and $(V_{G'}, V_G)$ and for both other 2-paths in C a directed loop. Generally, we set for all 2-paths in a digraph realization an incoming and an outgoing arc at each V_G. The number of 2-paths in each digraph realization is the constant value $\sum_{i=1}^{n} a_i b_i$. Hence, the vertex indegree (outdegree) at each vertex is $d_\Phi := d_\Phi^+ = d_\Phi^- = \binom{|A(G)|}{2} - 2\sum_{i=1}^{n}\binom{a_i}{2}$. □

In the next section we have to prove that our constructed state digraphs are strongly connected. This is sufficient to prove the reachability of each digraph realization independent of the starting digraph realization. Figure 5.5 shows an example how the digraph realization G from Figure 2.1 can be transformed to the digraph realization G' by a sequence of swap operations.

5.2.1 Symmetric differences of two different digraph realizations

Proposition 5.1. *Let S be a digraph sequence and G and G' be two different digraph realizations. If $G\Delta G'$ is exactly one weak component and $|G\Delta G'| \neq 6$, then there exists in $G\Delta G'$ an alternating oriented path of type P or Q, where $P = (v_1, a_1, v_2, a_2, v_3, a_3, v_4)$ with $a_1 = (v_1, v_2), a_3 = (v_3, v_4) \in A(G)$ and $a_2 = (v_3, v_2) \in A(G')$ and $Q = (w_1, b_1, w_2, b_2,$*

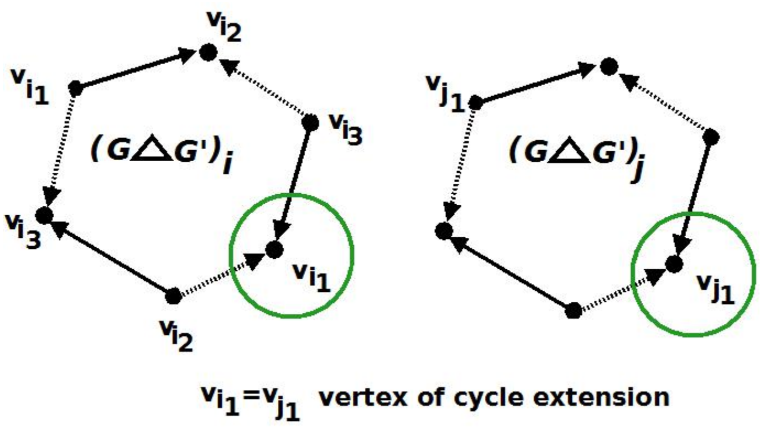


Figure 5.6: Alternating oriented cycles $(G\Delta G')_i$ and $(G\Delta G')_j$ of symmetric difference $G\Delta G'$.

$w_3, b_3, w_4)$ with $b_1 = (w_1, w_2), b_3 = (w_3, w_4) \in A(G')$ and $b_2 = (w_3, w_2) \in A(G)$.

Proof. Note that in $G\Delta G'$ an alternating oriented cycle of length two is not possible. Otherwise, there exists an arc $(u, v) \in A(G) \cap A(G')$ in contradiction to our assumption that $(u, v) \in G\Delta G'$. The symmetric difference $G\Delta G'$ may decompose into a number of alternating oriented cycles $(G\Delta G')_i$. We consider a decomposition into the minimum number of such cycles. (For more details see Chapter 2.) If one of these alternating oriented cycles $(G\Delta G')_i$ contains an alternating oriented 3–path P or Q as claimed, we are done. Otherwise, each vertex is repeated at each third step in $(G\Delta G')_i$ see Figure 5.6.

Hence, we get the alternating oriented cycles

$$(G\Delta G')_i := (v_{i_1}, a_{i_1}, v_{i_2}, a_{i_2}, v_{i_3}, a_{i_3}, v_{i_1}, b_{i_1}, v_{i_2}, b_{i_2}, v_{i_3}, b_{i_3} v_{i_1})$$

where $a_{i_1}, a_{i_3}, b_{i_2} \in A(G)$ and $a_{i_2}, b_{i_1}, b_{i_3} \in A(G')$. The cycle cannot be longer, as the graph induced by $G\Delta G'\langle\{v_1, v_2, v_3\}\rangle$ is already complete. Since $|(G\Delta G')_i| = 6$, there must be $(G\Delta G')_j$ with $i \neq j$. $(G\Delta G')_j$ shares at least one vertex with $(G\Delta G')_i$, because $G\Delta G'$ is weakly connected. There must be exactly one $v_{i_1} = v_{j_1}$, since otherwise these two cycles were not arc-disjoint. The union of these two cycles is an alternating oriented cycle, because vertex v_{j_1} is repeated at the third step in cycle $(G\Delta G')_j$. Hence, vertex v_{j_1} possesses there two incoming and two outgoing arcs which makes it possible to combine the cycle $(G\Delta G')_i$ at this vertex with cycle $(G\Delta G')_j$. Contradiction to the assumption of the minimality of the decomposition! □

Note that the above proposition does not assert that the symmetric difference con-

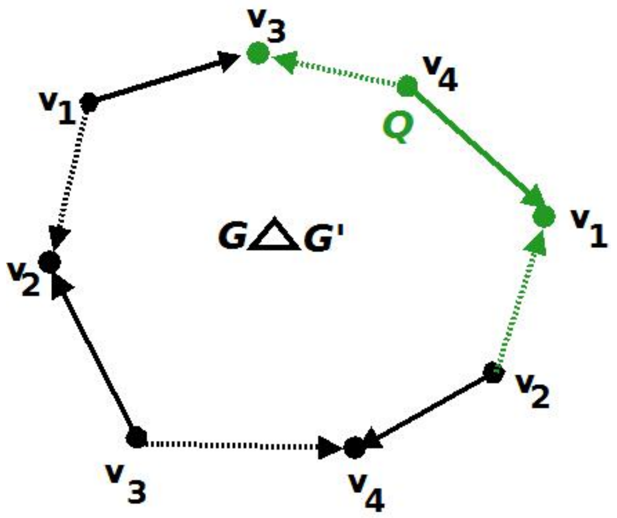


Figure 5.7: Symmetric difference $G\Delta G'$ which only possesses one type of path, namely Q, from Proposition 5.1.

tains P **and** Q. The smallest counter-example (see Figure 5.7) are the digraph realizations $G = (V, A)$ and $G' = (V, A')$ with $V = \{v_1, v_2, v_3, v_4\}$ and $A = \{(v_1, v_3), (v_3, v_2), (v_2, v_4), (v_4, v_1)\}$ and $A' = \{(v_1, v_2), (v_2, v_1), (v_3, v_4), (v_4, v_3)\}$. Here, we only find path Q.

Proposition 5.2. *Let S be a digraph sequence and G and G' be two different digraph realizations. If $|G\Delta G'| = 6$, then there exist*

a) digraph realizations G_0, G_1, G_2 with $G_0 := G$, $G_2 := G'$ and $|G_i\Delta G_{i+1}| = 4$ for $i \in \{0, 1\}$ or

b) G and G' are different in the orientation of exactly one directed 3-cycle.

Proof. First observe that the symmetric difference is weakly connected whenever $|G\Delta G'| = 6$. We consider the alternating 6-cycle $C := G\Delta G'$.

case 1: C contains at least four different vertices. Assume first that C contains four different vertices (see Figure 5.8).

The only possibility to realize this scenario is $C = (v_1, a_1, v_2, a_2, v_3, a_3, v_1, a_4, v_4, a_5, v_3, a_6, v_1)$ with $a_1 = (v_1, v_2), a_3 = (v_3, v_1), a_4 = (v_4, v_3) \in A(G)$ and $a_2 = (v_3, v_2), a_4 = (v_4, v_1), a_6 = (v_1, v_3) \in A(G')$. (A permutation of $\{1, 2, 3\}$ does not influence the result.) We get the alternating oriented path $P = (v_4, a_5, v_3, a_3, v_1, a_1, v_2)$.

(i): Assume $(v_4, v_2) \notin A(G)$. It follows $(v_4, v_2) \notin A(G')$. Otherwise, we would get

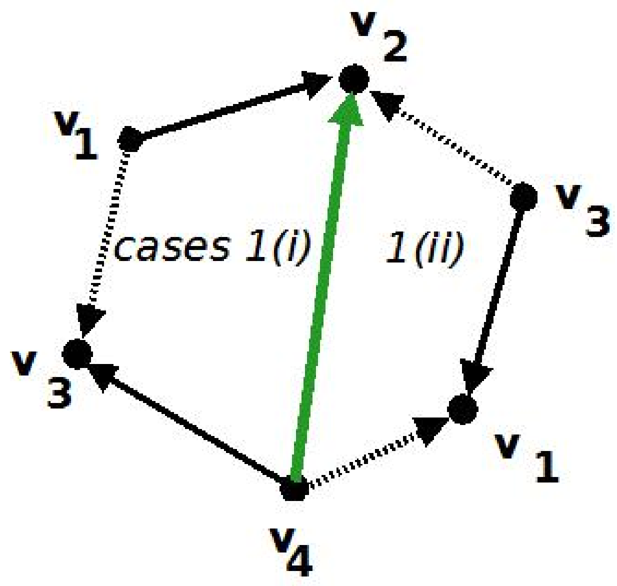


Figure 5.8: Case 1 (i) and (ii) from Proposition 5.2 with exactly four different vertices.

$(v_4, v_2) \in G\Delta G'$ in contradiction to our assumption. We set

$$G_1 := (G_0 \setminus \{(v_4, v_3), (v_1, v_2)\}) \cup \{(v_4, v_2), (v_1, v_3)\} \text{ and}$$

$$G_2 := (G_1 \setminus \{(v_4, v_2), (v_3, v_1)\}) \cup \{(v_4, v_1), (v_3, v_2)\}.$$

We get $G_2 = G'$ and digraph realizations G_0, G_1, G_2 with $|G_i \Delta G'_{i+1}| = 4$.

(ii): Assume $(v_4, v_2) \in A(G)$. It follows $(v_4, v_2) \in A(G')$. Otherwise, we would get $(v_4, v_2) \in G\Delta G'$ in contradiction to our assumption. We set

$$G_1 := (G_0 \setminus \{(v_4, v_2), (v_3, v_1)\}) \cup \{(v_4, v_1), (v_3, v_2)\} \text{ and}$$

$$G_2 := (G_1 \setminus \{(v_4, v_3), (v_1, v_2)\}) \cup \{(v_4, v_2), (v_1, v_3)\}.$$

We get $G_2 = G'$ and digraph realizations G_0, G_1, G_2 with $|G_i \Delta G'_{i+1}| = 4$.

We can argue analogously if C contains five our six different vertices.

case 2: C contains exactly three different vertices. Then C is the alternating oriented cycle $C = (v_1, a_1, v_2, a_2, v_3, a_3, v_1, a_4, v_2, a_5, v_3, a_6, v_1)$ with $a_1 = (v_1, v_2), a_3 = (v_3, v_1), a_5 = (v_2, v_3) \in A(G)$ and $a_2 = (v_3, v_2), a_4 = (v_2, v_1), a_6 = (v_1, v_3) \in A(G')$. Hence, G and G' are different in the orientation of exactly one directed 3-cycle.

