k-Connectedness and k-Factors in the Semi-Random Graph Process

by

Hidde Koerts

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Mathematics
in
Combinatorics and Optimization

Waterloo, Ontario, Canada, 2022

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

The semi-random graph process is a single-player graph game where the player is initially presented an edgeless graph with n vertices. In each round, the player is offered a vertex u uniformly at random and subsequently chooses a second vertex v deterministically according to some strategy, and adds edge uv to the graph. The objective for the player is then to ensure that the graph fulfils some specified property as fast as possible.

We investigate the properties of being k-connected and containing a k-factor. We settle the open case for 2-connectedness by showing that the player has a strategy to construct a 2-connected graph asymptotically almost surely in $(\ln 2+\ln(\ln 2+1)+o(1))n$ rounds, which matches a known lower bound asymptotically. We also provide a strategy for building a k-factor asymptotically almost surely in $(\beta + 10^{-5})n$ rounds, where β is derived from the solution of a system of differential equations.

Additionally, we consider a variant that was recently suggested by Wormald where the player chooses the first vertex and the second vertex is chosen uniformly at random. We show that the bounds for k-connectedness for the traditional setting are also tight for this variant.

Acknowledgements

First of all, I would like to thank my supervisor, Jane Gao, for all her guidance and support throughout my master's studies. I am grateful to her for opening my eyes to the world of random graph theory and helping me take small steps into it.

Next, I would like to thank my readers, Luke Postle and Sophie Spirkl, not only for their valuable feedback, but also especially for stimulating my enthusiasm for structural graph theory during the program and shaping my view of graph theory as a whole.

Finally, I would like to thank all my fellow students and friends for making Waterloo such a welcoming home and an overall amazing experience.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The semi-random graph process is a single player game introduced by Ben-Eliezer, Hefetz, Kronenberg, Parczyk, Shikhelman and Stojaković [4]. In this game, a graph is iteratively constructed from an empty graph on n vertices. Every round, one edge is added to the graph. The first endpoint of the edge, u, is chosen uniformly at random (u.a.r.) from all vertices in the graph. Subsequently, the second endpoint v is chosen deterministically by the player in accordance with their strategy.

The semi-random graph process is part of a larger category of semi-random processes where a player has limited deterministic power in an otherwise random process. This category of combinatorial random processes traces its origins to the work of Azar, Broder, Karlin and Upfal [1] on placing n balls into n bins. They showed that if the player can choose from two u.a.r. selected bins rather than just one, there exists a strategy to decrease the expected number of balls in the fullest bin by an exponential factor. Similar load-balancing schemes have been investigated by Mitzenmacher [16].

The first semi-random process on graphs was suggested by Dimitris Achlioptas during a Fields Institute workshop. Instead of adding a single edge picked u.a.r. every round like in the classical Erdős-Rényi random graph model [7], he suggested that every round the player is offered two such edges to choose from. Random graph processes where each round the player is presented with m edges to choose from according to some strategy are now known as Achlioptas graph processes. These processes were first investigated by Bohman and Frieze [5], who showed that allowing the player to choose from two edges enables the player to delay the formation of a giant connected component.

In the seminal paper on the semi-random graph process, Ben-Eliezer, Hefetz, Kronenberg, Parczyk, Shikhelman and Stojaković [4] provide asymptotic upper and lower bounds on the number of rounds needed to achieve certain objectives with high probability, including minimum degree, clique number, and k-connectedness. Additionally, they show how the semi-random graph process can be used to model other random graph models. Specifically, they show the ability to couple the process to the Erdős-Rényi random graph model, the k-out model, and the min-degree process.

Subsequent work by Gao, MacRury and Prałat [11] provides further bounds on the minimum number of rounds needed to construct a perfect matching. Similarly, Gao, Kamiński, MacRury and Prałat [10] show bounds on the number of rounds required to construct a Hamiltonian cycle.

Further research by Behague, Marbach, Prałat and Rucinski [2] gives tight asymptotic bounds for the minimum number of rounds needed to construct a subgraph isomorphic to a fixed graph H. They moreover generalise the semi-random graph process to hypergraphs.

In this thesis we consider two graph properties in the semi-random graph process. In Chapter 5 we show a tight asymptotic upper bound for the number of rounds needed to ensure the graph becomes 2-connected. Combined with the results of Ben-Eliezer, Hefetz, Kronenberg, Parczyk, Shikhelman and Stojaković [4], this settles the number of rounds needed to ensure k-connectivity for all $k \in \mathbb{N}$.

In Chapter 6 we extend the work of Gao, MacRury and Prałat [11] on perfect matchings to the property of containing a k-factor for all fixed $k \geq 2$. We introduce and analyse a strategy that gives an asymptotic upper bound on the number of rounds needed to ensure the existence of a k-factor for all $k \geq 2$.

Additionally, we consider a modification of the semi-random graph process suggested by Wormald. In this variant, which we introduce in Chapter 3, the strategy for choosing a vertex each round cannot depend on the vertex randomly selected that same round.

Chapter 2

Notation

In this section we will more formally define the semi-random graph process and the notation we will use in this thesis.

For a graph G, we denote its vertex and edge sets by $V(G)$ and $E(G)$ respectively. We denote the degree of a vertex $v \in V(G)$ in graph G by $\deg_G(v)$. We use $\delta(G)$ and $\Delta(G)$ to denote the minimum and maximum degrees of a graph respectively. For a set $S \subseteq V(G)$ of vertices, we use $G[S]$ for the subgraph induced by set S in graph G. The open and closed neighbourhoods of a vertex $v \in V(G)$ in graph G will be denoted by $N_G(v)$ and $N_G[v]$ respectively.

The semi-random graph process is a single-player game in which a multi-graph is iteratively constructed in a sequence of rounds. Because all the graph properties we consider are invariant under adding multi-edges and loops, we generally consider the underlying simple graph. Notably, we define the degree of a vertex in the multi-graph to be the number of distinct neighbours, not including itself. That is, $\deg_G(v) = |N_G(v) \setminus \{v\}|$ for each vertex $v \in V(G)$. Moreover, we will use the previously introduced notation for simple graphs for the graphs generated by the process as well.

In each round in the semi-random graph process, a single edge is added to the graph. We will denote the graph obtained after ℓ rounds by G_ℓ. The initial graph, G_0, is an empty graph with vertex set $\{1, 2, \ldots, n\}$. We denote the set $\{1, 2, \ldots, k\}$ by $[k]$, and thus $V(G_0) = [n]$. In the t^{th} round, we construct graph G_t from graph G_{t-1} as follows. Let u_t be a vertex picked u.a.r. in graph G_{t-1}. We say that vertex u_t is hit in round t. We then choose a vertex $v_t \in [n]$ according to some strategy, and add edge $u_t v_t$ to graph G_{t-1} to obtain graph G_t. Note that if $u_t = v_t$ the new edge is a loop, and if G_{t-1} already contained $u_t v_t$ the new edge is a multi-edge. Thus, $V(G_t) = V(G_{t-1}) = [n]$, and $E(G_t) = E(G_{t-1}) \cup \{u_t v_t\}$. Additionally, we refer to u_t as a square, and v_t as a circle in round t, as introduced by Gao, MacRury and Prałat [11]. Each edge in graph G_t then connects a square and a circle in the round that it is added.

We denote a graph G having property $\mathcal{P}$ by $G \in \mathcal{P}$. We say that a graph property $\mathcal{P}$ is *monotone* if for each graph $G \in \mathcal{P}$ and each potential edge $e \in V(G) \times V(G)$ it holds that

the graph G' given by $V(G') = V(G)$ and $E(G') = E(G) \cup \{e\}$ also has property $\mathcal{P}$. Note that by this definition, if $G_t \in \mathcal{P}$ for some $t > 0$ and a monotone graph property $\mathcal{P}$, it follows that $G_{t'} \in \mathcal{P}$ for all $t' \geq t$ as well.

Moreover, we say an event $A = A_n$ occurs asymptotically almost surely (a.a.s.) in G_t if $\mathbb{P}(A_n) \to 1$ as $n \to \infty$. The aim of the game is then to a.a.s. force graph G_t to have specified monotone graph properties in as few rounds as possible. We refer to this monotone graph property as the *objective property*. We are specifically interested in the asymptotic value of the minimum number of rounds required to ensure G_t has the objective property when taking n to infinity. Let the *history* at time t, denoted by $\mathcal{H}_t$, be a record of all events in the process up to round t (that is, $u_1, v_1, u_2, v_2, \ldots, u_{t-1}, v_{t-1}$). A *strategy* $\mathcal{S}$ is then a sequence of functions $f_1, f_2, \ldots$ such that f_t takes history $\mathcal{H}_t$ and vertex u_t as input, and gives a probability distribution over $[n]$. Vertex v_t is then chosen according to this probability distribution in round t.

For an objective property $\mathcal{P}$, a strategy $\mathcal{S}$, and a real number $0 < q < 1$, let $\tau_{\mathcal{P}}(\mathcal{S}, q, n)$ be the minimum value $t \geq 0$ such that $\mathbb{P}[G_t \in \mathcal{P}] \geq q$; recalling that n is the number of vertices in G_t. If no such value t exists, we say that $\tau_{\mathcal{P}}(\mathcal{S}, q, n) = \infty$. Let $\tau_{\mathcal{P}}(q, n)$ denote the minimum value of $\tau_{\mathcal{P}}(\mathcal{S}, q, n)$ over all possible strategies $\mathcal{S}$. We are interested in the asymptotic value of $\tau_{\mathcal{P}}(q, n)$ when probability q approaches 1. Therefore, we define

$$\tau_{\mathcal{P}} := \lim_{q \uparrow 1} \limsup_{n \to \infty} \frac{\tau_{\mathcal{P}}(q, n)}{n},$$

where the limit superior is used to avoid potential issues related to parity and divisibility conditions.

We note that upper bounds on $\tau_{\mathcal{P}}$ can be obtained by considering explicit strategies. In this thesis, we will design and analyse various strategies to provide bounds for several properties $\mathcal{P}$. In this thesis, all the considered objective properties are monotonically increasing. It is therefore sufficient to construct a graph G_t which contains a spanning subgraph that has property $\mathcal{P}$. In some rounds, given certain histories and vertex u_t, we may choose vertex v_t arbitrarily and not use the edge $u_t v_t$ for the construction of a spanning subgraph of G_t that has property $\mathcal{P}$. We will consider such a round a *failure round*. Allowing failure rounds in some cases leads to algorithms that are easier to analyse.

Unless specified otherwise, all asymptotic notation relates to n, i.e. $o(1)$ implies a function that tends to 0 as $n \to \infty$.

Chapter 3

Pre- and post-positional models

Recently, Nick Wormald proposed (via personal contact) an alternative version of the semi-random graph process. Instead of the first vertex being chosen u.a.r. in each round and the second vertex being chosen according to some strategy, he proposed switching this order. That is, the first vertex in each round is chosen deterministically, and the second vertex is chosen u.a.r. We refer to this new model as the *pre-positional semi-random graph process*, and the original model by Ben-Eliezer, Hefetz, Kronenberg, Parczyk, Shikhelman and Stojaković [4] as the *post-positional semi-random graph process*.

For the pre-positional process we use notation analogous to the post-positional process. A strategy $\mathcal{S}$ for the pre-positional process is a sequence of functions $f_1, f_2, \ldots$, where each function f_t depends only on history $\mathcal{H}_t$ to give a probability distribution over $[n]$. For a property $\mathcal{P}$, a strategy $\mathcal{S}$, and probability q, we then define $\tau'_{\mathcal{P}}(\mathcal{S}, q, n)$, $\tau'_{\mathcal{P}}(q, n)$, and $\tau'_{\mathcal{P}}$ for the pre-positional process analogous to $\tau_{\mathcal{P}}(\mathcal{S}, q, n)$, $\tau_{\mathcal{P}}(q, n)$, and $\tau_{\mathcal{P}}$ in the post-positional process respectively.

Lemma 3.1. *For any graph property $\mathcal{P}$, $\tau_{\mathcal{P}} \leq \tau'_{\mathcal{P}}$.*

Proof. Let $\mathcal{S}$ be an optimal strategy for property $\mathcal{P}$ in the pre-positional process for fixed n and probability q such that $\tau'_{\mathcal{P}}(\mathcal{S}, q, n) = \tau'_{\mathcal{P}}(q, n)$. We note that $\mathcal{S}$ is also a valid strategy for the post-positional process, where the potential dependence on vertex u_t in round t is not used. Hence, $\tau_{\mathcal{P}}(\mathcal{S}, q, n) \geq \tau_{\mathcal{P}}(q, n)$.