□

Lemma 5.2. *Let S be a digraph sequence and G and G' be two different digraph realizations. Then there exist digraph realizations $G_0, G_1, \dots, G_k$ with $G_0 := G$, $G_k := G'$ and*

1. $|G_i \Delta G_{i+1}| = 4$ *or*
2. $|G_i \Delta G_{i+1}| = 6$

where $k \leq \frac{1}{2}|G\Delta G'| - 1$. In case (2), $G_i \Delta G_{i+1}$ consists of a directed 3-cycle and its opposite orientation.

Proof. We prove the lemma by induction according to the cardinality of the symmetric difference $|G\Delta G'| = 2\kappa$. For $\kappa := 2$ we get $|G\Delta G'| = 4$. The correctness of our claim follows with $G_1 := G'$. For $\kappa := 3$ we apply Proposition 5.2. In both of its cases it follows $k \leq 2$.

We assume the correctness of our claim for all $\kappa \leq \ell$. Let $|G\Delta G'| = 2\ell + 2$. We can assume that $\kappa > 3$. Assume further, that the symmetric difference consists of λ weakly connected components $(G\Delta G')_i$ for $i \in \{1, \dots, \lambda\}$.

Consider first the case that for all these components $|(G\Delta G')_i| = 6$ and that each component contains exactly three distinct vertices. Then each of them is a directed 3-cycle and its reorientation. We choose $(G\Delta G')_1$, perform a 3-cycle reorientation on it, and obtain digraph realization G^*. Thus $|G^*\Delta G'| = 2\ell - 4$. By the induction hypothesis, there are digraph realizations $G_0 = G^*, G_1, \dots, G_k = G'$ such that $k \leq \frac{1}{2}|G^*\Delta G'| - 1 = \frac{1}{2}|G\Delta G'| - 4$. Combining the first 3-cycle reorientation with this sequence of digraph realizations gives the desired bound. If there is a component $|(G\Delta G')_i| = 6$ with at least four distinct vertices, we can apply Proposition 5.2, case a) to it and handle the remaining components by induction.

Otherwise, there is a component with $|(G\Delta G')_i| = 4$ or $|(G\Delta G')_i| \geq 8$. For the first case we apply the same idea as before. Swap the arcs in $(G\Delta G')_i$ and obtain a digraph G^*. By the induction hypothesis one can find a sequence of digraph realizations $G^* = G_0, \dots, G_k = G'$ with $k \leq \frac{1}{2}|G^*\Delta G'| = \frac{1}{2}|G\Delta G'| - 3$. So let as consider the case $|(G\Delta G')_i| \geq 8$. Due to Proposition 5.1, we may assume that there is an alternating oriented path $P = (v_1, a_1, v_2, a_2, v_3, a_3, v_4)$ with $a_1 = (v_1, v_2), a_3 = (v_3, v_4) \in A(G)$ and $a_2 = (v_3, v_2) \in A(G')$. Otherwise, there exists $Q = (w_1, b_1, w_2, b_2, w_3, b_3, w_4)$ with $b_1 = (w_1, w_2), b_3 = (w_3, w_4) \in A(G')$ and $b_2 = (w_3, w_2) \in A(G)$. In that case we can exchange the roles of G and G' and consider $G'\Delta G$. Clearly, a sequence of digraph realizations $G' = G'_0, G'_1, \dots, G'_k = G$ can be reversed and then fulfills the conditions

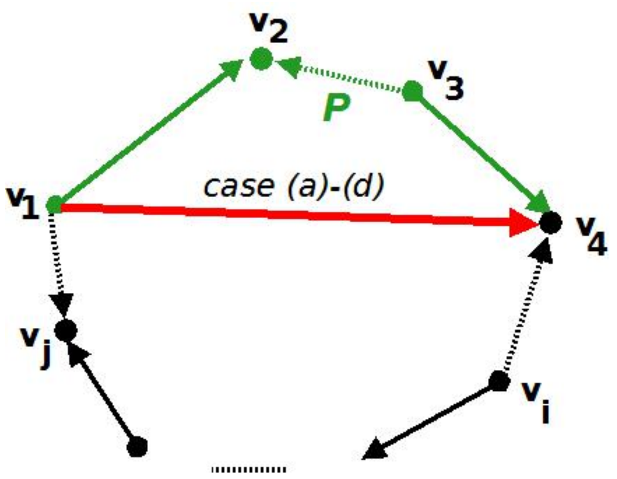


Figure 5.9: The four distinct cases of Lemma 5.2.

of the lemma. So from now on we work with P. For the following case distinction consider Figure 5.9.

case 1: Assume $(v_1, v_4) \in A(G') \setminus A(G)$.
This implies $(v_1, v_4) \in G\Delta G'$. $G_1 := (G_0 \setminus \{(v_1, v_2), (v_3, v_4)\}) \cup \{(v_3, v_2), (v_1, v_4)\}$ is a digraph realization of S and it follows $|G_0\Delta G_1| = 4$ and $|G_1\Delta G'| = 2\ell + 2 - 4 = 2(\ell - 1)$. Note that after this step, $G_1\Delta G'$ may consist of several connected components, but each of them has strictly smaller cardinality. Therefore, we can apply the induction hypothesis on $|G_1\Delta G'|$. Thus, we obtain digraph realizations $G_1, G_2, \ldots, G_k$ with $G_k := G'$ and $|G_i\Delta G_{i+1}| = 4$ or $|G_i\Delta G_{i+1}| = 6$ where $k - 1 \leq \frac{1}{2}|G_1\Delta G'| - 1$. Hence, we get the sequence $G_0, G_1, \ldots, G_k$ with $k = 1 + \frac{1}{2}|G_1\Delta G'| - 1 = \frac{1}{2}(|G\Delta G'| - 4) \leq \frac{1}{2}|G\Delta G'| - 1$ which fulfills 1. and 2.

case 2: Assume $(v_1, v_4) \in A(G) \cap A(G')$.
This implies $(v_1, v_4) \notin G\Delta G'$. Consider an alternating oriented cycle $C = (v_4, c_1, v_i, \ldots, v_j, c_2, v_1, a_1, v_2, a_2, v_3, a_3, v_4)$ of $(G\Delta G')_i$ such that each vertex has indegree two or outdegree two. Then P is an alternating subpath of C with $c_1 = (v_i, v_4), c_2 = (v_1, v_j) \in A(G')$. We construct a new alternating oriented cycle $C^* := (C \setminus P) \cup \{(v_1, v_4)\}$ with length $|C^*| = |C| - 2$. We swap the arcs in C^* and get a digraph realization G^* of S with $|G_0\Delta G^*| = |C^*| \leq 2\ell$ and $|G^*\Delta G'| = |G\Delta G'| - (|C^*| - 1) + 1 \leq 2\ell$. According to the induction hypothesis there exist sequences $G^1_0 := G, G^1_1, \ldots, G^1_{k_1} := G^*$ and $G^2_0 := G^*, G^2_1, \ldots, G^2_{k_2} = G'$ with $k_1 \leq \frac{1}{2}|G_0\Delta G^*| - 1$ and $k_2 \leq \frac{1}{2}|G^*\Delta G'| - 1$. We arrange these sequences one after another and get a sequence which fulfills 1. and 2. and $k = k_1 + k_2 =$

$\frac{1}{2}|G_0\Delta G^*|-1+\frac{1}{2}|G^*\Delta G'|-1=\frac{1}{2}|G\Delta G'|-1$.

case 3: Assume $(v_1,v_4)\in A(G)\setminus A(G')$.

This implies $(v_1,v_4)\in G\Delta G'$. The alternating oriented path P can be extended to an alternating oriented cycle $C=(v_1,a_1,v_2,a_2,v_3,a_3,v_4,a_4,v_5,\dots,v_{2t},a_{2t},v_1)$, $t\geq 3$ using only arcs from $G\Delta G'$. To construct C, start with P, and keep adding alternating arcs until you reach the start vertex v_1 for the first time. Obviously, you will not get stuck before reaching v_1. Note that the arc (v_1,v_4) does not belong to C. Therefore, there exists an alternating oriented sub-cycle $C^*:=C\cup\{(v_1,v_4)\}\setminus P$ formed by arcs in $G\Delta G'$. We swap the arcs in C^* and get a digraph realization G^* of S with $|G_0\Delta G^*|=|C^*|\leq 2\ell$ and $|G^*\Delta G'|=|G\Delta G'|-|C^*|\leq 2\ell$. According to the induction hypothesis there exist digraph sequences $G_0^1:=G,G_1^1,\dots,G_{k_1}^1:=G^*$ and $G_0^2:=G^*,G_1^2,\dots,G_{k_2}^2=G'$ with $k_1\leq\frac{1}{2}|G_0\Delta G^*|-1$ and $k_2\leq\frac{1}{2}(|G\Delta G'|-|C^*|)-1$. We arrange these sequences one after another and get a sequence which fulfills 1. and 2. and $k=k_1+k_2=\frac{1}{2}|G_0\Delta G^*|-1+\frac{1}{2}(|G\Delta G'|-|C^*|)-1=\frac{1}{2}(|G\Delta G'|)-2$.

case 4: Assume $(v_1,v_4)\notin A(G)\cup A(G')$. This implies $(v_1,v_4)\notin G\Delta G'$. It exists the alternating oriented cycle $C:=(P,(v_1,v_4))$ with $(v_1,v_4)\notin A(G)$. $G_1:=(G_0\setminus\{(v_1,v_2),(v_3,v_4)\})\cup\{(v_3,v_2),(v_1,v_4)\}$ is a digraph realization of S and it follows $|G_0\Delta G_1|=4$ and $|G_1\Delta G'|=2\ell+2-2=2\ell$. According to the induction hypothesis there exist digraph realizations $G_1,G_2,\dots,G_k$ with $G_k:=G'$ which fulfill 1. and 2. where $k_1:=1$ and $k_2:=k-1\leq\frac{1}{2}|G_1\Delta G'|-1$. Hence, we get the sequence $G_0,G_1,\dots,G_k$ with $k=k_1+k_2=1+\frac{1}{2}|G_1\Delta G'|-1=\frac{1}{2}(|G\Delta G'|-3+1)=\frac{1}{2}|G\Delta G'|-1$ which fulfills 1. and 2.

□

Corollary 5.1. *State digraph Φ is a strongly connected digraph with loops.*

5.2.2 Random Walks

A random walk on $\Phi=(V_\Phi,A_\Phi)$ can be described by Algorithm 3. We now require a data structure DS containing all pairs of non-adjacent arcs and all directed 2-paths in the current digraph realization.

Theorem 5.2. *Algorithm 3 is a random walk on state digraph Φ which samples uniformly at random a digraph realization $G'=(V,A)$ of digraph sequence S for $\tau\to\infty$.*

Algorithm 3 Sampling digraph realizations

```
Require: digraph sequence $S$, a digraph realization $G = (V, A)$ of $S$, a mixing time $\tau$.
Ensure: A sampled digraph realization $G' = (V, A')$ of $S$.
 1: $t := 0,\ G' := G$ {initialization}
 2: while $t < \tau$ do
 3:   Choose an element $p$ from $DS$ uniformly at random.{$p$ is a pair of non-adjacent
      arcs or a directed 2-path.}
 4:   if $p$ is a pair of non-adjacent arcs $(v_{i_1}, v_{i_2}), (v_{i_3}, v_{i_4})$ then
 5:     if $(v_{i_1}, v_{i_4}), (v_{i_3}, v_{i_2}) \notin A(G')$ then
 6:       {Either walk on $\Phi$ to an adjacent digraph realization $G'$}
 7:       Delete $(v_{i_1}, v_{i_2}), (v_{i_3}, v_{i_4})$ in $A(G')$.
 8:       Add $(v_{i_1}, v_{i_4}), (v_{i_3}, v_{i_2})$ to $A(G')$.
 9:     else
10:       {or walk a loop: ‘Do nothing’}
11:     end if
12:   else
13:     {$p$ is a directed 2-path $P = (v_{i_1}, v_{i_2}, v_{i_3})$}
14:     if $((v_{i_3}, v_{i_1}) \in A(G')) \wedge ((v_{i_2}, v_{i_1}), (v_{i_3}, v_{i_2}), (v_{i_1}, v_{i_3}) \notin A(G')) \wedge (i_3 > i_1) \wedge (i_3 >
        i_2)$ then
15:       {Walk on $\Phi$ to an adjacent digraph realization $G'$ with a reoriented directed
          3-cycle}
16:       Delete $(v_{i_1}, v_{i_2}), (v_{i_2}, v_{i_3}), (v_{i_3}, v_{i_1})$ in $A(G')$.
17:       Add $(v_{i_2}, v_{i_1}), (v_{i_3}, v_{i_2}), (v_{i_1}, v_{i_3})$ to $A(G')$.
18:     else
19:       {Walk a loop: ‘Do nothing’}
20:     end if
21:   end if
22:   update data structure $DS$
23:   $t \leftarrow t + 1$
24: end while
```

Proof. Algorithm 3 chooses elements in DS with the same constant probability. For a vertex $V_G \in V_\Phi$ there exist for all these pairs of arcs in $A(G')$ either incoming and outgoing arcs on $V_{G'}$ in Φ or a loop. Let $d_\Phi := \binom{|A(G)|}{2} - 2\sum_{i=1}^{n}\binom{a_i}{2}$. We get a transition matrix M for Φ with $p_{ij} = \frac{1}{d_\Phi}$ for $i,j \in A(\Phi), i \neq j$, $p_{ij} = 1 - \sum_{\{k|(k,l)\in A(\Phi),\ k\neq l\}} \frac{1}{d_\Phi}$ for $i,j \in V_\Phi, i = j$, otherwise we set $p_{ij} = 0$. Since, Φ is a regular, strongly connected, symmetrical and non-bipartite digraph, the distribution of all digraph realizations in the tth step converges asymptotically to the uniform distribution. $\square$

5.3 Arc-Swap Sequences

In this section, we study under which conditions the simple switching algorithm works correctly for digraphs. The Markov chain used in the switching algorithm works on the following simpler state digraph $\overline{\Phi} = (V_{\overline{\phi}}, A_{\overline{\Phi}})$. We define $A_{\overline{\Phi}}$ as follows.

a) We connect two vertices $V_G, V_{G'} \in V_{\overline{\Phi}}, G \neq G'$ with arcs $(V_G, V_{G'})$ and $(V_{G'}, V_G)$ if and only if $|G\Delta G'| = 4$ is fulfilled.

b) We set for each pair of non-adjacent arcs $(v_{i_1}, v_{i_2}), (v_{i_3}, v_{i_4}) \in A(G), i_j \in \{1,\ldots,n\}$ a directed loop (V_G, V_G) if and only if $(v_{i_1}, v_{i_4}) \in A(G) \vee (v_{i_3}, v_{i_2}) \in A(G)$.

c) We set one directed loop (V_G, V_G) for all $V_G \in V_{\overline{\Phi}}$.

Lemma 5.3. *The state digraph $\overline{\Phi} = (V_{\overline{\Phi}}, A_{\overline{\Phi}})$ is non-bipartite, symmetric, and regular.*

Proof. Since each vertex $V_G \in V_{\overline{\Phi}}$ contains a loop, $\overline{\Phi}$ is not bipartite. At each time we set an arc we also do this for its opposite direction. Hence, $\overline{\Phi}$ is symmetric. The number of incoming and outgoing arcs at each V_G equals the number of non-adjacent arcs in G, which is the constant value $\binom{|A(G)|}{2} - \left(\sum_{i=1}^{n}\binom{a_i}{2} + \sum_{i=1}^{n}\binom{b_i}{2} + \sum_{i=1}^{n} a_i b_i\right)$. Thus, we get the regularity of $\overline{\Phi}$. $\square$

5.4 Characterization of Arc-Swap Sequences

As shown in Example 5.1, $\overline{\Phi}$ decomposes into several components, but we are able to characterize digraph sequences S for which strong connectivity is fulfilled in $\overline{\Phi}$. In fact, we will show that there are numerous digraph sequences which only require switching by 2-swaps. In the following we give necessary and sufficient conditions allowing to identify such digraph sequences in polynomial running time.

Definition 5.1. *Let S be a digraph sequence and let $G = (V, A)$ be an arbitrary digraph realization. We denote a vertex subset $V' \subseteq V$ with $|V'| = 3$ as an* induced cycle set *V' if and only if for each digraph realization $G^* = (V, A^*)$ the induced subdigraph $G^* \langle V' \rangle$ is a directed 3-cycle.*

Definition 5.2. *Let S be a digraph sequence and $G = (V, A)$ an arbitrary digraph realization. We call S an* arc-swap-sequence *if and only if each subset $V' \subseteq V$ of vertices with $|V'| = 3$ is not an induced cycle set.*

This definition enables us to use another state digraph for sampling a digraph realization G for arc-swap-sequences. In Theorem 5.5, we will show that in these cases we have only to switch the ends of two non-adjacent arcs.

Before, we study how to recognize arc-swap-sequences efficiently. Clearly, we may not determine all digraph realizations to identify a digraph sequence as an arc-swap-sequence. Fortunately, we are able to give a characterization of digraph sequences allowing us to identify an arc-swap-sequence by only considering one digraph realization. We need a further definition for a special case of symmetric differences.

Definition 5.3. *Let S be a digraph sequence and $G = (V, A)$ and $G^* = (V, A^*)$ arbitrary digraph realizations. We call $G \Delta G^*$* simple symmetric cycle *if and only if each vertex $v \in V(G \Delta G^*)$ possesses indegree $d^-_{G\Delta G^*}(v) \leq 2$ and outdegree $d^+_{G\Delta(v)G^*} \leq 2$, and if $G \Delta G^*$ is an alternating oriented cycle.*

Note that the alternating oriented cycle C_1 (in Figure 2.2) is not a simple symmetric cycle, because $d^+_{C_1}(v_4) = 4$. Cycle C_1 decomposes into two simple symmetric cycles $C_1' = (v_1, v_2, v_3, v_4, v_1)$ and $C_1'' = (v_2, v_3, v_5, v_4, v_2)$.

Theorem 5.3. *A digraph sequence S is an arc-swap-sequence if and only if for any digraph realization $G = (V, A)$ the following property is true:*
For each induced, directed 3-cycle $G \langle V' \rangle$ of G there exists a digraph realization $G^* = (V, A^*)$ so that $G \Delta G^*$ is a simple symmetric cycle and that the induced subdigraph $G^* \langle V' \rangle$ is not a directed 3-cycle.

Proof. $\Rightarrow$: Let S be an arc-swap sequence and $G = (V, A)$ be an arbitrary digraph realization. With Definition 5.2 it follows that each subset $V' \subset V$ with $|V'| = 3$ is not an induced cycle set. Hence, there exists for each induced, directed 3-cycle $G \langle V' \rangle$ of G a digraph realization $G' = (V, A')$ where the induced subdigraph $G' \langle V' \rangle$ is not a directed cycle. If the symmetric difference $G \Delta G'$ is not a simple symmetric cycle we delete as long alternating oriented cycles in $G \Delta G'$ as we get an alternating oriented cycle C^* where each vertex in C^* has at most indegree two and at most outdegree two. Furthermore, C^* shall contain at least one arc $(v, v') \in V' \times V'$. This is possible, because $G \Delta G'$ contains at least one such arc. On the other hand the alternating oriented cycle C^* does not contain all possible six of such arcs. Otherwise, the induced subdigraph $G' \langle V' \rangle$ is a directed cycle. Now, we construct the digraph realization $G^* = (V, A^*)$ with $A^* := (A(G) \setminus A(C^*)) \cup (A(C^*) \cap A(G'))$. It follows $G \Delta G^* = C^*$ is a simple symmetric difference.

$\Leftarrow$: Let G be any digraph realization of digraph sequence S. We only have to consider 3-tuples of vertices V' inducing directed 3-cycles in G. With our assumption there exists for each V' a digraph realization G^* so that $G^* \langle V' \rangle$ is not a directed 3-cycle. Hence, we find for each subset $V' \subset V$ of vertices with $|V'| = 3$ a digraph realization $G^* = (V, A)$, so that the induced subdigraph $G^* \langle V' \rangle$ is not a directed 3-cycle. We conclude that S is an arc-swap sequence. $\square$

This characterization allows us to give a simple polynomial-time algorithm to recognize arc-swap-sequences. All we have to do is to check for each induced 3-cycle of the given digraph realization, if it forms an induced cycle set. Therefore, we check for each arc (v, w) in an induced 3-cycle whether there is an alternating oriented walk from v to w (not using arc (v, w)) which does not include all five remaining arcs of the 3-cycle and its reorientation. Moreover, each node on this walk has at most indegree 2 and at most outdegree 2. Such an alternating oriented walk can be found in linear time by using a reduction to an f-factor problem in a bipartite graph. In this graph we search for an alternating path by growing alternating trees (similar to matching algorithms in bipartite graphs, no complications with blossoms will occur), see for example (Sch03). The trick to ensure that not all five arcs will appear in the alternating cycle is to iterate over these five arcs and exclude exactly one of them from the alternating path search between v and w. Of course, this loop stops as soon as one alternating path is found. Otherwise, no such alternating path exists. A linear-time recognition is possible with a parallel Havel-Hakimi algorithm of LaMar (LaM09).

Next, we are going to prove that $\overline{\Phi}$ is strongly connected for arc-swap-sequences. The structure of the proof is similar to the case of Φ, but technically slightly more involved.

Lemma 5.4. *Let S be an arc-swap-sequence and G and G^* be two different digraph realizations. Assume $V' := \{v_1, v_2, v_3\} \subseteq V$ such that $G\langle V' \rangle$ is an induced directed 3-cycle but $G^*\langle V' \rangle$ is not an induced directed 3-cycle. Moreover, assume that $G \Delta G^*$ is a simple symmetric cycle. Then there are digraph realizations $G_0, G_1, \ldots, G_k$ with*

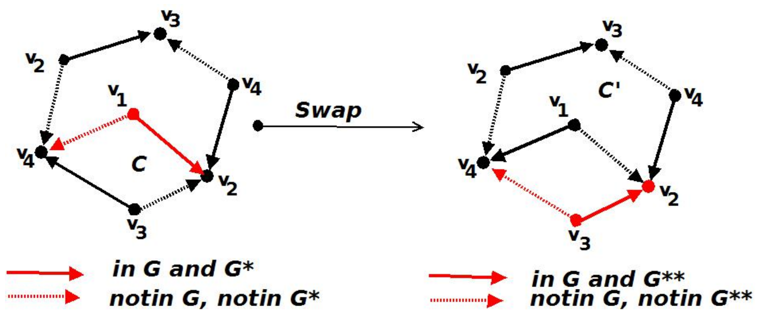


Figure 5.10: Lemma 5.4 case b): arc $(v_1, v_4) \notin A(G) \cup A(G^*)$.