We moreover observe that for strategy $\mathcal{S}$ not depending on vertex u_t in round t, the probability distribution over all possible edges in round t is equal between the two models. As a result, $\tau_{\mathcal{P}}(\mathcal{S}, q, n) = \tau'_{\mathcal{P}}(\mathcal{S}, q, n)$. Thus,

$$\tau_{\mathcal{P}}(q, n) \leq \tau_{\mathcal{P}}(\mathcal{S}, q, n) = \tau'_{\mathcal{P}}(\mathcal{S}, q, n) = \tau'_{\mathcal{P}}(q, n).$$

Then, as $\tau_{\mathcal{P}}(q, n) \leq \tau'_{\mathcal{P}}(q, n)$ for all $0 < q < 1$ and $n \geq 1$, we find

$$\tau_{\mathcal{P}} = \lim_{q \uparrow 1} \limsup_{n \to \infty} \frac{\tau_{\mathcal{P}}(q, n)}{n} \leq \lim_{q \uparrow 1} \limsup_{n \to \infty} \frac{\tau'_{\mathcal{P}}(q, n)}{n} = \tau'_{\mathcal{P}},$$

as desired. □

Moreover, by similar arguments, if $\mathcal{S}$ is an (asymptotically) optimal strategy in the post-positional process, and $\mathcal{S}$ does not depend on vertex u_t in round t for all $t \geq 0$, it follows that $\tau_{\mathcal{P}} = \tau'_{\mathcal{P}}$.

The following question was asked by Wormald.

Question 3.1. For what types of graph properties $\mathcal{P}$ does the pre-positional process take more rounds asymptotically than the post-positional process to obtain a graph on $[n]$ in $\mathcal{P}$?

Note that there exist properties $\mathcal{P}$ such that asymptotically the post-positional process strictly requires fewer rounds than the pre-positional process. Consider for instance the following property. Let $G_1, G_2, \ldots$ be a family of graphs on $[1], [2], \ldots$ respectively. Let $\mathcal{P}$ be the property of a graph on n vertices containing at least one of the edges in $E(G_n)$. If we choose the family of graphs such that $\delta(G_i) \geq 1$ for all $i \geq 2$, we can always select v_t to be a neighbour of u_t in G_n in the post-positional model. As such, there exists a strategy for the post-positional model that ensures we obtain a graph in $\mathcal{P}$ in exactly 1 round. On the other hand, we observe that in the pre-positional model the probability of successfully adding an edge contained in $E(G_n)$ is at most $\Delta(G_n)/n$. Thus, the expected number of rounds to obtain a graph in $\mathcal{P}$ is at least $n/\Delta(G_n)$. Specifically, for $\Delta(G_i) \leq k$ for all $i \geq 2$ for some constant $k \geq 1$, the required number of rounds to obtain a graph in $\mathcal{P}$ is exactly 1 in the post-positional model, and asymptotically at least linear in n in the pre-positional model.

In this thesis, we focus on the post-positional process. For k-connectivity, we will additionally provide tight results for the pre-positional process.

Unless stated otherwise, all results refer to the post-positional model.

Chapter 4

Tools

4.1 Concentration inequalities

To analyse the asymptotic behaviour of our algorithms, we make use of concentration inequalities. Notably, we extensively use Chernoff-Hoeffding bounds for binomial random variables. For the reader's convenience, we here include the concentration inequalities we use most frequently.

Remark 4.1 (Dubhashi and Panconesi, [6, Theorem 1.1])**.** *Let X be a binomial random variable with success probability p and n trials. Then the following inequalities hold.*

- *For all $t > 0$,*

$$\mathbb{P}\left(X > \mathbb{E}[X] + t\right) \leq \exp\left(-\frac{2t^2}{n}\right),$$
$$\mathbb{P}\left(X < \mathbb{E}[X] - t\right) \leq \exp\left(-\frac{2t^2}{n}\right).$$

- *For all $\epsilon > 0$,*

$$\mathbb{P}\left(X > (1+\epsilon) \cdot \mathbb{E}[X]\right) \leq \exp\left(-\frac{\epsilon^2}{3} \cdot \mathbb{E}[X]\right),$$
$$\mathbb{P}\left(X < (1-\epsilon) \cdot \mathbb{E}[X]\right) \leq \exp\left(-\frac{\epsilon^2}{2} \cdot \mathbb{E}[X]\right).$$

- *For all $t > 2e \cdot \mathbb{E}[X]$,*

$$\mathbb{P}\left(X > t\right) \leq 2^{-t}.$$

4.2 Wormald's differential equation method

First introduced by Wormald [18], the differential equation method is a tool to describe the trajectories of a set of random variables in a discrete-time random process using a system of differential equations. For such a process, it uses a system of differential equations where the derivatives are based on the expected change in the random variables in a single time-step. Wormald showed that under certain conditions the trajectories of the random variables converge to the solution of the system of differential equations a.a.s. The system of differential equations is generally easier to solve or approximate, enabling the analysis of complex random processes.

Wormald's differential equation method has been successfully applied to obtain bounds on a number of objectives in the semi-random graph process, including minimum degree (Ben-Eliezer, Hefetz, Kronenberg, Parczyk, Shikhelman and Stojaković, [4]), containing a perfect matching (Gao, MacRury and Prałat, [11]), and containing a Hamiltonian cycle (Frieze and Sorkin, [8]; Gao, Kamiński, MacRury and Prałat, [10]; Gao, MacRury and Prałat, [12]).

One of the theorem formulations of Wormald's differential equation method is included in Appendix A for the reader's convenience.

4.3 Min-degree process

The min-degree process is a variant on the classical random graph process and was introduced and first studied by Wormald [19]. In the min-degree process, G_0 is an edgeless graph on $[n]$. Given G_t, choose a vertex u of minimum degree in G_t u.a.r., and subsequently choose a vertex v not adjacent to vertex u in graph G_t u.a.r. Graph G_{t+1} is then constructed by adding edge uv to graph G_t. Wormald used his differential equation method described in Section 4.2 to prove that the number of rounds the graph process takes to ensure that graph G_t has minimum degree k for $k \geq 1$ is a.a.s. $c_k n$ for some specific constant c_k. We denote the graph property of having minimum degree k by $\mathcal{D}_k$.

Ben-Eliezer, Hefetz, Kronenberg, Parczyk, Shikhelman and Stojaković [4] have since studied adapted versions of the min-degree process as modelled by the semi-random graph process. By choosing v_t u.a.r. from all vertices of minimum degree not adjacent to u_t in graph G_t, the resulting semi-random graph process is contiguous to the min-degree process. That is, asymptotically the two processes are equivalent. We refer to this strategy as $\mathcal{S}_{\min}$. Ben-Eliezer, Hefetz, Kronenberg, Parczyk, Shikhelman and Stojaković additionally considered strategies without the restrictions on v_t and u_t to be non-adjacent in G_t and u_t and v_t to be distinct. They showed that each of these strategies are optimal in ensuring graph G_t having minimum degree k in as few rounds as possible when taking n to infinity, and each asymptotically require $c_k n$ rounds. Each of these strategies thus obtains a graph in $\mathcal{D}_k$ in asymptotically the same number of rounds as the min-degree process.

We first provide a formal definition of strategy $\mathcal{S}_{\min}$. For each round t, distribution function

f_t is defined as follows. Let $Y_{t,\min} = \{v \in [n] \mid \deg_{G_{t-1}}(v) = \delta(G_{t-1})\}$. Then, for u_t chosen u.a.r. from $[n]$, if $Y_{t,\min} \setminus N_{G_{t-1}}[u_t] = \emptyset$, the round is considered a failure round. Otherwise, vertex v_t is chosen u.a.r. from $Y_{t,\min} \setminus N_{G_{t-1}}[u_t]$. By this formulation, strategy $\mathcal{S}_{\min}$ does not create loops nor multi-edges.

Let $G_{\min}(n, m)$ be the graph on n vertices with m edges generated by the min-degree process. To show that strategy $\mathcal{S}_{\min}$ can be used to model the min-degree process for $m = o(n^2)$ with a.a.s. $o(m)$ failure rounds, Ben-Eliezer, Hefetz, Kronenberg, Parczyk, Shikhelman and Stojaković [4] look at an auxiliary strategy where v_t is chosen u.a.r. from all minimum degree vertices. This strategy thus does not take the neighbourhood of u_t into account. They then show that the number of multi-edges and loops is asymptotically bounded by $o(m)$, which directly bounds the failure rounds of strategy $\mathcal{S}_{\min}$ as well. We note that this auxiliary strategy is also valid in the pre-positional process. Hence, the pre-positional process can also model the min-degree process with a.a.s. $o(m)$ failure rounds.

In this thesis, we will only consider the min-degree process modelled by the semi-random graph process which avoids loops and multi-edges, as given by strategy $\mathcal{S}_{\min}$. Additionally, we will use *k-min process* to refer to using the strategy $\mathcal{S}_{\min}$ until all vertices in graph G_t have degree at least k.

Chapter 5

k-Connectivity

A connected graph G is said to be k-connected if it remains connected when removing fewer than k vertices. In their seminal paper, Ben-Eliezer, Hefetz, Kronenberg, Parczyk, Shikhelman and Stojaković [4] provide tight asymptotic bounds for the minimum number of rounds needed for a semi-random graph process to produce an a.a.s. k-connected graph for all $k \geq 3$. Their proof for the upper bound is based on a slightly modified variant of the k-min process tailored for multigraphs and builds on a proof by Kang, Koh, Ree and Łuczak [14]. The strategy underlying the modified process is identical to the strategy for the k-min process as long as the graph is simple, and simplifies the analysis in the semi-random graph process. The proof shows that the graph resulting from the modified k-min process is a.a.s. k-connected for all $k \geq 3$. This proof cannot be directly extended to $k < 3$, as Kang, Koh, Ree and Łuczak [14] showed that the graph resulting from the k-min process is only a.a.s. connected for $k \geq 3$.

We moreover note that the case where $k = 1$, i.e. being connected, is trivial. Namely, we observe that one can build a tree containing $m \leq n - 1$ edges in exactly m rounds. If the first vertex chosen u.a.r. in a round is contained in the tree constructed thus far, we pick a vertex not contained in the tree, and vice versa. Hence, we can build a spanning tree in $n - 1$ rounds. We moreover note that this is optimal, as spanning trees are edge-minimal 1-connected. However, the case for $k = 1$ is not as trivial in the pre-positional model. We will discuss this further in Section 5.4.

Therefore, this chapter will focus on proving tight asymptotic bounds for the final open case in the post-positional model, $k = 2$. The best bound previously known, as observed by Ben-Eliezer, Hefetz, Kronenberg, Parczyk, Shikhelman and Stojaković [4], is the tight upper bound for $k = 3$. That is, $\tau_{\mathcal{C}_2} \leq \tau_{\mathcal{C}_3}$. They also gave a lower bound, based on the 2-min process. The 2-min process aims to ensure that each vertex has degree at least two as fast as possible, a prerequisite for 2-connectedness. Using a known result by Wormald [18, 19] on the min-degree process, they showed that the 2-min process a.a.s. takes $(\ln 2 + \ln(\ln 2 + 1)) + o(1))n$ rounds to complete. Hence, $\tau_{\mathcal{C}_2} \geq \ln 2 + \ln(\ln 2 + 1)$.

In this chapter we show a novel upper bound, which asymptotically matches the known

lower bound.

Theorem 5.1. $\tau_{\mathcal{C}_2} = \ln 2 + \ln(\ln 2 + 1)$.

That is, the minimum number of rounds required for a semi-random process to build a 2-connected graph on n vertices is asymptotic to $(\ln 2 + \ln(\ln 2 + 1))n$.

Additionally, in Section 5.4, we consider property $\mathcal{C}_k$ in the pre-positional process. We show that all the proofs for the post-positional process extend to the pre-positional process, resulting in the following theorem.

Theorem 5.2. $\tau'_{\mathcal{C}_k} = \tau_{\mathcal{C}_k}$ *for all* $k \geq 1$.

5.1 Overview

For the upper bound, our approach differs significantly from the strategy used by Ben-Eliezer, Hefetz, Kronenberg, Parczyk, Shikhelman and Stojaković [4] to prove the tight upper bounds for k-connectedness for $k \geq 3$. Namely, while their approach is predominantly probabilistic, we use a more structural approach. Our strategy is based on analysing the structure of the maximal 2-connected components of the graph resulting from the 2-min process.

In the first phase, we use the 2-min process to obtain a graph in which each vertex has degree at least 2. We show that a.a.s. most of the vertices in this graph will be contained in relatively large 2-connected subgraphs. This moreover allows us to conclude that the graph contains $o(n)$ maximal 2-connected subgraphs.

In the second phase, the aim is to ensure that the graph becomes connected. We bound the number of components by the number of maximal 2-connected subgraphs, recalling that the graph has $o(n)$ such subgraphs after the first phase. As such, by adding edges between connected components, we can quickly ensure the graph becomes connected.

In the third phase, we then want to make the graph 2-connected. We achieve this by considering a tree structure on the maximal 2-connected subgraphs, and showing that by balancing this tree, we can efficiently eliminate cut-vertices.

We show that the second and third phases both take $o(n)$ steps a.a.s. Therefore, the first phase, consisting of the 2-min process, dominates the total number of rounds in the process of building a 2-connected graph on $[n]$.

In Section 5.2, we first introduce the purely structural definitions and results we will use. Section 5.3 then builds upon these structural results to analyse the random process given by our strategy.