$G_0 := G, G_k := G^*$, $|G_i \Delta G_{i+1}| = 4$ *and* $k \leq \frac{1}{2}|G\Delta G^*|$.

Proof. We prove this lemma by induction on the cardinality of $G\Delta G^*$. The base case $|G\Delta G^*| = 4$ is trivial. Consider next the case $|G\Delta G^*| = 6$. We distinguish between two subcases.

case a) $G\Delta G^*$ consists of at least four different vertices.

By Proposition 5.2, case a), there are realizations $G = G_0, G_1, G_2 = G^*$ with $|G_i \Delta G_{i+1}| = 4$.

case b) $G\Delta G^*$ consists of exactly three vertices.

Observe that $G\Delta G^*$ contains at least one arc from $G\langle V'\rangle$ or its reorientation but not all three vertices V', as otherwise $G^*\langle V'\rangle$ would be an induced 3-cycle. In fact, it turns out that $G\Delta G^*$ contains exactly one arc, say (v_2, v_3), from $G\langle V'\rangle$ and its opposite arc (v_3, v_2), because $G\Delta G^*$ is the alternating oriented cycle $(v_2, a_2, v_3, a_3, v_4, a_4, v_2, a_5, v_3, a_6, v_4, a_7, v_2)$ with $v_4 \neq v_1$. We have two subcases. Assume first that $(v_1, v_4) \notin A(G) \cup A(G^*)$, see Figure 5.10.

So we can swap the alternating oriented cycle $C := (v_1, v_2, v_3, v_4, v_1)$ in a single step and obtain digraph realization G^{**}. We then obtain the alternating oriented 6-cycle $C' := (v_2, v_3, v_4, v_2, v_1, v_4, v_2)$ which consists of four different vertices. By Proposition 5.2, case a), we can swap the arcs of this cycle in two steps, thus in total in three steps as claimed. Otherwise, $(v_1, v_4) \in A(G) \cap A(G^*)$, see Figure 5.11.

Then we obtain the alternating oriented cycle $C := (v_1, v_4, v_2, v_3, v_1)$ which can be swapped in a single step to digraph realization G^{**}. By that, we obtain a new cycle $C' := (v_1, v_3, v_4, v_2, v_3, v_4, v_1)$ which consists of four different vertices. By Proposition 5.2, case a), we can swap the arcs of this cycle in two steps, thus in total in three steps as claimed.

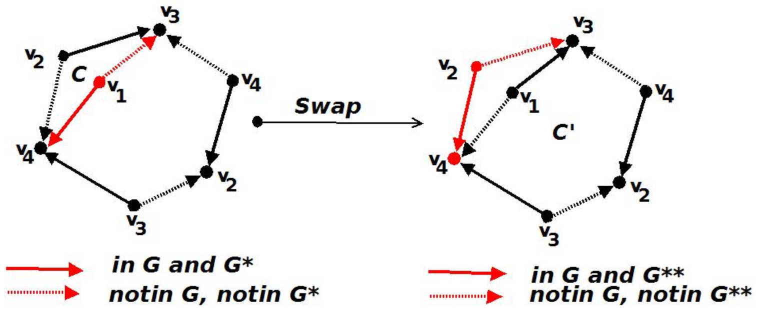


Figure 5.11: Lemma 5.4 case b): arc $(v_1, v_4) \in A(G) \cap A(G^*)$.

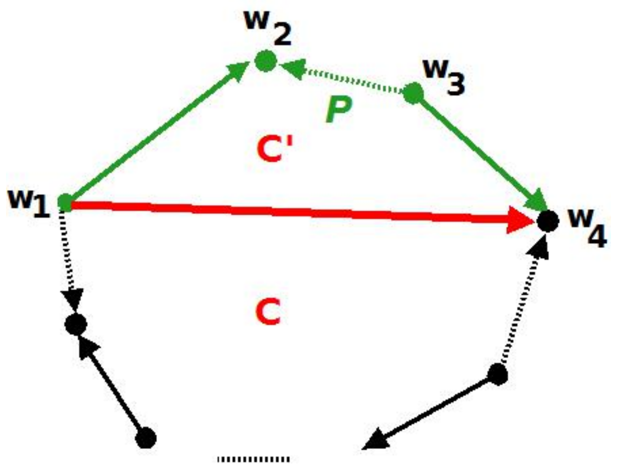


Figure 5.12: Lemma 5.4 Induction step: P contains no arc from $G\langle V'\rangle$ or its reorientation.

For the induction step, let us consider $|G\Delta G^*| = 2\ell + 2 \geq 8$. Then $G\Delta G^*$ contains between one and five arcs from $G\langle V'\rangle$ and its reorientation. By Proposition 5.1, there is an alternating oriented path $P = (w_1, b_1, w_2, b_2, w_3, b_3, w_4)$ in $G\Delta G^*$ with $(w_1, w_2) \in A(G) \setminus A(G^*)$ or $(w_1, w_2) \in A(G^*) \setminus A(G)$. Suppose that P contains no arc from $G\langle V'\rangle$ and its reorientation. We consider the case $(w_1, w_2) \in A(G) \setminus A(G^*)$. (The case $(w_1, w_2) \in A(G^*) \setminus A(G))$ can be shown with the same arguments.) Note, that $(w_1, w_4) \notin G\Delta G^*$, because $G\Delta G^*$ is a simple symmetric cycle with $|G\Delta G^*| \geq 8$. If $(w_1, w_4) \in A(G) \cap A(G^*)$, then we consider $C := (G\Delta G^*) \cup \{(w_1, w_4)\} \setminus P$, see Figure 5.12.

We swap the arcs of C and obtain a digraph realization G^{**}. Clearly, $G\Delta G^{**}$ contains an arc from $G\langle V'\rangle$ or its reorientation, and it is a simple symmetric cycle. Moreover, $G^{**}\langle V'\rangle$ is not an induced 3-cycle. As $|G\Delta G^{**}| = 2\ell$, we can apply the induction hypothesis. We obtain a sequence of realizations $G = G_0, G_1, \ldots, G_k = G^{**}$

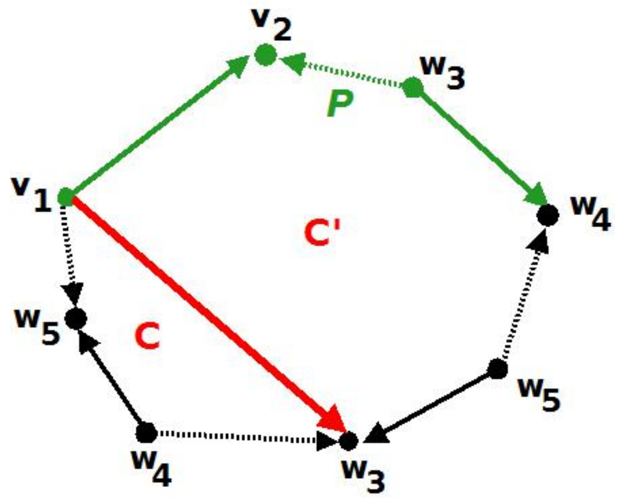


Figure 5.13: Lemma 5.4 Induction step: P contains exactly one arc from $G\langle V'\rangle$ and its reorientation

.

with $|G_i \Delta G_{i+1}| = 4$ and $k \leq \frac{1}{2}|G\Delta G^{**}| \leq \frac{1}{2}(|G\Delta G^*|-2)$. Finally, we apply a last swap on the alternating oriented cycle $(w_1, b_1, w_2, b_2, w_3, b_3, w_4, b_4, w_1)$ and thereby transform G^{**} to G^*. In total, the number of swap operations is at most $\frac{1}{2}|G\Delta G^*|$.

The case $b_4 = (w_1, w_4) \notin A(G) \cup A(G^*)$ is similar. This time, we start with a single swap on the alternating oriented cycle $C' := (w_1, b_1, w_2, b_2, w_3, b_3, w_4, b_4, w_1)$ and afterwards apply induction to the remaining cycle. Hence, in the remainder we may assume that there is an alternating oriented 3-path P.

Hence, we consider such an alternating oriented 3-path $P = (w_1, b_1, w_2, b_2, w_3, b_3, w_4)$ in $G\Delta G^*$ with $b_1 = (w_1, w_2) \in A(G) \setminus A(G^*)$ or $b_1 = (w_1, w_2) \in A(G^*) \setminus A(G)$ but at least one arc of P is from $G\langle V'\rangle$ and its reorientation.
Recall that $G\Delta G^*$ contains between one and five arcs from $G\langle V'\rangle$ and its reorientation. We distinguish between three cases:

case I: $G\Delta G^*$ contains exactly one of these arcs, say $(v_1, v_2) \in A(G)\setminus A(G^*)$. (The case that $(v_2, v_1) \in A(G^*) \setminus A(G)$ is this special arc can be treated analogously.) We claim that the cycle $G\Delta G^*$ must have the form $(v_1, v_2, w_3, w_4, w_5, w_3, w_4, w_5, v_1)$ see Figure 5.13.

Note that w_3, w_4, w_5 are repeated every third step, as otherwise we would obtain an alternating oriented path as excluded above. The cycle cannot be longer than eight, since then we would obtain an alternating oriented 3-path (w_3, w_4, w_5, w_6), excluded above. It might be that $w_4 = v_3$, but $w_3, w_5 \neq v_3$, as otherwise the

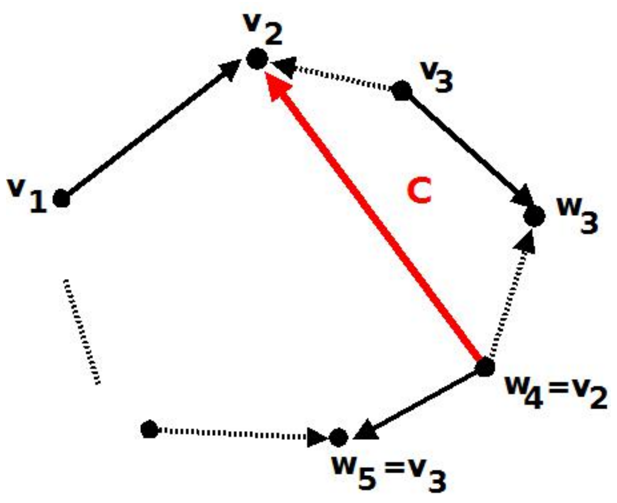


Figure 5.14: Lemma 5.4 Induction step: Two adjacent arcs from $G\langle V'\rangle$ and its reorientation in $G\Delta G^*$.

symmetric difference would contain more than one arc from $G\langle V'\rangle$ and its reorientation. If $(v_1, w_3) \in A(G) \cup A(G^*)$ there is the alternating oriented 4-cycle $C := (v_1, w_3, w_4, w_5, v_1)$ which can be swapped. In the remaining 6-cycle the arc (v_1, v_2) is contained, so the induction hypothesis can be applied. Otherwise, if $(v_1, w_3) \notin A(G) \cap A(G^*)$, we first apply the induction hypothesis to the 6-cycle $C' := (v_1, v_2, w_3, w_4, w_5, w_3, v_1)$, and afterwards we swap the remaining 4-cycle $C := (v_1, w_3, w_4, w_5, v_1)$.

case II: $G\Delta G^*$ contains exactly two arcs from $G\langle V'\rangle$ and its reorientation.
Suppose first that these two arcs are adjacent, say $(v_1, v_2), (v_3, v_2)$, see Figure 5.14.

Consider the following arcs $(v_3, w_3), (w_4, w_3), (w_4, w_5)$ along the symmetric difference. If $w_4 \neq v_2$ there is an alternating oriented path (v_2, v_3, w_3, w_4). Depending whether $(w_4, v_2) \in A(G) \cap A(G^*)$ or not, we can either swap the alternating oriented 4-cycle $C := (v_2, v_3, w_3, w_4, v_2)$ or the remaining part of the symmetric difference together with (w_4, v_2) by the induction hypothesis. Hence, we may assume $w_4 = v_2$. Moreover, $w_5 = v_3$, as otherwise there would be the alternating oriented 3-path (v_3, w_3, w_4, w_5) excluded above. But then (v_2, v_3) is also in the symmetric difference, a contradiction. Thus, the two arcs from $G\langle V'\rangle$ and its reorientation are not adjacent. Then, there are at least two other arcs between them (otherwise the one arc between them would also be from $G\langle V'\rangle$ and its reorientation). By our assumptions, there is an alternating oriented 3-path

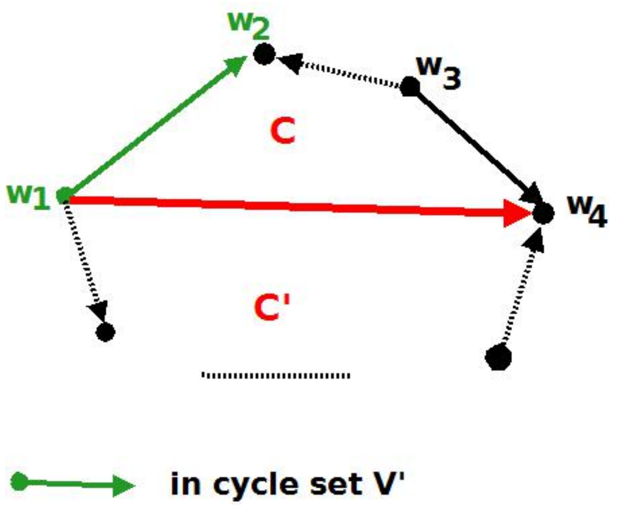


Figure 5.15: Lemma 5.4 Induction step: Two non-adjacent arcs from $G\langle V'\rangle$ and its reorientation in $G\Delta G^*$.

$P = (w_1, w_2, w_3, w_4)$ with at least one arc from $G\langle V'\rangle$ and its reorientation. In our scenario it must be exactly one such arc, see Figure 5.15.

Depending whether $(w_1, w_4) \in A(G) \cap A(G^*)$ or not, we can either swap the alternating oriented 4-cycle $C := (w_1, w_2, w_3, w_4, w_1)$ or the remaining part of the symmetric difference together with (w_1, w_4) (yielding cycle C') by the induction hypothesis.

case III: $G\Delta G^*$ contains between three and five arcs of $G\langle V'\rangle$ and its reorientation. Suppose first that all of them follow consecutively on the alternating oriented cycle. Consider the last two of these arcs, and append the next arc which must end in a vertex $v_4 \notin V'$. Then we have an alternating oriented 3-path which contains two arcs from $G\langle V'\rangle$ and its reorientation, and the remaining part of the symmetric difference also has such an arc. Thus we can apply the induction hypothesis and are done. Otherwise the three to five arcs from $G\langle V'\rangle$ and its reorientation are separated. So no alternating oriented 3-path may contain all of them, in particular not P. We can proceed as in case II.

□

Proposition 5.3.
Let S be an arc-swap-sequence and G and G' be two different digraph realizations. If $|G\Delta G'| = 6$ and $G\Delta G'$ consists of exactly three vertices $V' := \{v_1, v_2, v_3\}$, then there

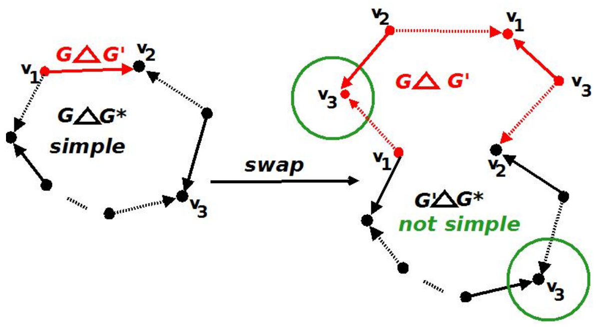


Figure 5.16: An example for the symmetric differences $G\Delta G^{*}$ and $G^{'}\Delta G^{*}$. Exactly one arc $(v_1, v_2) \in G\Delta G'$ is contained in $G\Delta G^{*}$.

exist digraph realizations $G_0, G_1, \ldots, G_k$ *with* $G_0 := G, G_k := G'$, $|G_i \Delta G_{i+1}| = 4$ *and* $k \leq 2n+2$.

Proof. Since S is an arc-swap-sequence, Theorem 5.3 implies the existence of a digraph realization G^{*} such that $G\Delta G^{*}$ is a simple symmetric cycle and $G^{*}\langle V'\rangle$ is not a directed 3-cycle. By Lemma 5.4, there are digraph realizations $G_0, G_1, \ldots, G_{k'} := G^{*}$ with $|G_i \Delta G_{i+1}| = 4$ and $k' \leq \frac{1}{2}|G\Delta G^{*}| \leq n$, since $G\Delta G^{*}$ is simple. Moreover, we have $|G^{*}\Delta G'| \leq |G\Delta G^{*}| + 4 \leq 2n+4$ since G and G' differ only in their orientation of the 3-cycle induced by V'. Let us consider the set of possible arcs in $G^{*}\Delta G'$. We distinguish between the following cases:

1.) $a \in G\Delta G^{*} \wedge a \notin G\Delta G'$. A simple case distinction yields $a \in G'\Delta G^{*}$.

2.) $a \in G\Delta G' \wedge a \in G\Delta G^{*}$. In this case, it follows $a \notin G'\Delta G^{*}$.

3.) $a \in G\Delta G' \wedge a \notin G\Delta G^{*}$. It follows $a \in G'\Delta G^{*}$.

Hence, we can conclude: All arcs which are not contained in $G\Delta G'$ but in $G\Delta G^{*}$ are also arcs in $G'\Delta G^{*}$. Arcs which appear in the alternating oriented 6-cycle $G\Delta G'$ and in the symmetric difference $G\Delta G^{*}$ are not contained in $G^{'}\Delta G^{*}$. They are “replaced” by the remaining arcs of 6-cycle $G\Delta G'$. In Figure 5.16 we give an example for a scenario where exactly one arc from $G\Delta G'$ is contained in $G\Delta G^{*}$.

The symmetric difference $G^{*}\Delta G'$ is not necessarily a simple symmetric cycle as can be seen in Figure 5.16, but can be decomposed into simple symmetric cycles, each containing at least one arc from $G\langle V'\rangle$ and its reorientation. The reason is that vertices with a larger in- or outdegree than two in $G'\Delta G^{*}$ can only be generated from vertices in $G\Delta G'$. Hence, each simple symmetric cycle has to start (or to end) with an arc from $G\Delta G'$ in $G'\Delta G^{*}$. For each of these simple symmetric cycles we apply our auxiliary

Lemma 5.4. We obtain a sequence $G^* := G'_0, \dots, G'_{k''} := G'$ with $|G_i \Delta G_{i+1}| = 4$ and $k'' \leq n+2$, because we have $|G' \Delta G^*| \leq 2n+4$. Combining both sequences we obtain a sequence with $k = k' + k'' \leq 2n+2$. □

Lemma 5.5. *Let S be an arc-swap-sequence, and G and G' be two different digraph realizations. Then there exist digraph realizations $G_0, G_1, \dots, G_k$ with $G_0 := G$, $G_k := G'$ and $|G_i \Delta G_{i+1}| = 4$, where $k \leq \left(\frac{1}{2}|G \Delta G'| - 1\right) \cdot (n+1)$.*

Proof. We prove the lemma by induction according to the cardinality of the symmetric difference $|G \Delta G'| = 2\kappa$. For $\kappa := 2$ we get $|G \Delta G'| = 4$. The correctness of our claim follows with $G_1 := G'$.

For $\kappa := 3$ we distinguish two cases. If $G \Delta G'$ consists of exactly three vertices, then by Proposition 5.3 we get a sequence of digraph realizations $G_0, G_1, \dots, G_k$ and $k \leq 2n+2 = 2(n+1)$, as claimed. Otherwise, the symmetric difference $G \Delta G'$ consists of more than three vertices. By Proposition 5.2, case a), there are digraph realizations $G_0, G_1, G_2 = G'$.

We assume the correctness of our induction hypothesis for all $\kappa \leq \ell$. Let $|G \Delta G'| = 2\ell + 2$. We can assume that $\kappa > 3$. Suppose first that the symmetric difference $G \Delta G'$ decomposes into t simple symmetric cycles of length $|(G \Delta G')_i| = 6$. Suppose further that all these $(G \Delta G')_i$ consist of exactly three vertices. Clearly, $|G \Delta G'| = 6t$. We apply our Proposition 5.3 to each of these t cycles one after another and get a sequence of digraph realizations $G_0, G_1, \dots, G_k = G'$ with $k \leq 2t(n+1) \leq (3t-1)(n+1) = \left(\frac{1}{2}|G \Delta G'| - 1\right) \cdot (n+1)$.

Otherwise, there is a $(G \Delta G')_1$ which contains at least four vertices. Swapping the arcs in $(G \Delta G')_1$ leads to a digraph realization G^*. By Proposition 5.2, there are digraph realizations $G = G_0, G_1, G_2 = G^*$ with $|G_i \Delta G_{i+1}| = 4$. We can apply the induction hypothesis on the remaining part of the symmetric difference. Obviously, we obtain the desired bound in this case.

It remains the case that there exists a simple symmetric cycle $(G \Delta G')_1$ of $G \Delta G'$ with $|(G \Delta G')_1| \neq 6$. If $|(G \Delta G')_1| = 4$, we use a single swap on $(G \Delta G')_1$ and obtain a digraph realization G^*, where $|G^* \Delta G'| = |G \Delta G'| - 4$. By the induction hypothesis, there is a sequence of digraph realizations $G^* = G_1, G_2 \dots, G_k = G'$ with $k - 1 \leq \left(\frac{1}{2}|G^* \Delta G'| - 1\right) \cdot (n+1) = \left(\frac{1}{2}|G \Delta G'| - 3\right) \cdot (n+1)$. Otherwise, $|(G \Delta G')_1| \geq 8$. Using Proposition 5.1, we may assume that there exists an alternating oriented path $P = (v_1, v_2, v_3, v_4)$ in $(G \Delta G')_1$ with $(v_1, v_2), (v_3, v_4) \in A(G)$ and $(v_3, v_2) \in A(G')$, for the same reasons as in the proof of Lemma 5.2.

case 1: Assume $(v_1, v_4) \in A(G') \setminus A(G)$.