5.2 Supporting structural results

In this section we restate the conventional definitions of blocks and block graphs (see for instance [15]).

Definition 5.1 (Block)**.** Let $B \subseteq V(G)$ be a maximal set of vertices such that for any two vertices $x, y \in B$ with $xy \notin E(G)$, in order to separate vertex x from vertex y, it is necessary to remove at least 2 vertices from G. Then B is called a block.

Note that by this definition, each block in a graph either induces a maximal 2-connected subgraph, an edge, or an isolated vertex. Moreover, when considering connected graphs on at least 2 vertices, each block thus induces a maximal 2-connected subgraph or an edge. Based on this definition, we can then decompose a graph G into such blocks.

Definition 5.2 (Block decomposition)**.** Let $\mathcal{B}(G) \subseteq \mathcal{P}(V(G))$ denote the set of all blocks of graph G. Then $\mathcal{B}(G)$ is called the block decomposition of graph G.

We observe that by the definition of blocks, for each edge $uv \in E(G)$ in a graph G there exists a unique block $B \in \mathcal{B}(G)$ such that $u, v \in B$. Moreover, by the maximality of the blocks, the block decomposition $\mathcal{B}(G)$ is unique. Note that $\mathcal{B}(G)$ is generally not a partition of $V(G)$. However, each pair of blocks shares at most one vertex, as given in the following proposition.

Proposition 5.3 (Kőnig, [15, Theorem XIV.7])**.** *Let G be a graph. Then, for each pair of blocks $B_1, B_2 \in \mathcal{B}(G)$, it holds that $|B_1 \cap B_2| \leq 1$.*

Definition 5.3 (Block graph)**.** Let G be a graph. Then let $G_{\mathcal{B}}$ be the graph defined by $V(G_{\mathcal{B}}) = \mathcal{B}(G)$ and $E(G_{\mathcal{B}}) = \{B_1B_2 \mid B_1 \cap B_2 \neq \emptyset\}$. Then graph $G_{\mathcal{B}}$ is called the block graph of graph G.

For a graph G to be 2-connected, it must hold that $\mathcal{B}(G) = \{V(G)\}$. We aim to use the blocks and their relative structure in a graph to identify moves in a semi-random process which join multiple blocks together into a single larger block. If a semi-random edge u_tv_t joins two blocks then we call the addition of such an edge an *augmentation*. A natural augmentation to consider is to join two blocks B_i and B_j where there is a path between B_i and B_j in $G_{\mathcal{B}}$. If u_t and v_t are not themselves cut-vertices, this augmentation will immediately join all blocks along the path into a single block. To that purpose, we want to consider a tree structure on the blocks.

The traditional such structure, called the block-cut tree of a graph, was originally introduced independently by Gallai [9], and Harary and Prins [13].

Definition 5.4 (Block-cut tree)**.** Let G be a connected graph, and let S be the set of cut vertices of graph G. Then, the graph T, given by $V(T) = \mathcal{B}(G) \cup S$ and $E(T) = \{vB \mid v \in S, B \in \mathcal{B}(G), v \in B\}$, is a tree and called the block-cut tree of graph G.

We consider a structure similar to the block-cut tree, based on the block graph. Instead of including the cut-vertices in the tree, we take a spanning tree on the block graph. This ensures that we only have to work with blocks, while still providing the desired tree structure. To that aim, we introduce the following definition.

Definition 5.5 (Reduced block tree)**.** Let $G_{\mathcal{B}}$ be the block graph of a connected graph G. Then, a spanning tree $T_{\mathcal{B}}$ of graph $G_{\mathcal{B}}$ is called a reduced block tree of graph G.

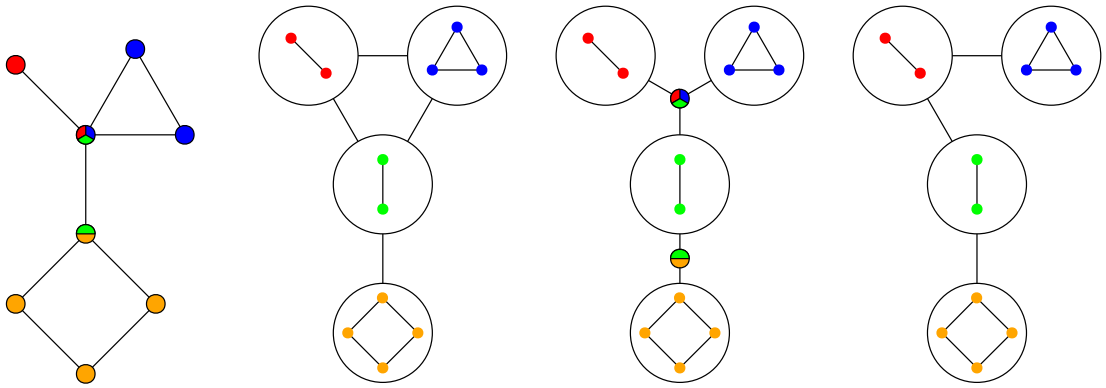

Figure 5.1: Illustration of the different structural representations of the block decomposition $\mathcal{B}(G)$ of graph G.

A reduced block tree can equivalently be constructed recursively. Let $v \in V(G)$ be a cut-vertex in a connected graph G, and let G_1 and G_2 be the induced subgraphs of graph G such that $V(G_1) \cup V(G_2) = V(G)$, $E(G_1) \cup E(G_2) = E(G)$, and $V(G_1) \cap V(G_2) = \{v\}$. We note that as vertex v is a cut-vertex, each block $B \in \mathcal{B}(G)$ is contained in either graph G_1 or graph G_2. Therefore, $\mathcal{B}(G_1) \cup \mathcal{B}(G_2) = \mathcal{B}(G)$. Let $T_{\mathcal{B}_1}$ and $T_{\mathcal{B}_2}$ be reduced block trees for graphs G_1 and G_2 respectively. Then, we can construct a reduced block tree for graph G with block decomposition $\mathcal{B}(G)$ by joining trees $T_{\mathcal{B}_1}$ and $T_{\mathcal{B}_2}$ with a single edge from a vertex in $T_{\mathcal{B}_1}$ representing a block containing vertex v to a vertex in $T_{\mathcal{B}_2}$ also representing a block containing vertex v. We observe that by Definition 5.5, the reduced block tree of a graph is generally not unique. This occurs when a vertex is contained in at least three blocks, and the block graph thus contains a clique of size at least 3.

Proposition 5.4. *Let $T_{\mathcal{B}}$ be a reduced block tree of a connected graph G. For $v \in V(G)$, the set $\{B \in V(T_{\mathcal{B}}) \mid v \in B\}$ induces a (connected) subtree in $T_{\mathcal{B}}$.*

Proof. Suppose not. Let $S \subseteq V(T_{\mathcal{B}})$ be the set of all blocks $B \in V(T_{\mathcal{B}})$ such that $v \in B$. Then the set S induces a disconnected subgraph in tree $T_{\mathcal{B}}$. Let C_1 and C_2 be two components of this induced subgraph $T_{\mathcal{B}}[S]$. Moreover, let P be a shortest path between sets $V(C_1)$ and $V(C_2)$ in $T_{\mathcal{B}}$, and let blocks $B_1, B_2 \in S$ be the endpoints of this path P such that $B_1 \in V(C_1)$ and $B_2 \in V(C_2)$. We note that P has length at least 2. Then, as P is a shortest such path, none of the internal vertices of P are contained in S. Hence, the corresponding blocks do not contain vertex v. Let G_P be the subgraph of $T_{\mathcal{B}}$ induced by the internal vertices of path P. Additionally, let $S_P \subseteq V(G)$ be the set of all vertices of graph G contained in at least one of the blocks in G_P.

We observe that by the definition of path P, subgraph G_P contains blocks adjacent to blocks B_1 and B_2, respectively, in the tree $T_{\mathcal{B}}$. Therefore, $B_1 \cap S_P, B_2 \cap S_P \neq \emptyset$. Moreover, by Proposition 5.3 we find that $B_1 \cap B_2 = \{v\}$. Therefore, as $v \notin S_P$, there exist vertices $v_1 \in B_1 \cap S_P$ and $v_2 \in B_2 \cap S_P$. Then, because blocks B_1 and B_2 are by definition connected,

there exists a $v-v_1$ path P_1 in block B_1 and a $v-v_2$ path P_2 in block B_2. Similarly, the set S_P induces a connected subgraph in G, and thus contains a v_1-v_2 path P'. We note that the union of the paths P_1, P_2 and P' gives a subgraph of G containing a cycle C containing vertex v. We note that the cycle C is 2-connected and hence is contained in a block B_C. Moreover, as this cycle contains at least 2 vertices of block B_1, by Proposition 5.3, we find that $B_1 = B_C$. Analogously, it follows that $B_2 = B_C$. However, this implies that $B_1 = B_2$, contradicting these blocks being in different connected components C_1 and C_2. By this contradiction, we conclude that the proposition holds. □

Proposition 5.5. *Let $T_{\mathcal{B}}$ be a reduced block tree of a connected graph G with $\delta(G) \geq 2$. Let $B \in \mathcal{B}$ be a block such that $B = \{u, v\}$. Then there exist distinct blocks $B_u, B_v \in \mathcal{B}$ adjacent to B in $T_{\mathcal{B}}$ such that $u \in B_u$ and $v \in B_v$.*

Proof. Because $\delta(G) \geq 2$, there exists another edge $uw \in E(G)$. Hence, as each edge is contained in a block, there exists a block $B' \in \mathcal{B}$ such that $u \in B'$ and $B' \neq B$. It then follows from Proposition 5.4 that there exists a block $B_u \in \mathcal{B}$ such that $u \in B_u$ and B_u adjacent to B in $T_{\mathcal{B}}$. Analogously, there exists a block $B_v \in \mathcal{B}$ adjacent to B in $T_{\mathcal{B}}$ such that $v \in B_v$. By the maximality of block B, it follows that $v \notin B_u$ and $u \notin B_v$. Hence, $B_u \neq B_v$, as desired. □

Corollary 5.6. *Let $T_{\mathcal{B}}$ be a reduced block tree of a connected graph G with $\delta(G) \geq 2$. Then each leaf in $T_{\mathcal{B}}$ corresponds to a 2-connected block in graph G of at least 3 vertices.*

Proof. By Proposition 5.5, blocks of size 2 cannot be leaves in $T_{\mathcal{B}}$. Then, by the definition of blocks, the result follows. □

Proposition 5.7. *Let G be a connected graph such that $|B| < n/4$ for all blocks $B \in \mathcal{B}(G)$, and let $T_{\mathcal{B}}$ be a corresponding reduced block tree. Then there exists a vertex $B^* \in V(T_{\mathcal{B}})$ and a colouring $\phi : V(T_{\mathcal{B}}) \setminus \{B^*\} \to \{red, blue\}$ such that all components of $T_{\mathcal{B}} - B^*$ are monochromatic and that for $S_{red} = \{v \in B \setminus B^* \mid B \in \mathcal{B}, \phi(B) = red\}$ and $S_{blue} = \{v \in B \setminus B^* \mid B \in \mathcal{B}, \phi(B) = blue\}$ it holds that $|S_{blue}| \leq |S_{red}| \leq 3|S_{blue}|$.*

Proof. Firstly, we note by Proposition 5.4 that $S_{\text{red}} \cap S_{\text{blue}} = \emptyset$ and hence $V(G)$ is partitioned by the sets B^*, S_{red}, and S_{blue}. Therefore, $|B^*| + |S_{\text{red}}| + |S_{\text{blue}}| = n$.

Assume that the proposition does not hold. Then, let $B^* \in V(T_{\mathcal{B}})$ and $\phi : V(T_{\mathcal{B}}) \to \{\text{red,blue}\}$ be a vertex and a colouring respectively such that all components of $T_{\mathcal{B}} - B^*$ are monochromatic, $|S_{\text{red}}| \geq |S_{\text{blue}}|$, subject to which $|S_{\text{red}}|$ is minimised. We note that as it concerns a counterexample, we must have $|S_{\text{red}}| > 3|S_{\text{blue}}|$.

We observe that as $|B^*| < n/4$, $T_{\mathcal{B}} - B^*$ is non-empty. Therefore, due to $|S_{\text{red}}| \geq |S_{\text{blue}}|$, $T_{\mathcal{B}} - B^*$ contains at least one red component. Suppose that $T_{\mathcal{B}} - B^*$ contains exactly one red component. Then, because $T_{\mathcal{B}}$ is a tree, vertex B^* has exactly one red neighbour $B' \in V(T_{\mathcal{B}})$ in $T_{\mathcal{B}}$. Then consider using vertex B' instead of vertex B^*, uncolouring B' and colouring B^* blue. Let ϕ' denote the resulting new colouring, and let S'_{red} and S'_{blue} be the

sets of vertices in G corresponding to ϕ'. We note that as blocks B^* and B' both contain less than $n/4$ vertices, it holds that $|S'_{\text{red}}| > |S_{\text{red}}| - n/4$ and $|S'_{\text{blue}}| < |S_{\text{blue}}| + n/4$. Moreover, we note that by the maximality of blocks, $B^* \setminus B', B' \setminus B^* \neq \emptyset$, and hence $|S'_{\text{red}}| < |S_{\text{red}}|$ and $|S'_{\text{blue}}| > |S_{\text{blue}}|$. If $|S'_{\text{red}}| > |S'_{\text{blue}}|$, the new colouring ϕ' is more balanced, and thus contradicts the minimality of $|S_{\text{red}}|$. Therefore, it holds that $|S'_{\text{red}}| < |S'_{\text{blue}}|$. Because we assumed that $|S_{\text{red}}| > 3|S_{\text{blue}}|$, and as $|B^*|+|S_{\text{red}}|+|S_{\text{blue}}| = n$, it follows that $|S_{\text{blue}}| \leq n/4$. Thus, $|S'_{\text{blue}}| < |S_{\text{blue}}| + n/4 \leq n/2$. But then, as $|B'| + |S'_{\text{red}}| + |S'_{\text{blue}}| = n$, it follows that

$$\begin{aligned} |S'_{\text{red}}| &= n - |S'_{\text{blue}}| - |B'| \\ &> n - \frac{n}{2} - \frac{n}{4} \\ &= \frac{n}{4}. \end{aligned}$$

Then, inverting the colours red and blue results in a colouring satisfying all the conditions of the proposition, contradicting $T_{\mathcal{B}}$ being a counterexample.

Hence, we may assume that forest $T_{\mathcal{B}} - B^*$ contains at least 2 red components. Then let $C_1, C_2, \ldots, C_\ell$ be the red components of $T_{\mathcal{B}} - B^*$, and let $S_1, S_2, \ldots, S_\ell$ be defined by $S_i = \{v \in B \setminus B^* \mid B \in C_i\}$ for $i \in [\ell]$. Then, by Proposition 5.4, the sets $S_1, S_2, \ldots, S_\ell$ partition set S_{red}.

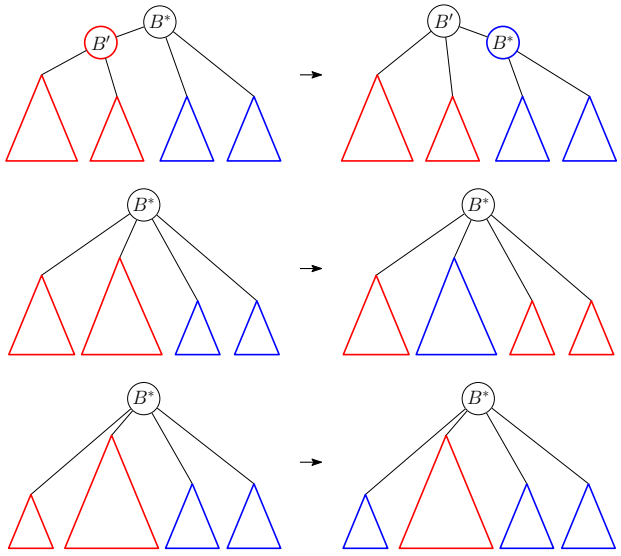

Figure 5.2: Illustration of the balancing moves used in the proof of Proposition 5.7.

Suppose that there exists an index $i \in [\ell]$ such that $|S_i| > |S_{\text{blue}}|$. Then, recolouring all blue components red, and recolouring component C_i blue leads to sets S'_{red} and S'_{blue} such that, as $\ell \geq 2$, $\min(|S'_{\text{red}}|, |S'_{\text{blue}}|) > \min(|S_{\text{red}}|, |S_{\text{blue}}|)$. Thus, as $|S'_{\text{red}}| + |S'_{\text{blue}}| = |S_{\text{red}}| + |S_{\text{blue}}|$, by possibly inverting the colours, we find a more minimal counterexample. Hence, we may assume that $|S_i| \leq |S_{\text{blue}}|$ for all $i \in [\ell]$. Then, as $|S_{\text{red}}| = \sum_{i=1}^{\ell} |S_i|$, we find that $|S_{\text{red}}| \leq \ell |S_{\text{blue}}|$. Therefore, as $|S_{\text{red}}| > 3|S_{\text{blue}}|$, it holds that $\ell > 3$.

Similarly, suppose that there exists an index $i \in [\ell]$ such that $|S_i| < (|S_{\text{red}}| - |S_{\text{blue}}|)/2$. Then clearly recolouring component C_i blue contradicts the minimality of $|S_{\text{red}}|$. Hence, we may assume that $|S_i| \geq (|S_{\text{red}}| - |S_{\text{blue}}|)/2$ for all $i \in [\ell]$. Then, as $|S_{\text{red}}| = \sum_{i=1}^{\ell} |S_i|$, we find that $|S_{\text{red}}| \geq \ell \cdot (|S_{\text{red}}| - |S_{\text{blue}}|)/2$. It then follows that, because $\ell > 3$, $|S_{\text{red}}| \leq \frac{\ell}{\ell-2}|S_{\text{blue}}|$. But then, as $\frac{\ell}{\ell-2} < 3$ for $\ell > 3$, we conclude that vertex B^* and colouring ϕ do not form a counterexample. Thus, we conclude that the proposition holds. □

5.3 Building 2-connected semi-random graphs

In this section, we describe our strategy and analyse the corresponding process for building a 2-connected semi-random graph, and obtain the tight upper bound of $\tau_{\mathcal{C}_2}$ as in Theorem 5.1. Our strategy consists of three phases.

In the first phase, we use the 2-min process. Recall from Section 4.3 that the 2-min process chooses v_t to be an isolated vertex until G_t no longer contains isolated vertices. Subsequently, it chooses v_t to be a vertex of degree 1 until $\delta(G_t) \geq 2$, at which point the process terminates. The following proposition shows useful properties of the resulting graph.

Proposition 5.8. *Let G be the semi-random graph resulting from the 2-min process. Then, a.a.s., G contains $o(n)$ vertices that are contained in 2-connected induced subgraphs of order at most $\sqrt{\ln n}$ in graph G.*

Proof. Let X be the number of vertices contained in 2-connected induced subgraphs of order at most $\sqrt{\ln n}$. We note that it suffices to show that $\mathbb{E}[X] = o(n)$. Moreover, let Y_ℓ denote the number of 2-connected induced subgraphs of order ℓ for $1 \leq \ell \leq \sqrt{\ln n}$. Thus, by linearity of expectation, $\mathbb{E}[X] \leq \sum_{\ell=1}^{\sqrt{\ln n}} \ell \mathbb{E}[Y_\ell]$.

For $1 \leq \ell \leq \sqrt{\ln n}$, let Z_ℓ denote the number of induced subgraphs of order ℓ with at least ℓ edges. Because each 2-connected graph contains at least as many edges as the vertices, it follows immediately that $Y_\ell \leq Z_\ell$, and thus, $\mathbb{E}[X] \leq \sum_{\ell=1}^{\sqrt{\ln n}} \mathbb{E}[\ell Z_\ell]$. Hence it suffices to show that $\sum_{\ell=1}^{\sqrt{\ln n}} \mathbb{E}[\ell Z_\ell] = o(n)$.

Let $1 \leq \ell \leq \sqrt{\ln n}$, and fix $S \subseteq [n]$ such that $|S| = \ell$. Let p_S be the probability that $G[S]$ contains at least ℓ edges. Note that $\mathbb{E}[Z_\ell] = \sum_{S \in \binom{[n]}{\ell}} p_S$. Next, we estimate p_S.

We first split the 2-min process into two phases. The first phase ends after the step where the last isolated vertex becomes incident with an edge, and thus the second phase starts with a graph with minimum degree one. We further split each phase into subphases for analysis. Specifically, for the first phase we define subphases $\alpha_1, \alpha_2, \ldots$ such that α_i consists of the steps where $\frac{n}{2^i} < |\{v \in V(G) \mid \deg(v) = 0\}| \leq \frac{n}{2^{i-1}}$ for $i \in \{1, 2, \ldots\}$. We note that these subphases are well defined, as by the definition of the first phase of the 2-min process, the number of isolated vertices is strictly decreasing. We then define subphases $\beta_1, \beta_2, \ldots$ of the second phase the 2-min process such that subphase β_i consists of the steps where $\frac{n}{2^i} < |\{v \in V(G) \mid \deg(v) = 1\}| \leq \frac{n}{2^{i-1}}$ for $i \in \{1, 2, \ldots\}$. Note that some of the subphases

might be empty, e.g. subphase β_1 is empty if the number of vertices with degree 1 at the beginning of the second phase is already smaller than $n/2$. We observe that there are $\log_2 n$ subphases of both phases of the 2-min process.

To bound p_S, we first choose a set T of ℓ edges from $\binom{S}{2}$. There are thus $\binom{\binom{\ell}{2}}{\ell} \leq \binom{\ell^2}{\ell}$ choices for set T. Then we determine an ordering for the edges in T. There are $\ell!$ ways to fix such an ordering. Fixing an ordering $e_1, \ldots, e_\ell$, we bound the probability that these edges are added to G in this order. The probability that a specific edge $xy \in T$ is added in a specified step in subphase α_i (and β_i) is at most $2 \cdot \frac{1}{n} \cdot \frac{2^{i-1}}{n} = \frac{2^i}{n^2}$, since the first vertex of the edge is chosen uniformly at random, and the second vertex is chosen uniformly from the isolated vertices, of which there are at most $n/2^{i-1}$. The factor 2 accounts for whether x or y is the square or the circle of the edge (note that due to the structure of the 2-min process, sometimes only one of the two may be relevant).

Let ℓ_{α_i} and ℓ_{β_i} be the number of edges of $e_1, \ldots, e_\ell$ that are added in subphases α_i and β_i respectively. Let $\boldsymbol{\ell}_\alpha = (\ell_{\alpha_i})_{i \geq 0}$ and $\boldsymbol{\ell}_\beta = (\ell_{\beta_i})_{i \geq 0}$. Note that the number of isolated vertices decreases by at least 1 in each step of the first phase of the 2-min process. Thus the number of steps in subphase α_i is at most $\frac{n}{2^{i-1}} - \frac{n}{2^i} = \frac{n}{2^i}$. Thus, given $\boldsymbol{\ell}_\alpha$ and $\boldsymbol{\ell}_\beta$, there are at most $\prod_i \binom{n/2^i}{\ell_{\alpha_i}}\binom{n/2^i}{\ell_{\beta_i}}$ ways to specify steps in the 2-min process where edges in T are added. Combining all, we have the following bound on p_S:

$$p_S \leq \binom{\ell^2}{\ell} \ell! \sum_{\boldsymbol{\ell}_\alpha, \boldsymbol{\ell}_\beta} \left(\prod_{i=1}^{\log_2 n} \binom{n/2^i}{\ell_{\alpha_i}} \binom{n/2^i}{\ell_{\beta_i}} \left(\frac{2^i}{n^2}\right)^{\ell_{\alpha_i} + \ell_{\beta_i}} \right),$$

where the first summation is over all choices for $\boldsymbol{\ell}_\alpha$ and $\boldsymbol{\ell}_\beta$ such that $\sum_{i=1}^{\log_2 n} (\ell_{\alpha_i} + \ell_{\beta_i}) = \ell$. Using $\binom{n/2^i}{\ell_{\alpha_i}} \leq (n/2^i)^{\ell_{\alpha_i}}$ and $\binom{n/2^i}{\ell_{\beta_i}} \leq (n/2^i)^{\ell_{\beta_i}}$, we then obtain

$$p_S \leq \binom{\ell^2}{\ell} \ell! n^{-\ell} \sum_{\boldsymbol{\ell}_\alpha, \boldsymbol{\ell}_\beta} 1.$$

The set of $\{(\boldsymbol{\ell}_\alpha, \boldsymbol{\ell}_\beta) \mid \sum_{i=1}^{\log_2 n} (\ell_{\alpha_i} + \ell_{\beta_i}) = \ell\}$ corresponds to the set of weak integer compositions of ℓ into $2\log_2 n$ parts of non-negative integers, and thus has cardinality $\binom{\ell + 2\log_2 n - 1}{2\log_2 n - 1} \leq \binom{\ell + 2\log_2 n}{\ell}$.