This implies $(v_1, v_4) \in G\Delta G'$. $G_1 := (G\setminus\{(v_1,v_2),(v_3,v_4)\})\cup\{(v_3,v_2),(v_1,v_4)\}$ is a digraph realization of S and it follows $|G\Delta G_1| = 4$ and $|G_1\Delta G'| = 2\ell + 2 - 4 = 2(\ell - 1)$. Therefore, we can apply the induction hypothesis on $|G_1\Delta G'|$. Thus, we obtain digraph realizations $G_1, G_2, \dots, G_k$ with $G_k := G'$ and $|G_i\Delta G_{i+1}| = 4$. where $k - 1 \le \left(\frac{1}{2}|G_1\Delta G'| - 1\right)\cdot(n+1) = \left(\frac{1}{2}|G\Delta G'| - 3\right)\cdot(n+1)$.

case 2: Assume $(v_1, v_4) \in A(G)\cap A(G')$.
This implies $(v_1, v_4) \notin G\Delta G'$. We construct a new alternating oriented cycle $C^* := ((G\Delta G')_1 \setminus P)\cup\{(v_1,v_4)\}$ with length $|C^*| = |(G\Delta G')_1| - 2$. We swap the arcs in C^* and get a digraph realization G^* of S with $|G\Delta G^*| = |C^*| \le 2\ell$ and $|G^*\Delta G'| = |G\Delta G'| - (|C^*| - 1) + 1 \le 2\ell$. According to the induction hypothesis there exist digraph sequences $G_0^1 := G, G_1^1, \dots, G_{k_1}^1 := G^*$ and $G_0^2 := G^*, G_1^2, \dots, G_{k_2}^2 = G'$ with $k_1 \le \left(\frac{1}{2}|G_0^1\Delta G^*| - 1\right)\cdot(n+1)$ and $k_2 \le \left(\frac{1}{2}|G^*\Delta G'| - 1\right)\cdot(n+1)$. We arrange these digraph sequences one after another and get a sequence with $k = k_1 + k_2 = \left(\frac{1}{2}|G_0^1\Delta G^*| - 1 + \frac{1}{2}|G^*\Delta G'| - 1\right)\cdot(n+1) \le \left(\frac{1}{2}|G\Delta G'| - 1\right)\cdot(n+1)$.

case 3: Assume $(v_1, v_4) \in A(G)\setminus A(G')$.
This implies $(v_1, v_4) \in G\Delta G'$. Note that the arc (v_1, v_4) does not belong to $(G\Delta G')_1$. Therefore, there exists an alternating oriented sub-cycle
$C^* := (G\Delta G')_1 \cup \{(v_1,v_4)\}\setminus P$ formed by arcs in $G\Delta G'$. We swap the arcs in C^* and get a digraph realization G^* of S with $|G\Delta G^*| = |C^*| \le 2\ell$ and $|G^*\Delta G'| = |G\Delta G'| - |C^*| \le 2\ell$. According to the induction hypothesis there exist digraph sequences $G_0^1 := G, G_1^1, \dots, G_{k_1}^1 := G^*$ and $G_0^2 := G^*, G_1^2, \dots, G_{k_2}^2 = G'$ with $k_1 \le \left(\frac{1}{2}|G\Delta G^*| - 1\right)\cdot(n+1)$ and $k_2 \le \left(\frac{1}{2}(|G\Delta G'| - |C^*|) - 1\right)\cdot(n+1)$. We arrange these digraph sequences one after another and get a sequence with $k = k_1 + k_2 = \left(\frac{1}{2}|G\Delta G^*| - 1 + \frac{1}{2}(|G\Delta G'| - |C^*|) - 1\right)\cdot(n+1) = \left(\frac{1}{2}(|G\Delta G'|) - 2\right)\cdot(n+1)$.

case 4: Assume $(v_1, v_4) \notin A(G)\cup A(G')$.
This implies $(v_1, v_4) \notin G\Delta G'$. It exists the alternating oriented cycle $C := (P, (v_1, v_4))$ with $(v_1, v_4) \notin A(G)$. $G_1 := (G\setminus\{(v_1,v_2),(v_3,v_4)\})\cup\{(v_3,v_2),(v_1,v_4)\}$ is a digraph realization of S and it follows $|G\Delta G_1| = 4$ and $|G_1\Delta G'| = 2\ell + 2 - 2 = 2\ell$. According to the induction hypothesis there exist digraph realizations $G_1, G_2, \dots, G_k$ with $G_k := G'$ where $k_1 := 1$ and $k_2 := k - 1 \le \left(\frac{1}{2}|G_1\Delta G'| - 1\right)\cdot(n+1)$. Hence, we get the sequence $G := G_0, G_1, \dots, G_k$ with $k = k_1 + k_2 = 1 + \left(\frac{1}{2}|G_1\Delta G'| - 1\right)\cdot(n+1) \le \left(\frac{1}{2}|G\Delta G'| - 1\right)\cdot(n+1)$.

□

Corollary 5.2. *State digraph $\overline{\Phi}$ is a strongly connected digraph if and only if a given digraph sequence S is an arc-swap-sequence.*

An arc-swap-sequence implies the connectedness of the state digraph $\overline{\Phi}$. Therefore, for such sequences we are able to make random walks on state digraph $\overline{\Phi}$ which can be implemented easily. We simplify the random walk Algorithm 3 for arc-swap-sequences by using state digraph $\overline{\Phi}$. Hence, our data structure DS only contains pairs of non-adjacent arcs. We can ignore lines 12 to 21 in Algorithm 3. We denote this modified algorithm as the *arc-swap-algorithm.*

Theorem 5.4. *The arc-swap-algorithm is a random walk on the state digraph $\overline{\Phi}$ which uniformly samples a digraph realization $G = (V, A)$ of an arc-swap-sequence S for $\tau \to \infty$.*

Proof. The arc-swap-algorithm chooses all elements in DS with the same constant probability. For a vertex $V_G \in V_{\overline{\Phi}}$ there exist for all these pairs of arcs in $A(G')$ either incoming and outgoing arcs on $V_{G'} \in V_{\overline{\Phi}}$ or a loop. Define $d := \binom{|A(G)|}{2} - 2\sum_{i=1}^{n}\binom{a_i}{2} - \sum_{i=1}^{n} a_i b_i$. We get a transition matrix M for $\overline{\Phi}$ with $p_{ij} = \frac{1}{d}$ for $i, j \in A_{\overline{\Phi}}, i \neq j$, $p_{ii} = 1 - \sum_{\{k|(k,l)\in A_{\overline{\Phi}}\}} \frac{1}{d}$ for $i \in V_{\Phi}$, otherwise we set $p_{ij} = 0$. Since, $\overline{\Phi}$ is a regular, strongly connected, symmetric, and non-bipartite digraph, the distribution of all digraph realizations in a tth step converges asymptotically to the uniform distribution, see Lovász (Lov96). □

5.5 Practical Insights and Applications

Many "practitioners" use the switching algorithm for the purpose of network analysis, regardless whether the corresponding digraph sequence is an arc-swap-sequence or not. In this section we would like to discuss under which circumstances this common practice can be well justified and when it may lead to wrong conclusions.

What would happen if we sample using the state digraph $\overline{\Phi}$ for a digraph sequence S which is not an arc-swap-sequence? Clearly, we get the insight that $\overline{\Phi}$ has several connected components, but as we will see $\overline{\Phi}$ consists of at most $2^{\lfloor\frac{|V|}{3}\rfloor}$ isomorphic components containing exactly the same digraph realizations up to the orientation of directed 3-cycles each consisting of an induced cycle set V'. Fortunately, we can identify all induced cycle sets using our results in Theorem 5.3 by only considering an arbitrary digraph realization G.

Proposition 5.4. *Let S be a digraph sequence which is not an arc-swap-sequence and has at least two different induced cycle sets V' and V''. Then it follows $V' \cap V'' = \emptyset$.*

Proof. Without loss of generality we can label the vertices in V' with v'_1, v'_2, v'_3 and in V'' with v''_1, v''_2, v''_3. Let G be a digraph realization where

$$\{(v'_1, v'_2), (v'_2, v'_3), (v'_3, v'_1), (v''_1, v''_2), (v''_2, v''_3), (v''_3, v''_1)\} \subset A(G).$$

We distinguish between two cases.

a): Assume $|V' \cap V''| = 1$ where $v'_1 = v''_1$. If arc $(v''_3, v'_3) \in A(G)$ exists, we find the alternating oriented 4-cycle $(v''_3, v'_3, v'_1, v''_2, v''_3)$ which implies a new digraph realization G^* where $G^* \langle V' \rangle$ is not an alternating oriented cycle in contradiction to our assumption that V' is an induced cycle set. Hence, it follows $(v''_3, v'_3) \notin A(G)$. In this case we find the alternating oriented cycle $(v''_3, v'_1, v'_2, v'_3, v''_3)$.

b): Assume $|V' \cap V''| = |\{v'_1, v'_2\}| = 2$ where $v'_1 = v''_1$ and $v'_2 = v''_2$. If arc $(v''_3, v'_3) \notin A(G)$ we find the alternating oriented 4-cycle $(v'_1, v''_3, v'_3, v'_2, v'_1)$ which implies a new digraph realization G^* where $G^* \langle V' \rangle$ is not a directed cycle in contradiction to our assumption that V' is an induced cycle set. Hence, it follows $(v''_3, v'_3) \in A(G)$. In this case we find the alternating oriented cycle $(v''_3, v'_3, v'_1, v'_2, v''_3)$.

□

As the induced 3-cycles which appear in every digraph realization are vertex-disjoint, we can reduce the in- and outdegrees of all vertices in these cycles by one, and obtain a new digraph sequence which must be an arc-swap sequence.