Thus, it follows that

$$\begin{aligned}
\mathbb{E}[X] &\leq \sum_{\ell=1}^{\sqrt{\ln n}} \mathbb{E}[\ell Z_\ell] \\
&= \sum_{\ell=1}^{\sqrt{\ln n}} \left(\ell \cdot \sum_{S \in \binom{[n]}{\ell}} p_S \right) \\
&\leq \sum_{\ell=1}^{\sqrt{\ln n}} \ell \binom{n}{\ell} \binom{\ell^2}{\ell} \ell! n^{-\ell} \binom{\ell + 2\log_2 n}{\ell}.
\end{aligned}$$

Using $\binom{n}{\ell} \leq n^\ell/\ell!$, $\binom{\ell^2}{\ell} \leq (e\ell)^\ell$ and $\binom{\ell+2\log_2 n}{\ell} \leq (e(\ell+\log_2 n)/\ell)^\ell \leq (10\log_2 n/\ell)^\ell$ (as $\ell \leq \sqrt{\ln n}$), we then obtain

$$\mathbb{E}[X] \leq \sum_{\ell=1}^{\sqrt{\ln n}} \ell(10e\log_2 n)^\ell = \exp\left(\sqrt{\ln n}\ln\log_2 n + O(\sqrt{\ln n})\right) = o(n),$$

as desired. □

Corollary 5.9. *Let G be the semi-random graph resulting from the 2-min process. Then, a.a.s., G contains $o(n)$ maximal 2-connected induced subgraphs.*

Proof. Consider the set $\mathcal{T} := \{(v, B) : v \in B, B \in \mathcal{B}(G)\}$. Using the block-cut tree structure (Definition 5.4), it follows that $|\mathcal{T}| \leq n + |\mathcal{B}| - 1$. Moreover, the sets $\mathcal{T}_B := \{(v', B') \in T : B' = B\}$ for $B \in \mathcal{B}(G)$ partition $\mathcal{T}$. Let $B_1, \ldots, B_\ell$ be the set of blocks of size at least $\sqrt{\ln n}$. Then, by Proposition 5.8, $\sum_{1\leq i\leq \ell} |\mathcal{T}_{B_i}| \leq |\mathcal{T}| \leq n + \ell + o(n)$. However, $|\mathcal{T}_{B_i}| \geq \sqrt{\ln n}$ for every i, and thus it follows then that $\ell\sqrt{\ln n} \leq n + \ell + o(n)$. Thus it follows that $\ell = o(n)$, as desired. □

The resulting graph thus contains $o(n)$ blocks of size at least 3. Because we have not bounded the number of blocks consisting of 2 vertices, we will use Corollary 5.6 and the other structural results in Section 5.2 to ensure the graph becomes 2-connected.

Let G_1 be the graph obtained after the first phase, i.e. the graph resulting from the 2-min process. In the second phase, we add semi-random edges to make G_1 connected. The following proposition shows that we can achieve this a.a.s. with $o(n)$ additional semi-random edges.

Proposition 5.10. *A.a.s. G_1 can be made connected by the addition of $o(n)$ semi-random edges.*

Proof. By Corollary 5.9, G_1 contains $o(n)$ maximal 2-connected induced subgraphs. We claim that each vertex not contained in a 2-connected induced subgraph is contained in a component that contains a 2-connected induced subgraph. Suppose not. Then let C be a component of graph G_1 not containing 2-connected induced subgraph. Hence, component C is acyclic, and thus a tree. However, each leaf of the tree has degree 1, contradicting the result of the 2-min process. Hence the number of components of graph G_1 is bounded from above by the number of maximal 2-connected induced subgraphs, and therefore is $o(n)$.

We moreover note that each round, in a disconnected graph, we can decrease the number of connected components by 1. Namely, for vertex u being chosen u.a.r., we select vertex v u.a.r. in a different connected component than vertex u. Hence, a.a.s., in $o(n)$ additional rounds, we can ensure graph G_1 to be connected, as desired. □

Let G_2 be the graph obtained after the second phase. In the third phase, we ensure that G_2 becomes 2-connected by adding $o(n)$ semi-random edges.

Proposition 5.11. *A.a.s. G_2 can be made 2-connected by the addition of $o(n)$ semi-random edges.*

Proof. Let $\mathcal{B}$ be the block decomposition of G_2 and $T_{\mathcal{B}}$ be a reduced block tree of G_2. By Corollary 5.6, each leaf in $T_{\mathcal{B}}$ is a 2-connected block. Thus, by Corollary 5.9, $T_{\mathcal{B}}$ a.a.s. contains $o(n)$ leaves.

First consider the case that $\mathcal{B}$ contains a block B^* such that $|B^*| \geq n/4$. We consider the following strategy. Take an arbitrary enumeration $B_1, \dots, B_h$ of all leaf blocks of $T_{\mathcal{B}}$. For each $1 \leq j \leq h$, we will add a semi-random edge between B^* and B_j in increasing order of j. Suppose these semi-random edges have already been added between B^* and B_i for all $i < j$. Let $B_j B'_1 B'_2 \dots B'_\ell B^*$ be the unique path from B_j to B^* in $T_{\mathcal{B}}$. Moreover, let x be the unique vertex in $B_j \cap B'_1$, and y the unique vertex in $B'_\ell \cap B^*$. Note that possibly $x = y$. Then, in each subsequent round t, if the square (i.e. the first vertex) u_t is contained in $B^* \setminus \{y\}$, we let v_t, the circle, be an arbitrary vertex in $B_j \setminus \{x\}$. We note that by the definitions of x and y, $u_t v_t$ cannot be an edge in G_2. If instead square u_t is not contained in $B^* \setminus \{y\}$, we consider the round a failure.

Note that in each round, the probability of the first vertex landing in $B^* \setminus \{y\}$ is $(|B^*| - 1)/n \geq 1/4 - o(1)$, and as a.a.s. $T_{\mathcal{B}}$ contains $o(n)$ leaves, the number of rounds required to add semi-random edges between B^* and all $B_1, \dots, B_h$ is $o(n)$ in expectation.

Let G'_2 be the graph resulting from the addition of the h semi-random edges as described above. Then, for each leaf block B, G'_2 contains two vertex-disjoint paths from B to B^*. Namely, one path via the blocks on the path between B and B^* in $T_{\mathcal{B}}$, and the other being the edge that was added between B and B^*. Because this holds for all leaves, using Proposition 5.5, the resulting graph is 2-edge-connected. Moreover, as each block is on a cycle with B^* and a leaf, and as the blocks of size at least 3 are 2-connected, for each cut-vertex v it follows that graph $G'_2 - v$ contains one large component containing $B^* \setminus \{v\}$, and all other components are of the form $B - v$ where $B \in \mathcal{B}$ is a block of size at least 3. We note that these blocks B such that $B - v$ is a component for some cut-vertex $v \in [n]$ correspond exactly to the blocks that are leaves in the block-cut tree (Definition 5.4), but not in $T_{\mathcal{B}}$.

By argumentation analogous to that used in the proof of Corollary 5.6, all such blocks B are 2-connected. Hence, by Proposition 5.9, there are $o(n)$ such blocks. Moreover, each such a block contains at most one cut-vertex. We then use the following strategy to absorb these cut-vertices. We iteratively consider each cut-vertex $v \in [n]$. If $u_t = v$, we consider the round a failure. Otherwise, we choose v_t in a different component than u_t in the graph $G'_2 - v$. Note that thus with probability $1 - o(1)$ we decrease the number of components by 1. Once there is a single component left, we continue on to the next cut-vertex. We note that as there are $o(n)$ such blocks, each of which contains at most one cut-vertex, the total number of rounds needed to absorb all such cut-vertices is $o(n)$ in expectation.

It thus takes at most $o(n)$ rounds in total in expectation to ensure that the graph becomes 2-connected. Standard concentration inequalities such as Chernoff bounds then immediately

imply that also a.a.s. it takes $o(n)$ rounds to extend G_2 to a 2-connected graph (indeed, if there exists a constants $c > 0$ and $p > 0$ such that the number of rounds is at least cn with probability at least p, this contradicts the final inequality in Remark 4.1).

Hence we may assume that each block B in $\mathcal{B}$ is of size strictly smaller than $n/4$. We use a different strategy in this case. Instead of adding edges from leaves to a single block, we will consider balancing the tree into two subforests. We will then add edges between the leaves in one forest and vertices in the other forest, and vice versa.

Let vertex $B^* \in V(T_\mathcal{B})$, colouring $\phi : V(T_\mathcal{B}) \setminus \{B^*\} \to \{\text{red, blue}\}$, and sets S_{red} and S_{blue} be as given by Proposition 5.7. For each $v \in B^*$ let $T_{\mathcal{B},v}$ denote the components of $T_\mathcal{B} - v$ that contain a block containing v. Thus, $T_{\mathcal{B},v}$ denotes the blocks B where v is the last cut-vertex on the path from B to B^* in $T_\mathcal{B}$. We refer to $T_{\mathcal{B},v}$ as the branch rooted at v. Moreover, let $S_{\mathcal{B},v}$ denote $\bigcup_{B \in V(T_{\mathcal{B},v})} B \setminus B^*$. That is, $S_{\mathcal{B},v}$ is the set of all vertices contained in blocks in $T_{\mathcal{B},v}$ except for vertex v itself. If $|S_{\mathcal{B},v}| \leq n/8$, we say branch $T_{\mathcal{B},v}$ is small. Otherwise we say $T_{\mathcal{B},v}$ is big. Finally, for all leaf blocks B, let v_B denote the vertex that block B has in common with the next block on the path from B to B^* in $T_\mathcal{B}$.

We first consider the leaves of $T_\mathcal{B}$ contained in small branches. Take two arbitrary enumerations $B_1, B_2, \ldots, B_{h_1}$ and $R_1, R_2, \ldots, R_{h_2}$ of all blue and red leaf blocks of $T_\mathcal{B}$ contained in small branches respectively. We will iteratively add edges between S_{red} and B_j in increasing order of j, and analogously between S_{blue} and R_j. Suppose that semi-random edges have already been added between S_{red} and B_i for all $i < j$. Let $T_{\mathcal{B},v}$ be the branch containing leaf B_j. Then, if square u_t lands in $S_{\text{red}} \setminus S_{\mathcal{B},v}$, we choose circle v_t to be an arbitrary vertex in $B_j \setminus \{v_{B_j}\}$. Because $|B_j| \geq 2$, such a choice for v_t always exists. Because u_t and v_t are in different branches, and as $u_t, v_t \notin B^*$, it follows that edge $u_t v_t$ cannot be contained in G_2.

Analogously, for R_i the red leaf in a small branch $T_{\mathcal{B},v}$ with the lowest index that has not previously received a circle, if square u_t is contained in $S_{\text{blue}} \setminus S_{\mathcal{B},v}$, we choose v_t in $R_i \setminus \{v_{R_i}\}$. Finally, if square u_t is contained in B^*, or in same branch as the considered leaf, we consider the round a failure.

Then, as tree $T_\mathcal{B}$ has $o(n)$ leaves, there are $o(n)$ blue and $o(n)$ red leaves. Moreover, by Proposition 5.7 $|S_{\text{blue}}| \leq |S_{\text{red}}| \leq 3|S_{\text{blue}}|$, and $|S_{\text{red}}| + |S_{\text{red}}| \geq 3n/4$. Thus, the probability that a vertex from $S_{\text{red}} \setminus S_{\mathcal{B},v}$ is chosen u.a.r. where $T_{\mathcal{B},v}$ a small branch, is at least $3n/8 - n/8 = n/4$. Similarly, the probability that a vertex from $S_{\text{blue}} \setminus S_{\mathcal{B},v}$ is chosen u.a.r. where $T_{\mathcal{B},v}$ a small branch, is at least $3n/16 - n/8 = n/16$. Hence, the expected number of rounds needed to add edges to all leaf blocks in small branches is $o(n)$.

Next, we consider the leaf blocks in big branches. We first note that there are at most 8 big branches. We use a similar strategy as for the small branches, but drop the requirement that u_t and v_t must be in distinct branches. Again take two arbitrary enumerations $B_1, B_2, \ldots, B_{h_3}$ and $R_1, R_2, \ldots, R_{h_4}$ of all blue and red leaf blocks of $T_\mathcal{B}$ contained in big branches respectively. Suppose that semi-random edges have already been added between S_{red} and B_i for all $i < j$. Then, if square u_t lands in S_{red}, we choose circle v_t to be an arbitrary vertex in $B_j \setminus \{v_{B_j}\}$. Because $|B_j| \geq 2$, such a choice for v_t always exists. Because

u_t and v_t are in different colour classes, and as $u_t, v_t \notin B^*$, it follows that edge $u_t v_t$ cannot be contained in G_2. If u_t lands in S_{blue}, we analogously choose v_t in the considered red leaf block. If instead u_t lands in B^*, we consider the round a failure.

Because the probability that a vertex from S_{red} is chosen u.a.r. is at least $3n/8$, and the probability that a vertex from S_{blue} is chosen u.a.r. is at least $3n/16$, it also takes $o(n)$ rounds in expectation to add edges to all leaf blocks in big branches.

After all leaves in both small and big branches have received an edge, there exist two internally vertex-disjoint paths from each leaf block B to B^*. Namely, as all of the edges we added have one red and one blue endpoint, each blue leaf has a path which only contains blue vertices and a vertex in B^*, and a path that starts with the added edge, and then only contains red vertices and one vertex in B^*. Analogously, there exist two such paths from each red leaf. As these two paths do not share their endpoint in leaf B, and as each leaf is 2-connected by Corollary 5.6, set $B \setminus \{v_B\}$ does not contain any cut-vertices.

We note that again the resulting graph is 2-edge-connected. We then use the same strategy as in the case where there exists a block of size at least $n/4$ to eliminate all the cut-vertices that separate individual blocks from the rest of the graph. Recall that this strategy a.a.s. takes $o(n)$ rounds. Let G_2'' be the resulting graph. We then observe that no vertex in $[n] \setminus B^*$ is a cut-vertex in graph G_2''. Hence, we consider a cut-vertex $v \in B^*$. First suppose that the branch rooted at v is empty. We that by Proposition 5.5 it then holds that $|B^*| \geq 3$. But then, B^* is 2-connected, contradicting v being a cut-vertex. Next suppose that the branch rooted at v is small. We note that for each vertex in $S_{\mathcal{B},v}$ there exists a path to a leaf of branch $T_{\mathcal{B},v}$ contained in $S_{\mathcal{B},v}$. As each such a leaf has an edge to another branch, and as B^* is either 2-connected or isomorphic to K_2, it follows that $G_2'' - v$ is connected. Hence, v is not a cut-vertex.

Finally, suppose that the branch rooted at v is big. In this case v may indeed be a cut-vertex. Namely, if $T_{\mathcal{B},v}$ contains multiple components of different colours, each of the edges added to the leafs in the branch could have both endpoints within the branch. To deal with such cut-vertices, we use a two-step strategy. In the first step, we want to ensure that the subgraph induced by $S_{\mathcal{B},v}$ becomes connected. We achieve this using the standard strategy of choosing v_t in a different component than u_t if $u_t \in S_{\mathcal{B},v}$. If $u_t \notin S_{\mathcal{B},v}$, we consider the round a failure. We note that as each component of $T_{\mathcal{B},v}$ contains at least one leaf of $T_{\mathcal{B}}$, by Corollary 5.9, $T_{\mathcal{B},v}$ contains at most $o(n)$ components. As $|S_{\mathcal{B},v}| > n/8$, the probability of adding successfully adding an edge in this first step is at least $1/8$. Then, by standard concentration inequalities, this step a.a.s. takes $o(n)$ rounds as well. We note that in the resulting graph, cut-vertex v then separates two components. In the second step of the strategy, we connect these two components by a single edge. We note that we can place such an edge successfully if $u_t \neq v$. As the probability of a failure round is thus $1/n$, by standard concentration inequalities, the number of rounds in this step is $O(1)$. We then note that there are at most 7 vertices $v \in B^*$ such $T_{\mathcal{B},v}$ is big. Hence, the total number of rounds to ensure that each of these cut-vertices is absorbed is a.a.s. $o(n)$.

Because the resulting graph then thus no longer contains any cut-vertices, the graph is

2-connected. Thus, in any case, we can ensure that graph G_2 becomes 2-connected in a.a.s. $o(n)$ rounds, as desired. □

Combining the analysis of these individual phases then results in the following lemma.

Lemma 5.12. $\tau_{\mathcal{C}_2} \leq \ln 2 + \ln(\ln 2 + 1))$.

Proof. The lemma directly follows from Propositions 5.8 and 5.11, and the fact that the 2-min process requires $(\ln 2 + \ln(\ln 2 + 1)) + o(1))n$ rounds. □

5.4 Pre-positional model

In this section, we consider k-connectedness in the pre-positional model. By Lemma 3.1, $\tau_{\mathcal{C}_k} \leq \tau'_{\mathcal{C}_k}$ for all $k \geq 1$. The lower bounds for $\tau_{\mathcal{C}_k}$ thus apply to $\tau'_{\mathcal{C}_k}$ as well.

We observe that the case for $k = 1$ is not as simple in the pre-positional model as in the post-positional model. Namely, there is no strategy to guarantee that u_t and v_t lie in different connected components (disregarding isolated vertices).

Lemma 5.13. $\tau'_{\mathcal{C}_1} = 1$.

Proof. We first observe that as $\tau_{\mathcal{C}_1} = 1$, we have the lower bound $\tau'_{\mathcal{C}_1} \geq 1$.

For the upper bound, we consider a strategy $\mathcal{S}$ which chooses u_t u.a.r. from all vertices contained in any connected component of minimum order. If v_t lands in a different connected component, we add edge $u_t v_t$. Otherwise we consider the round a failure round. Each successfully added edge then decreases the number of connected components in the graph by 1. We analyse the process with this strategy in a number of phases.

Let phase i be defined as the rounds in which the number of connected components in the graph decreases from $\frac{n}{2^{i-1}}$ to $\frac{n}{2^i}$. We note that phase i consists of $\frac{n}{2^{i-1}} - \frac{n}{2^i} = \frac{n}{2^i}$ non-failure rounds, and a number of failure rounds. Let T_i be the total number of rounds in phase i, and let f_i be the number of failure rounds in phase i. Thus, $T_i = \frac{n}{2^i} + f_i$.

We then observe that the smallest connected component in any round in phase i contains at most 2^i vertices. The probability that a round is a failure round is thus at most $2^i/n$. Suppose that $T_i \leq \alpha \cdot \frac{n}{2^i}$ for some fixed $\alpha > 1$. Then

$$\mathbb{E}[f_i] \leq \alpha \cdot \frac{n}{2^i} \cdot \frac{2^i}{n} = \alpha = O(1).$$

Moreover, by Markov's inequality, it then follows that a.a.s. $f_i = O(1)$. Thus, a.a.s. $T_i = \frac{n}{2^i} + O(1)$, validating our assumption that $T_i \leq \alpha \cdot \frac{n}{2^i}$.

The total number of rounds needed to ensure the graph is connected is then

$$\sum_{i>0} T_i = \sum_{i=1}^{\log_2 n} \left(\frac{n}{2^i} + O(1) \right) \leq n + O(\log_2 n) = (1 + o(1))n.$$

Therefore, $\tau'_{\mathcal{C}_1} \leq 1$, as desired. □

Thus, asymptotically, $\tau'_{\mathcal{C}_1} = \tau_{\mathcal{C}_1}$.

To prove the tight upper bound in the post-positional model for $k \geq 3$, Ben-Eliezer, Hefetz, Kronenberg, Parczyk, Shikhelman and Stojaković [4] introduce a strategy $\mathcal{S}^*_{\min}$. This strategy chooses vertex v_t u.a.r. from all vertices of $V(G_{t-1}) \setminus \{u_t\}$ that have the smallest number of distinct neighbours. Note that this process generates a multi-graph, and is therefore not quite equivalent to the min-degree process. They then analyse the graph resulting from this strategy in the post-positional process to show the tight upper bounds.

We note that strategy $\mathcal{S}^*_{\min}$ can be modelled in the pre-positional process by the following strategy: we choose u_t u.a.r. from all vertices that have the smallest number of distinct neighbours, and consider the round a failure if $u_t = v_t$. The probability of any given round being a failure round is thus $1/n$. Hence, the number of additional rounds needed to cover the additional failure rounds is a.a.s. $o(n)$. The tight upper bounds for $\tau_{\mathcal{C}_k}$ for $k \geq 3$ thus are also tight upper bounds for $\tau'_{\mathcal{C}_k}$. Hence, Lemma 5.14 follows.

Lemma 5.14. *$\tau'_{\mathcal{C}_k} = \tau_{\mathcal{C}_k}$ for all $k \geq 3$.*

The remaining open case in the pre-positional process is thus $k = 2$. We will consider the proof we have given for the tight upper bound for the post-positional case, and give an outline of how to adapt it for the pre-positional process.

As discussed in Section 4.3, the pre-positional process can be used to model the min-degree process in a similar fashion to the post-positional model. Hence, it suffices to consider the phases of the proof where we define and use different strategies.

We first note that connecting the graph in the second phase is not achievable with the same strategy as in the proof of Proposition 5.10. Instead we use the same strategy defined in the proof of Lemma 5.13. By analogous analysis, it then follows that Proposition 5.10 also holds in the pre-positional process.

Next, we consider the third phase, in which we aim to ensure that the graph is 2-connected. In the proof of Proposition 5.11, there are two cases to examine. If the graph contains a block B^* of size at least $n/4$, we choose u_t to be a leaf block in $T_{\mathcal{B}}$ that has not yet received an edge to B^*. We note there is then a probability of at least $n/4 - o(1)$ to successfully add an edge to B^* similar to in the post-positional argument. To eliminate the remaining cut-vertices that separate individual blocks from the rest of the graph, we again use a strategy in the style used in the proof of Lemma 5.13. It then follows by standard concentration inequalities that it takes $o(n)$ additional rounds to ensure the graph is 2-connected.

If the graph does not contain a block B^* of size at least $n/4$, we again use the balanced colouring to efficiently absorb cut-vertices. We pick u_t to be a leaf that has not yet received an edge, and by the size of the colour classes, there is probability at least c for some constant $c > 0$ such that v_t lands in the colour class not containing u_t. Moreover, for leaves in small branches, we can additionally require v_t to be in a different branch and maintain at least a constant positive probability of successfully adding an edge. We absorb the cut-vertices separating individual blocks from the rest of the graph in a similar fashion as the case where

there exists a block B^* of size at least $n/4$. Finally, we consider the two-step strategy to absorb cut-vertices in B^*. To connect all components in a big branch, one can again use the strategy in the style used in the proof of Lemma 5.13. For the second step, we choose u_t in the smaller of the two components of the graph obtained by removing the cut-vertex. As the success probability is then at least $1/2$, we note that this step again a.a.s. takes $O(1)$ rounds. Thus, overall, also in the case where the graph does not contain a block of size at least $n/4$, a.a.s. $o(n)$ rounds suffice to ensure the graph becomes connected.

Hence, we conclude that Proposition 5.11 also holds for the pre-positional process. As thus all phases of the upper bound algorithm for the post-positional process can be modelled by similar strategies in the pre-positional process, with asymptotically equal results, Lemma 5.15 follows.

Lemma 5.15. $\tau'_{\mathcal{C}_2} = \tau_{\mathcal{C}_2}$.

Theorem 5.2 then directly follows from Lemmas 5.13, 5.14, and 5.15.

Chapter 6

k-Factors

A *factor* of a graph G is a spanning regular subgraph in G. In this chapter, we consider the construction of a *k-factor*, a k-regular factor. A k-factor is thus a subgraph $H \subseteq G$ such that $V(H) = V(G)$ and $\deg_H(v) = k$ for all vertices $v \in V(H)$. Note that a 1-factor is therefore a perfect matching. We denote the property of a graph containing a k-factor by $\mathcal{F}_k$.

The property $\mathcal{F}_1$ of containing a perfect matchings was first studied in the seminal paper by Ben-Eliezer, Hefetz, Kronenberg, Parczyk, Shikhelman and Stojaković [4]. They showed both upper and lower bounds for $\tau_{\mathcal{F}_1}$. This problem has since been studied more extensively by Gao, MacRury and Prałat [11]. They provide stronger upper and lower bounds for $\tau_{\mathcal{F}_1}$, which they refer to as τ_{PM}. Specifically, they show $0.93261 \leq \tau_{\mathcal{F}_1} \leq 1.20524$.

Upper bounds for the case where k is dependent on n follow from the work of Ben-Eliezer, Gishboliner, Hefetz and Krivelevich [3]. By fixing a family of k-regular spanning graphs, their Theorem 1.3 shows that $\tau_{\mathcal{F}_k} \leq k/2 + o(k)$ if $k = \omega(\log(n))$ and $\tau_{\mathcal{F}_k} \leq 3k/2 + o(k)$ otherwise. Note that their bound is thus asymptotically tight for $k = \omega(\log(n))$.

A related problem is that of constructing a Hamiltonian cycle as fast as possible. Let HAM denote the property of containing a Hamiltonian cycle. We note that as a Hamiltonian cycle is a 2-factor, $\tau_{\mathcal{F}_2} \leq \tau_{\mathrm{HAM}}$. It was first observed by Ben-Eliezer, Hefetz, Kronenberg, Parczyk, Shikhelman and Stojaković [4] that $\tau_{\mathrm{HAM}} \leq 3$. Further bounds for τ_{HAM} were shown by Gao, Kamiński, MacRury and Prałat [10]. Gao, MacRury and Prałat [12] subsequently showed improved bounds. The strongest upper bound currently known was shown by Frieze and Sorkin [8]. Building on the work of Gao, MacRury and Prałat, they show that $\tau_{\mathrm{HAM}} \leq 1.85$.

Determining the existence and value of $\tau_{\mathcal{F}_k}$ for all fixed $k \geq 1$ was posed as Problem 1.6 by Ben-Eliezer, Gishboliner, Hefetz and Krivelevich [3]. In this chapter we consider a method for bounding $\tau_{\mathcal{F}_k}$ for all fixed $k \geq 2$ from above. We provide explicit upper bounds for a number of small values for k.

The best known lower bounds for $\tau_{\mathcal{F}_k}$ for general k are the tight asymptotic lower bounds shown by Ben-Eliezer, Hefetz, Kronenberg, Parczyk, Shikhelman and Stojaković [4] on $\tau_{\mathcal{D}_k}$

to bound $\tau_{\mathcal{F}_k}$ from below. Naturally, for a graph G to contain a k-factor, it must hold that $\delta(G) \geq k$.