Theorem 5.5. *Let S be a digraph sequence. Then the state digraph $\overline{\Phi}$ consists of at most $2^{\lfloor \frac{|V|}{3} \rfloor}$ isomorphic components.*

Proof. We assume that S is not an arc-swap-sequence, otherwise we apply Theorem 5.2 and get a strongly connected digraph $\overline{\Phi}$. With Proposition 5.4 it follows the existence of at most $\lfloor \frac{|V|}{3} \rfloor$ induced cycle sets for S. Consider all digraph realizations G^j possessing a fixed orientation of these induced 3-cycles which implies $G^j \langle V_i \rangle = G^{j'} \langle V_i \rangle$ for all such digraph realizations. We pick out one of these orientation scenarios and consider the symmetric difference $G^j \Delta G^{j'}$ of two such digraph realizations. Since, all induced 3-cycles are identical in G^j and $G^{j'}$, we delete all arcs of these induced cycle sets V_i and get the reduced graphs G^j_c and $G^{j'}_c$. Both are digraph realizations of an arc-swap-sequence S'. This is indeed the case, because $V_i := \{v_1, v_2, v_3\}$ has to be an independent set in each digraph realization G' of S'. Clearly, there cannot exist a single arc $(v_1, v_2) \in A(G')$ between v_1 and v_2, because this results in a digraph realization G of S, where V_i is not an induced cycle set. Assume both arcs (v_1, v_2) and (v_2, v_1) are contained in $A(G')$. Then $G^j \Delta G'$ contains two different simple symmetric cycles C_1, C_2 with one of these arcs in each of them. Note, that C_1 cannot consist of two opposite arcs between vertices of V_i, otherwise it is not simple. We swap the arcs in C_1 and get a digraph realization G' of S' with one single arc (v_2, v_1) between v_1 and v_2. We can apply this process until we get a digraph realization only containing one single arc between vertices of induced cycle sets V_i of S. Contradiction, because then we can add the arcs of one oriented cycle between these vertices in G' and get a digraph realization of S where V_i is not an induced cycle set. Hence, each digraph realization G' of S' possesses the independent vertex sets V_i. We conclude that G' cannot contain a new induced cycle set. Hence, S' is an arc-swap-sequence. Applying Lemma 5.5 we obtain, that there exist digraph realizations $G_0 := G_c^j, \dots, G_k := G_c^{j'}$ $|G_i \Delta G_{i+1}| = 4$, $k \leq \frac{1}{2}(|G_c^j \Delta G_c^{j'}| - 1)(n+1)$. Hence, each induced subdigraph $\overline{\Phi}\left\langle \{V_{G^j} | V_{G^j} \in V_{\overline{\phi}} \text{ and } G^j \text{ is a realization for one fixed orientation scenario}\} \right\rangle$ is strongly connected. On the other hand, we get for each fixed orientation scenario exactly the same digraph realizations G^j. Since, all induced 3-cycles are isomorphic, it follows that all digraph realizations which are only different in the orientation of such induced cycle sets are isomorphic. By Theorem 5.3, there does not exist an alternating oriented cycle destroying an induced 3-cycle. Hence, the state digraph $\overline{\Phi}$ consists of exactly 2^k strongly connected isomorphic components where k is the number of induced cycle sets V_i. □

We propose a new sampling algorithm *swap-shuffling algorithm.*

1. Sample in one component with 2-swaps and 3-cycle reorientations of directed induced 3-cycles which are not induced cycle sets

2. At the end use a permutation sampling ("card shuffling") to choose one of the k isomorphic components uniformly.

The latter point can be applied by using one of the well-known permutation samplings, for example *random transpositions.* This Markov chain is rapidly mixing (in $O(k^2)$), see Diaconis and Shahshahani (DM81). In point 1 we propose 3-cycle orientations because we conjecture a rapidly mixing Markov chain which is by one order smaller than without this operation. Our intuition is guided by the well-known lower bound for the mixing time τ_ϵ of a random walk which is half the diameter of the state digraph for any $\epsilon < \frac{1}{2}$ (see for example chapter 7 of the book by Levin, Peres and Wilmer (LPW06)). The diameter without 3-cycle reorientations is $O(mn)$, see Theorem 5.5, and with 3-cycle reorientations we find a diameter of $O(m)$, see Theorem 5.2.

Applications in network analysis. Since the switching algorithm samples for non-arc-swap sequences only in one single component of $\overline{\Phi}$, one has to be careful to get the correct estimations for certain network statistics. For network statistics on unlabeled digraphs, it suffices to sample in a single component which reduces the size of $V_{\overline{\Phi}}$ by a factor of 2^k, the number of components in $\overline{\Phi}$, where k is the number of induced cycle sets of the digraph sequence. Examples where this approach is feasible are network statistics like the average diameter or the motif content over all digraph realizations. For labelled digraphs, however, the random walk on $V_{\overline{\Phi}}$ systematically over- and under-samples the probability that an arc is present. Suppose that the random walk starts with a digraph realization $G = (V, A)$. If an arc $(v_1, v_2) \in A(G)$ belongs to an induced cycle set, it appears with probability 1 in all digraph realizations of the random walk. The opposite arc $(v_2, v_1) \notin A(G)$, will never occur. In an unbiased sampling over all digraph realizations, each of these arcs, however, occurs with probability 1/2. All other arcs occur with the same probability in a single component of $V_{\overline{\Phi}}$ as in the whole state digraph. This observation can be used to compute correct probabilities for all arcs.

6

Conclusion and Future Work

“Was wirklich zählt, ist Intuition.” (Albert Einstein)

Dag realization problem. In Chapter 4, we discussed the problem to find a valid topological sorting of a given dag sequence. The reason is that in this case it is easy to determine a dag realization. We defined a new relation “opposed” resulting in a special class of opposed sequences for which we are able to determine one topological ordering. Furthermore, we can restrict the number of possible topological orderings for all sequences. On the other hand, it is clear that there is still a gap in our understanding. Only few days ago, the complexity status was settled as NP-complete by Nichterlein (Nic11). Hence, several new questions arise, for example:

1. Is the dag realization problem fixed parameter tractable?

2. Can we extend the class of opposed sequences for which the problem is polynomial-time solvable?

3. Is it possible to characterize dag sequences which are solvable with the lexmax-strategy such that we are able to estimate the fraction of such sequences with a given number of n tuples and density m?

A further problem is to uniformly sample a dag realization for a given dag sequence. The development of a randomized algorithm for a subclass of dag sequences seems to be a promising option as long as the problem is open. Furthermore, the theoretical investigation and characterization of subclasses with a large success probability seems to be an interesting approach.

Uniform sampling of digraph sequences. In Chapter 5, we proposed a Markov chain which makes it possible to choose a uniformly sampled solution of each digraph sequence in theory. In practice, it remains as an open problem whether this chain is rapidly mixing or not. Up to now, it is not clear whether the well-known proof strategy "canonical path method" (Sin92) cannot be applied in our scenario or is "only" too complicated and too difficult to apply. The idea is to find in a given state digraph $G_\Phi = (V_\Phi, A_\Phi)$ between all pairs of vertices a shortest path such that each no arc is used by more than $|V_\Phi|$ paths. In this case, it has been proven that we get a rapidly mixing Markov chain. Hence, we want to find out in systematic experiments if we are able to construct canonical paths with such a property or not. Clearly, the state digraphs are often too large to be handled explicitly. On the other hand, we observe that the number of digraph realizations not only depends on the size of a given sequence but also on the property to have a small distance to its *threshold sequence*. Threshold sequences can be found in the literature as a special type of graph sequences. An excellent overview is given in the book of Mahadev and Peled (MP95). However, this concept can easily be extended to digraph sequences with and without loops. Hence, we will give a general definition with respect to one of these characteristics.

Definition 6.1 (threshold sequence)**.** *Given is a sequence S which is a graph sequence (digraph sequence, digraph sequence with loops). We denote S by* threshold sequence *if and only if there exists exactly one unique graph realization G (digraph realization, digraph realization with loops).*

Clearly, an equivalent characterization is that it is not possible to find an alternating cycle (alternating oriented cycles) with alternating arcs in G and not in G. The number of different characterizations for such sequences in the context of undirected graphs is very impressive leading to relations such as split graphs and to the so-called *polytope of degree sequences*. It turns out that a graph sequence S is an extreme point of this polytope if and only if S is a threshold sequence (Kor73). Here, we concentrate on a different connection to Ferrers matrices (Chapter 3.2). The corresponding sequence of a Ferrers matrix only possesses one unique digraph realization (with loops in the case of canonical Ferrers matrices). In fact, it is proven for graph sequences that a sequence is a threshold sequence if and only if it is the corresponding sequence for its Ferrers matrix (HIS81). This result can easily be extended to the directed case using the argument that no alternating oriented cycle exists. We conjecture that an arbitrary digraph sequence with a "small distance" to its threshold distance only possesses a small number of digraph realizations. In such a case one could determine the distance to its threshold sequence and then estimate the number of digraph realizations. For that, we need a measure for the distance which is known for graph sequences ((PA94)). We extend this measure to the directed case.

Definition 6.2 (threshold distance)**.** *Given are a digraph sequence* $S = \binom{a_1}{b_1}, \dots, \binom{a_n}{b_n}$, *its diagonal-free Ferrers matrix* F *and the corresponding sequence* $S' = \binom{a'_1}{b_1}, \dots, \binom{a'_n}{b_n}$. *We define the* threshold distance $\delta(S) := \sum_{i=1}^{n}(a'_i - a_i)^+$ *as the sum of all the positive differences* $(a'_i - a_i)^+ := \max(a'_i - a_i, 0)$ *to* S'.

We conjecture that one can find a relation between the size of the threshold distance of S and the number of its digraph realizations.

Further related open problems. An interesting related problem is to find a digraph realization of a sequence where directed 2-cycles are forbidden. We denote this problem by *orientation realization problem* and denote a corresponding digraph $G := (V, A)$ as *orientation realization.* We ask for the complexity status for this problem. Furthermore, we want to find a solution for the corresponding uniform sampling problem.

A similar unsolved problem is to find a solution for *mixed graphs.* A mixed graph $G = (V, M)$ consists of vertices where each pair of vertices can be connected via one directed arc, by two opposed directed arcs or by one undirected edge. Directed Arcs and undirected edges are not allowed simultaneously between the same pairs of vertices. Hence, one vertex v has an undirected degree $d_G(v)$, an indegree $d_G^-(v)$ and an outdegree $d_G^+(v)$. Now, we state the following problem.

Problem 6.1 (mixed graph realization problem)**.** *Given is a finite sequence* $S := \begin{pmatrix} a_1 \\ b_1 \\ c_1 \end{pmatrix}, \dots, \begin{pmatrix} a_n \\ b_n \\ c_n \end{pmatrix}$ *with* $a_i, b_i, c_i \in \mathbb{Z}_0^+$. *Does there exist a mixed graph* $G = (V, M)$ *with the labeled vertex set* $V := \{v_1, \dots, v_n\}$ *such that we have indegree* $d_G^-(v_i) = a_i$, *outdegree* $d_G^+(v_i) = b_i$ *and degree* $d_G(v) = c_i$ *for all* $v_i \in V$*?*

Note that a relaxation of the types of mixed graphs – such that between two vertices undirected edges as well as directed arcs are allowed at the same time – leads to a simple solution, because one could apply one after another a graph realization algorithm and then a digraph realization algorithm. Furthermore, the uniform sampling problem for the mixed graph problem is a very interesting open problem.

The connection to algebraic graph theory. We consider the set $M(G)$ of all arcs sets A' of subdigraphs $G' = (V, A')$ (including the empty set) from a given digraph $G = (V, A)$. It is well known that the sum operation $\Delta : M(G) \times M(G) \mapsto M(G)$ with $A_1 \Delta A_2 := (A_1 \backslash A_2) \cup (A_2 \backslash A_1)$ together with a scalar multiplication $\cdot : GF(2) \times M(G) \mapsto M(G)$ with $0 \cdot A' = \emptyset$ and $1 \cdot A' = A'$ for all $A' \in M(G)$ form an $|A|$-dimensional vector space $(M(G), \Delta, \cdot)$ over $GF(2)$ see for example (Die05). The subset $C(G)$ of $M(G)$ corresponding to all underlying undirected cycles of G together with the sum operation Δ and $\cdot$ is an $(|A| - |V| - 1)$-dimensional vector subspace of $(M(G), \Delta, \cdot)$ when G is weakly connected. Clearly, the set of all directed cycles in G form a subset of $C(G)$. Hence, all arbitrary linear combinations of these directed cycles form a vector subspace, "the directed cycle space" of $(C(G), \Delta, \cdot)$. Hence, our dag realization problem is the question of finding a digraph realization with a "directed cycle space" of dimension zero. We believe a consideration of basic questions with respect to this vector subspace could lead to more insights for our problem. Moreover, it is an elementary problem in algebraic graph theory.

References

[AZ04] M. Aigner and G. M. Ziegler, *Proofs from the book*, Springer, 2004. vii

[Bat04] V. Batagelj, *Pajek datasets: Food webs*, `vlado.fmf.uni-lj.si/pub/networks/data/bio/foodweb/foodweb.htm`, 2004. 53

[BBV07] I. Bezáková, N. Bhatnagar, and E. Vigoda, *Sampling binary contingency tables with a greedy start*, Random Structures and Algorithms **30** (2007), 168–205. 5, 78

[BC78] E. A. Bender and E. R. Canfield, *The asymptotic number of labeled graphs with given degree sequences*, Journal of Combinatorial Theory, Series A **24** (1978), no. 3, 296–307. 78, 79

[BKS10] M. Bayati, J. H. Kim, and A. Saberi, *A sequential algorithm for generating random graphs*, Algorithmica **58** (2010), no. 4, 860–910. 79

[BM10] A. Berger and M. Müller-Hannemann, *Uniform sampling of digraphs with a fixed degree sequence*, Proceedings of the 36th International Conference on Graph-Theoretic Concepts in Computer Science (Berlin, Heidelberg), WG'10, Lecture Notes in Computer Science, vol. 6410, pp. 220–231, Springer, 2010, full version available as Preprint in Arxiv:0912.0685v3. 81

[BM11a] A. Berger and M. Müller-Hannemann, *Dag characterizations of directed degree sequences*, Tech. Report 2011/6, Martin-Luther-Universität Halle-Wittenberg, Department of Computer Science, 2011. 25

[BM11b] ———, *Dag realisations of directed degree sequences*, FCT 2011, LNCS, vol. 6914, pp. 264–275, Springer, Heidelberg, 2011, full version available as Technical Report 2011/5, Martin-Luther-Universität Halle-Wittenberg, Department of Computer Science. 25

[Bol80] B. Bollabas, *A probabilistic proof of an asymptotic formula for the number of labelled regular graphs*, European J. Combin. **1** (1980), no. 4, 311 – 316. 78

[CDG07] C. Cooper, M. Dyer, and C. Greenhill, *Sampling regular graphs and a peer-to-peer network*, Combinatorics, Probability and Computing **16** (2007), 557–593. 78

[Che66] W. Chen, *On the realization of a (p,s)-digraph with prescribed degrees*, Journal of the Franklin Institute **281** (1966), no. 5, 406 – 422. 1, 3, 18, 23

[Die05] R. Diestel, *Graph theory (graduate texts in mathematics)*, Springer, August 2005. 110

[DM81] P. Diaconis and S. Mehrdad, *Generating a random permutation with random transpositions*, Probability Theory and Related Fields **57** (1981), 159–179. 104

[Dur64] Richard Durstenfeld, *Algorithm 235: Random permutation*, Commun. ACM **7** (1964), 420. 54

[EG60] P. Erdős and T. Gallai, *Graphs with prescribed degree of vertices (Hungarian)*, Mat. Lapok **11** (1960), 264–274. 1

[EMS10] P. L. Erdős, I. Miklós, and L. Soukup, *Towards random uniform sampling of bipartite graphs with given degree sequence*, arXiv:1004.2612v3, 2010. 78

[EMT09] P. L. Erdős, I. Miklós, and Z. Toroczkai, *A simple Havel-Hakimi type algorithm to realize graphical degree sequences of directed graphs*, arXiv:0905.4913v1, 2009. 80

[Fer] `http://www-history.mcs.st-andrews.ac.uk/Biographies/Ferrers.html`, visited september 2011. 19

[FF56] L.R. Jr. Ford and D.R. Fulkerson, *On representatives of subsets*, Canad. J. Math. **8** (1956), 399–404. 19

[Ful60] D.R. Fulkerson, *Zero-one matrices with zero trace*, Pacific J. Math. **10** (1960), 831–836. 1, 3, 18, 22

[Gal57] D. Gale, *A theorem on flows in networks*, Pacific J.Math. **7** (1957), 1073–1082. 1, 3, 18, 21

[Hak62] ———, *On the realizability of a set of integers as degrees of the vertices of a simple graph*, J. SIAM Appl. Math. **10** (1962), 496–506. 1, 15, 28

[Hal35] P. Hall, *On representatives of subsets*, J. London Math. Soc. **10** (1935), 26–30. 18

[Hav55] V. Havel, *A remark on the existence of finite graphs*, Casopis Pest. Math. **80** (1955), 477–480. 1, 15, 28

[HIS81] P. L. Hammer, T. Ibaraki, and B. Simeone, *Threshold sequences*, SIAM Journal on Algebraic and Discrete Methods **2** (1981), no. 1, 39–49. 108

[JS96] M. Jerrum and A. Sinclair, *The Markov chain Monte Carlo method: An approach to approximate counting and integration*, Approximation Algorithms for NP-hard Problems (D.S. Hochbaum, ed.), PWS Publishing, Boston, 1996, pp. 482–520. 75

[JSV04] M. Jerrum, A. Sinclair, and E. Vigoda, *A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries*, Journal of the ACM **51** (2004), 671–697. 5, 55, 78

[Kim99] C. Kimberling, *The origin of the Ferrers graphs*, The Mathematical Gazette 83 (497) **2** (1999), 194–198. 19

[KLP$^+$05] D. Koschützki, K. A. Lehmann, L. Peeters, S. Richter, D. Tenfelde-Podehl, and O. Zlotowski, *Centrality indices*, Network Analysis: Methodological Foundations (U. Brandes and T. Erlebach, eds.), Lecture Notes in Computer Science, vol. 3418, pp. 16–61, Springer, 2005. 7, 81

[Kor73] M. Koren, *Extreme degree sequences of simple graphs*, Journal of Combinatorial Theory, Series B **15** (1973), no. 3, 213 – 224. 108

[KTV99] R. Kannan, P. Tetali, and S. Vempala, *Simple Markov-chain algorithms for generating bipartite graphs and tournaments*, Random Structures and Algorithms **14** (1999), 293–308. 77

[Kun73] S. Kundu, *The k-factor conjecture is true*, Discrete Mathematics **6** (1973), 367–376. 16

[KV03] J.H. Kim and V.H. Vu, *Generating random regular graphs*, STOC 2003, 2003, pp. 213–222. 79

[KW73] D. J. Kleitman and D. L. Wang, *Algorithms for constructing graphs and digraphs with given valences and factors*, Discrete Mathematics **6** (1973), no. 1, 79 – 88. 1, 3, 15, 16, 17

[LaM09] M. D. LaMar, *Algorithms for realizing degree sequences for directed graphs*, arXiv.org:0906.0343v1 (2009), no. 1. 17, 80, 93

[Lov96] L. Lovász, *Random walks on graphs: A survey*, Combinatorics, Paul Erdős is Eighty (D. Miklós et al., ed.), vol. 2, János Bolyai Mathematical Society, 1996, pp. 353–397. 75, 76, 102

[LPW06] D. A. Levin, Y. Peres, and E. L. Wilmer, *Markov chains and mixing times*, American Mathematical Society, 2006. 104

[MKI$^+$04] R. Milo, N. Kashtan, S. Itzkovitz, M.E.J. Newman, and U. Alon, *On the uniform generation of random graphs with arbitrary degree sequences*, arXiv:cond-mat/0312028v2, 30 May 2004, 2004. 80

[MP95] N. V. R. Mahadev and U. N. Peled, *Threshold graphs and related topics*, North-Holland, 1995. 21, 108

[MSOI$^+$02] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, *Network motifs: simple building blocks of complex networks*, Science **298** (2002), 824–827. 7, 81

[MW90] B. McKay and N. C. Wormald, *Uniform generation of random regular graphs of moderate degree*, J. Algorithms **11** (1990), 52–67. 79

[MW91] ______, *Asymptotic enumeration by degree sequence of graphs with degrees* $o(n^{1/2})$, Combinatorica **11** (1991), 369–382. 79

[Nic11] A. Nichterlein, *Realizing Degree Sequences for Directed Acyclic Graphs is Hard*, arXiv:1110.1510, 2011. iv, 2, 25, 107

[PA94] U. N. Peled and S. Arikati, *Degree sequences and majorization*, no. 199, 179–212. 109

[RJB96] A.R. Rao, R. Jana, and S. Bandyopadhyay, *A Markov chain Monte Carlo method for generating random (0,1)–matrices with given marginals*, Sankhya: The Indian Journal of Statistics **58** (1996), 225–242. 6, 80

[Rys57] H.J. Ryser, *Combinatorial properties of matrices of zeros and ones*, Canad J.Math. **9** (1957), 371–377. 1, 3, 18, 21, 77

[Sch03] A. Schrijver, *Combinatorial optimization: Polyhedra and efficiency*, Springer, 2003. 93

[SF82] J. J. Sylvester and F. Franklin, *A constructive theory of partitions, arranged in three acts, an interact and an exodion*, American Journal of Mathematics **5** (1882), no. 1, pp. 251–330. 19

[Sin92] A. Sinclair, *Improved bounds for mixing rates of Markov chains and multi-commodity flow*, Combinatorics, Probability & Computing **1** (1992), 351–370. 75, 108

[Sin93] ———, *Algorithms for random generation and counting: A Markov chain approach*, Birkhäuser, 1993. 75

[SW99] A. Steger and N. Wormald, *Generating random regular graphs quickly*, Combinatorics, Probability, and Computing **8** (1999), 377–396. 79

[Tut52] W.T. Tutte, *The factors of graphs*, Canadian J. of Mathematics **4** (1952), 314–328. 5

[Tut54] ———, *A short proof of the factors theorem for finite graphs*, Canadian J. Of Mathematic **6** (1954), 347–352. 25, 78

[Tut81] W. T. Tutte, *Graph factors*, Combinatorica **1** (1981), no. 1, 79–97. 3

[Val79] L. G. Valiant, *The complexity of computing the permanent*, Theoretical Computer Science **8** (1979), no. 2, 189 – 201. 5, 55

[Ven13] J. Venn, *Norman Macleod Ferrers*, The Dictionary of National Biography Second Supplement **2** (1913). 19

[VrAM04] J. Venn and rev. A. McConnell, *Biography of Norman Macleod Ferrers*, Dictionary of National Biography, Oxford University Press, Oxford, 2004. 19

Declaration

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides Statt, dass ich diese Arbeit selbständig und ohne fremde Hilfe verfasst, andere als die von mir angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Werken wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Um einen Doktorgrad habe ich mich bisher nicht beworben.

Halle/Saale, den 10.10.2011

Annabell Berger