6.1 Overview

Our general strategy for bounding $\tau_{\mathcal{F}_k}$ from above is to describe algorithms that construct a k-factor in the semi-random graph process. That is, we maintain a spanning subgraph $H_t \subseteq G_t$ for each round t, where we aim to make H_t become k-regular. In our algorithms, we will ensure that $\Delta(H_t) \leq k$ at all times. We refer to vertices $v \in V(G_t)$ such that $\deg_{H_t}(v) = k$ as *saturated* in round t. Correspondingly, we refer to all vertices $v \in V(G_t)$ such that $\deg_{H_t}(v) < k$ as *unsaturated* vertices. Additionally, we refer to vertices $v \in V(G_t)$ such that $\deg_{H_t}(v) = k - 1$ as *critical* vertices.

A natural way of measuring the progress of constructing a k-factor is the difference between the desired degree k and the actual degree $\deg_{H_t}(v)$, summed over all vertices $v \in V(H_t)$. We denote this measure, which we refer to as the *missing degree sum*, by Λ_k, and define it formally as

$$\Lambda_k(H_t) = \sum_{v \in V(H_t)} \Big(k - \deg_{H_t}(v)\Big).$$

Note that $\Lambda_k(H_t) \geq 0$ always holds, and that once $\Lambda_k(H_t) = 0$, subgraph H_t is a k-factor in graph G_t.

6.2 Perfect matching

The proof for the upper bound for $\tau_{\mathcal{F}_1}$ by Gao, MacRury and Prałat [11] is based on applying Wormald's differential equation method to a probabilistic algorithm. Their main tool is that of augmentation paths of length 3. Let $u, v \in V(G_t)$ be two saturated vertices (that is, $\deg_{H_t}(u) = \deg_{H_t}(v) = 1$) connected by an edge in subgraph H_t. Moreover, let $x, y \in V(G_t)$ be two unsaturated vertices. As vertices u and v are already saturated, we cannot simply add more edges incident with u and v in subgraph H_t. However, if we add edges ux and vy to subgraph H_t and remove edge uv, the degrees of u and v are not altered. At the same time, vertices x and y have become saturated. Hence, all four vertices u, v, x, and y are now saturated. This augmentation technique becomes increasingly relevant as more vertices become saturated. See Figure 6.1 for an illustration of such an augmentation as used in the perfect matching algorithm.

We also note that augmentation is not perfect. Because the original edge uv is no longer considered as part of subgraph H_t, the round it was added in is wasted. The missing degree sum Λ_k decreased by 2 because of the augmentation operation, while also taking 2 rounds. However, if we add an edge between two unsaturated vertices, the missing degree sum decreases by 2 in a single round. Hence, in most cases, augmentation is only utilised to be able to make some use of rounds where vertex u_t is already saturated.

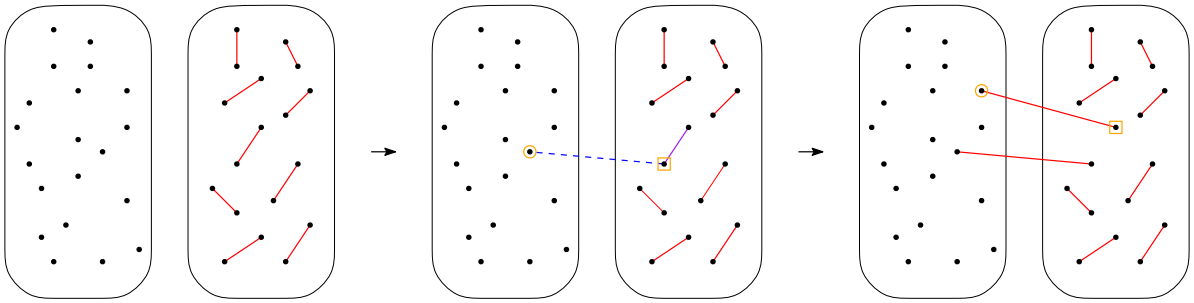

Figure 6.1: Illustration of an augmentation as used in the perfect matching algorithm by Gao, MacRury and Prałat [11].

The algorithm by Gao, MacRury and Prałat [11] adds an edge between two unsaturated vertices if possible, and otherwise attempts to construct such augmenting paths. By choosing v_t strategically based on the number of circles and squares that have hit each vertex, they achieve the bound $\tau_{\mathcal{F}_1} \leq 1.20524$.

6.3 Terminology

We aim to use the same main idea of augmentation paths of length 3 for algorithms constructing k-factors. To that purpose, we first generalise this concept.

During our algorithm, we will maintain a list of edges $E \subseteq E(H_t)$ that we wish to use for augmentation. We refer to these edges as *augmentation edges*. Let $uv \in E$ be such an augmentation edge. As previously noted, we only wish to start creating an augmenting path if we hit a saturated vertex. Hence, for all $uv \in E$, either vertex u or vertex v is saturated. Note that this differs from the perfect matching case, where the existence of an edge necessitates that both endpoints are saturated. Then, for $\ell = \min\left(\deg_{H_t}(u), \deg_{H_t}(v)\right)$, we refer to uv as a *degree-ℓ augmentation edge*. Thus, if both u and v are saturated, uv is a degree-k augmentation edge, and otherwise ℓ is the degree of the unsaturated vertex. We say that ℓ is the degree of the augmentation edge.

Let $x, y \in V(H_t)$ be two unsaturated vertices, distinct from u and v. Suppose that u is a saturated vertex. If we hit u, we may add an edge connecting u to x. We refer to edge ux as the *priming edge*. Moreover, we refer to vertex x as the *priming vertex*, and to vertex u as the *primed vertex*. We note that edge ux is only added to graph G_t and not subgraph H_t, to maintain $\deg_{H_t}(u) \leq k$. If a priming edge is incident to an augmentation edge, we say that the augmentation edge is *primed*. Otherwise, we say the augmentation edge is *unprimed*. We refer to unsaturated vertices that are not incident with an augmentation edge as *uncovered vertices*.

In some cases, we may no longer wish to consider an edge $e \in E(H_t)$ as an augmentation edge, and hence remove it from set E. We refer to this step as *demoting* augmentation edge e. In other cases, we may wish to no longer consider a priming edge $e \in E(G_t)$. We

refer to this as *destroying* priming edge p. Note that the incident augmentation edges may become unprimed as a result.

In accompanying figures, we will colour unprimed augmentation edges red, primed augmentation edges purple, and priming edges blue.

6.4 Running the perfect matching algorithm k times

We note that the disjoint union of k edge-disjoint perfect matchings gives a k-factor. Hence, one approach for constructing a k-factor for fixed k is to use the perfect matching algorithm by Gao, MacRury and Prałat [11] k times. However, the resulting perfect matchings are not necessarily edge-disjoint. In this section, we show that the number of such parallel edges is bounded. We will then use a clean-up algorithm to obtain a k-factor.

Let $G_{\mathrm{PM},k}$ be the multigraph resulting from running the perfect matching algorithm k times with spanning subgraph $H_{\mathrm{PM},k}$ such that $\deg_{H_{\mathrm{PM},k}}(v) = k$ for all vertices $v \in V(G_{\mathrm{PM},k})$. Let $S_{\mathrm{PM},k}$ be the set of edges e in $E(H_{\mathrm{PM},k})$ such that there exists at least one other edge in $E(H_{\mathrm{PM},k})$ with the same endpoints.

Proposition 6.1. $|S_{PM,k}| = O(1)$.

Proof. Each iteration of the perfect matching algorithm results in $n/2$ edges being added to subgraph $H_{\mathrm{PM},k}$. Moreover, as the vertices are indistinguishable, each of the $\binom{n}{2}$ possible edges is equally likely to be added in each iteration. Let S_i for $1 \leq i \leq k$ be the set of edges added to $H_{\mathrm{PM},k}$ in the i^{th} perfect matching iteration. That is, sets $S_1, S_2, \ldots, S_k$ partition $E(H_{\mathrm{PM},k})$. Let X be a random variable denoting the number of pairs $\{e_1, e_2\} \subseteq E(H_{\mathrm{PM},k})$ such that e_1 and e_2 have the same endpoints. Because each pair of edges from different iterations has a probability of $1/\binom{n}{2}$ of having identical endpoints, and as there are $\binom{kn/2}{2} - k \cdot \binom{n/2}{2}$ such pairs,

$$\begin{aligned}
\mathbb{E}[X] &= \left(\binom{k \cdot \frac{n}{2}}{2} - k \cdot \binom{\frac{n}{2}}{2}\right) \cdot \frac{1}{\binom{n}{2}} \\
&= \frac{kn(kn-2) - kn(n-2)}{4} \cdot \frac{2}{n(n-1)} \\
&\leq \frac{k(kn-2) - k(n-2)}{2(n-2)} \\
&= O(1).
\end{aligned}$$

Then, as X is an upper bound for $|S_{\mathrm{PM},k}|$, the desired result follows. □

Next, we describe a generalisation of the clean-up algorithm as used by Gao, MacRury and Prałat [11]. This clean-up algorithm is used to fix the small number of multi-edges in the constructed spanning subgraph.

Lemma 6.2 (Clean-up Lemma)**.** *Suppose that for $H_t \subseteq G_t$ and $\epsilon = \Lambda_k(H_t)/n$ it holds that $\epsilon k^2 < 10^{-14}$ for all $n \geq 1$. Then there exists a strategy such that a k-factor H'_t is constructed a.a.s. in $10^{-5} \cdot n$ additional rounds.*

Proof. Let $j_0 = \Lambda_k(H_t)$ be the missing degree sum. Thus, $j_0 = \epsilon n$. We may assume that $\epsilon = \theta(1)$ by removing additional edges from H_t. Let $j_\ell = \lfloor (\frac{4k-1}{4k})^\ell j_0 \rfloor$ for all $\ell \geq 1$. The clean-up algorithm is an iterative algorithm. In iteration ℓ, the missing degree sum is decreased by $j_{\ell-1} - j_\ell$ in the following two phases:

(1) We create a set S_ℓ of augmentation edges in $\lfloor \frac{\sqrt{3j_{\ell-1}n}}{4} \rfloor$ rounds, which is initially empty. In each of these rounds, if $\deg_{H_t}(u_t) = 0$, we consider the round a failure round. Otherwise, pick a neighbour w of u_t in H_t. If set S_ℓ already contains edge $u_t w$, we also consider the round a failure. If not, we add $u_t w$ to S_ℓ and pick v_t u.a.r. to be an unsaturated vertex not adjacent to u_t or w in H_t. We consider $u_t w$ to be a primed augmentation edge with priming edge $u_t v_t$.

(2) The aim for the subsequent rounds is to use the primed augmentation edges created in the first phase to decrease the missing degree sum. In these rounds, if u_t is unsaturated, we choose v_t u.a.r. to be an unsaturated vertex not adjacent to u_t in H_t and add edge $u_t v_t$ to H_t. Any edge in S_ℓ for which u_t or v_t is the priming vertex is removed from S_ℓ. Otherwise, if u_t is saturated and incident to a primed augmentation edge $u_t w \in S_\ell$ such that w is the primed vertex, we choose v_t u.a.r. to be an unsaturated vertex not adjacent to u_t and w in H_t and distinct from the priming vertex for augmentation edge $u_t w$. We then augment along augmentation edge $u_t w$, and remove $u_t w$ from S_ℓ. Additionally, we remove every augmentation edge with v_t or the priming vertex of $u_t w$ as its priming vertex from S_ℓ. This phase terminates once $\Lambda_k(H_t) \leq j_\ell$.

Note that the algorithm as outlined above fails if there is a single unsaturated vertex left. In this case, a single additional iteration where we allow said unsaturated vertex to become incident with both endpoints of augmentation edges suffices.

Let T_ℓ be the total number of rounds in iteration ℓ of the algorithm. We claim that $\sum_{\ell \geq 1} T_\ell \leq 12\sqrt{\epsilon} k^2 n$ a.a.s.

Proposition 6.3. *In each iteration $\ell \geq 1$, $|S_\ell| \geq \frac{\sqrt{3j_{\ell-1}n}}{5}$ after phase 1 a.a.s.*

Proof. We first aim to bound the probability that a round in phase 1 is a failure round. We observe that the missing degree sum is an upper bound for the number of unsaturated vertices, and hence for the number of isolated vertices in H_t. Hence, as $j_\ell \leq j_0 \leq \epsilon n$, there are at most ϵn vertices v such that $\deg_{H_t}(v) = 0$.

Moreover, the probability that we hit a non-isolated vertex and choose w such that $u_t w$ is already contained in S_ℓ is bounded by the probability of hitting a vertex incident with an

edge in S_ℓ. We can bound this probability in turn by

$$\begin{aligned}\frac{2\left|S_\ell\right|}{n} &\leq \frac{2 \cdot \frac{\sqrt{3j_{\ell-1}n}}{4}}{n} \\ &= \frac{\sqrt{3\left(\frac{4k-1}{4k}\right)^{\ell-1}\epsilon n}}{2\sqrt{n}} \\ &< \sqrt{\epsilon}.\end{aligned}$$

Hence, the probability of adding a new augmentation edge to S_ℓ in any given round in phase 1 is at least $1-\epsilon-\sqrt{\epsilon}$. We then stochastically couple the number of rounds where an edge is added to S_ℓ with the binomial distribution with $\sqrt{3j_{\ell-1}n}/4$ trials and a success probability of $1-\epsilon-\sqrt{\epsilon}$. Thus, let $X_\ell \sim \text{Bin}\left(\sqrt{3j_{\ell-1}n}/4, 1-\epsilon-\sqrt{\epsilon}\right)$ be coupled such that $|S_\ell| \geq X_\ell$ (that is, we let X_ℓ and $|S_\ell|$ be random variables in the same probability space such that the i^{th} trial of X_ℓ is forced to be unsuccessful if no edge is added to S_ℓ in the corresponding round, and otherwise the i^{th} trial is successful with a exactly the probability needed to ensure that the overall probability of the i^{th} trial being successful is $1-\epsilon-\sqrt{\epsilon}$). We note that thus $\mathbb{P}\left(|S_\ell| < \frac{\sqrt{3j_{\ell-1}n}}{5}\right) \leq \mathbb{P}\left(X_\ell < \frac{\sqrt{3j_{\ell-1}n}}{5}\right)$. We will bound the latter to show the desired result.

Using the Chernoff-Hoeffding bound and the fact that $\epsilon < \epsilon k^2 < 10^{-14}$, we find

$$\begin{aligned}\mathbb{P}\left(X_\ell < \frac{\sqrt{3j_{\ell-1}n}}{5}\right) &= \mathbb{P}\left(X_\ell < \left(1 - \frac{1-5\epsilon-5\sqrt{\epsilon}}{5-5\epsilon-5\sqrt{\epsilon}}\right) \cdot \frac{\sqrt{3j_{\ell-1}n}}{4} \cdot (1-\epsilon-\sqrt{\epsilon})\right) \\ &\leq \exp\left(-\frac{\left(\frac{1-5\epsilon-5\sqrt{\epsilon}}{5-5\epsilon-5\sqrt{\epsilon}}\right)^2}{2} \cdot \frac{\sqrt{3j_{\ell-1}n}}{4} \cdot (1-\epsilon-\sqrt{\epsilon})\right) \\ &= \exp\left(-\theta(1)\sqrt{j_{\ell-1}n}\right) \\ &= \exp\left(-\theta(1)\sqrt{\left(\frac{4k-1}{4k}\right)^{\ell-1}\epsilon n^2}\right) \\ &= O\left(e^{-n}\right).\end{aligned}$$

Hence, as desired,

$$\begin{aligned}\mathbb{P}\left(|S_\ell| \geq \frac{\sqrt{3j_{\ell-1}n}}{5}\right) &= 1 - \mathbb{P}\left(|S_\ell| < \frac{\sqrt{3j_{\ell-1}n}}{5}\right) \\ &\geq 1 - \mathbb{P}\left(X_\ell < \frac{\sqrt{3j_{\ell-1}n}}{5}\right) \\ &= 1 - O\left(e^{-n}\right).\end{aligned}$$

■

Next, we consider a lower bound on the number of edges in set S_ℓ during phase 2.

Proposition 6.4. *$|S_\ell| \geq \frac{\sqrt{2j_{\ell-1}n}}{6}$ in any round in phase 2 of iteration ℓ.*

Proof. Let Y_ℓ be a random variable denoting the number of edges that are removed from S_ℓ in phase 2 in iteration ℓ. Because there are at least j_ℓ/k unsaturated vertices in each round in iteration ℓ, the expected number of augmentation edges for which a vertex v is a priming vertex is at most $\frac{\sqrt{3j_{\ell-1}n}/4}{j_\ell/k}$. Hence, the expected number of edges to be removed from S_ℓ in a single round in phase 2 is at most $1 + \frac{\sqrt{3j_{\ell-1}n}/4}{j_\ell/k}$ for non-failure rounds, and 0 for failure rounds. There are $(j_{\ell-1} - j_\ell)/2$ non-failure rounds in iteration ℓ to decrease the missing degree sum by $j_{\ell-1} - j_\ell$. Hence, $\mathbb{E}[Y_\ell]$ is bounded from above by $(j_{\ell-1} - j_\ell)/2 \cdot \left(1 + \frac{\sqrt{3j_{\ell-1}n}/4}{j_\ell/k}\right)$.

Let $Y_{\ell,m}$ be a random variable denoting the number of priming edges that get destroyed in round m in phase 2 of iteration ℓ. This quantity can then be stochastically coupled with a random variable $Y'_{\ell,m}$ such that $Y'_{\ell,m} \sim \text{Bin}\left(\frac{\sqrt{3j_{\ell-1}n}}{4}, \frac{1}{j_\ell/k}\right)$. Moreover, let $Y'_\ell \sim \text{Bin}\left(\frac{j_{\ell-1}-j_\ell}{2} \cdot \frac{\sqrt{3j_{\ell-1}n}}{4}, \frac{1}{j_\ell/k}\right)$ such that $\sum_{m=1}^{m'} Y_{\ell,m} \leq Y'_\ell$ where m' is the number of rounds in phase 2 of iteration ℓ. Note that by this definition, $Y_\ell \leq (j_{\ell-1} - j_\ell)/2 + Y'_\ell$.

Using the Chernoff-Hoeffding bound with $\mathbb{E}[Y'_\ell] = \frac{j_{\ell-1}-j_\ell}{2} \cdot \frac{\sqrt{3j_{\ell-1}n}}{4} \cdot \frac{1}{j_\ell/k}$, we then find

$$\begin{aligned}
\mathbb{P}\left(Y'_\ell > \frac{3}{2} \cdot \frac{j_{\ell-1} - j_\ell}{2} \cdot \frac{\sqrt{3j_{\ell-1}n}}{4} \cdot \frac{1}{j_\ell/k}\right) &\leq \exp\left(-\frac{(1/2)^2}{3} \cdot \frac{j_{\ell-1} - j_\ell}{2} \cdot \frac{\sqrt{3j_{\ell-1}n}}{4} \cdot \frac{1}{j_\ell/k}\right) \\
&= \exp\left(-\theta(1) \cdot \frac{j_{\ell-1}}{4k} \cdot \sqrt{j_{\ell-1}n} \cdot \frac{1}{j_{\ell-1} \cdot \frac{4k-1}{4k}}\right) \\
&= \exp\left(-\theta(1) \cdot \sqrt{j_{\ell-1}n}\right) \\
&= \exp\left(-\theta(1) \cdot \sqrt{\left(\frac{4k-1}{4k}\right)^{\ell-1} \epsilon n^2}\right) \\
&= O\left(e^{-n}\right).
\end{aligned}$$

Thus, $Y_\ell \leq \frac{j_{\ell-1}-j_\ell}{2} + \frac{3}{2} \cdot \frac{j_{\ell-1}-j_\ell}{2} \cdot \frac{\sqrt{3j_{\ell-1}n}}{4} \cdot \frac{1}{j_\ell/k}$ a.a.s.

Hence, a.a.s. at the end of phase 2, using Proposition 6.3,

$$\begin{aligned}
|S_\ell| &\geq \frac{\sqrt{3j_{\ell-1}n}}{5} - Y_\ell \\
&\geq \frac{\sqrt{3j_{\ell-1}n}}{5} - \frac{j_{\ell-1}-j_\ell}{2} - \frac{3}{2}\cdot\frac{j_{\ell-1}-j_\ell}{2}\cdot\frac{\sqrt{3j_{\ell-1}n}}{4}\cdot\frac{1}{j_\ell/k} \\
&= \frac{\sqrt{3j_{\ell-1}n}}{5} - \frac{j_{\ell-1}}{8k} - \frac{3}{64}\cdot\frac{j_{\ell-1}}{j_\ell}\cdot\sqrt{3j_{\ell-1}n} \\
&\geq \frac{\sqrt{3j_{\ell-1}n}}{5} - \frac{j_{\ell-1}}{8k} - \frac{3}{64}\cdot\frac{4}{3}\cdot\sqrt{3j_{\ell-1}n} \\
&= \frac{11}{80}\cdot\sqrt{3j_{\ell-1}n} - \frac{j_{\ell-1}}{8k} \\
&\geq \frac{\sqrt{2j_{\ell-1}n}}{6},
\end{aligned}$$

where the last step follows from the fact that $j_{\ell-1} \leq j_0 = \epsilon n \leq \epsilon k^2 n \leq 10^{-14}n$.

Then, no edges are added to set S_ℓ in phase 2, the proposition follows. ■

Proposition 6.4 gives a lower bound on the probability of a non-failure round in phase 2. We then consider the number of rounds in phase 2 for a single iteration of the clean-up algorithm.

Proposition 6.5. *For each iteration $\ell \geq 1$, it a.a.s. holds that $T_\ell < \frac{3}{2}\sqrt{j_{\ell-1}n}$.*

Proof. Let $T_{\ell,1}$ and $T_{\ell,2}$ denote the number of rounds in phases 1 and 2 of iteration ℓ respectively. Thus, $T_\ell = T_{\ell,1} + T_{\ell,2}$. Moreover, by the definition of the clean-up algorithm, $T_{\ell,1} = \lfloor \frac{\sqrt{3j_{\ell-1}n}}{4} \rfloor$.

Let Z be a negative binomially distributed random variable with the number of successful draws being $(j_{\ell-1} - j_\ell)/2$ and the probability of success for a single draw being $\frac{\sqrt{2j_{\ell-1}n}/6}{n}$. Using Proposition 6.4, we then stochastically couple $T_{\ell,2}$ with random variable Z such that $T_{\ell,2} \leq Z$.

We note that $\mathbb{E}[Z] = \left(\frac{\sqrt{2j_{\ell-1}n}/6}{n}\right)^{-1} \cdot \frac{j_{\ell-1}-j_\ell}{2} = \frac{6\sqrt{n}}{\sqrt{2j_{\ell-1}}} \cdot \frac{j_{\ell-1}}{8k}$. We aim to bound the probability that $T_{\ell,2} > (3/2)\cdot\mathbb{E}[Z]$. To that purpose, we note that

$$\mathbb{P}\left(T_{\ell,2} > \frac{3}{2}\mathbb{E}[Z]\right) \leq \mathbb{P}\left(Z > \frac{3}{2}\mathbb{E}[Z]\right) < \mathbb{P}\left(Z' < \frac{j_{\ell-1}}{8k}\right),$$

where $Z' \sim \mathrm{Bin}\left(\frac{3}{2}\mathbb{E}[Z], \frac{\sqrt{2j_{\ell-1}}}{6\sqrt{n}}\right)$. We observe that

$$\mathbb{E}[Z'] = \frac{3}{2}\cdot\frac{6\sqrt{n}}{\sqrt{2j_{\ell-1}}}\cdot\frac{j_{\ell-1}}{8k}\cdot\frac{\sqrt{2j_{\ell-1}}}{6\sqrt{n}} = \frac{3}{2}\cdot\frac{j_{\ell-1}}{8k}.$$

Then, using the Chernoff-Hoeffding bound, we find

$$\begin{aligned}\mathbb{P}\left(Z' < \frac{j_{\ell-1}}{8k}\right) &= \mathbb{P}\left(Z' < \frac{2}{3}\cdot\mathbb{E}[Z']\right)\\ &\leq \exp\left(-\frac{(1/3)^2}{2}\cdot\mathbb{E}[Z']\right)\\ &= \exp\left(-\frac{1}{18}\cdot\frac{3}{2}\cdot\frac{j_{\ell-1}}{8k}\right)\\ &= \exp\left(-\theta(1)\left(\frac{4k-1}{4k}\right)^{\ell-1}\epsilon n\right)\\ &= O\left(e^{-n}\right).\end{aligned}$$

Hence, it a.a.s. follows that

$$\begin{aligned}T_\ell &= T_{\ell,1} + T_{\ell,2}\\ &\leq \frac{\sqrt{3j_{\ell-1}n}}{4} + \frac{3}{2}\mathbb{E}[Z]\\ &= \frac{\sqrt{3j_{\ell-1}n}}{4} + \frac{3}{2}\cdot\frac{6\sqrt{n}}{\sqrt{2j_{\ell-1}}}\cdot\frac{j_{\ell-1}}{8k}\\ &< \frac{\sqrt{3j_{\ell-1}n}}{4} + \sqrt{j_{\ell-1}n}\\ &< \frac{3}{2}\sqrt{j_{\ell-1}n},\end{aligned}$$

as desired. ■

Finally, we then bound the number of rounds required by the complete clean-up algorithm. By Proposition 6.5,

$$\begin{aligned}\sum_{\ell\geq 1} T_\ell &\leq \sum_{\ell\geq 1}\frac{3}{2}\sqrt{j_{\ell-1}n}\\ &= \sum_{\ell\geq 1}\frac{3}{2}\sqrt{\left(\frac{4k-1}{4k}\right)^{\ell-1}\epsilon n^2}\\ &= \frac{3}{2}\sqrt{\epsilon}n\cdot\frac{2k}{2k-\sqrt{4k^2-k}}\\ &\leq \frac{3}{2}\sqrt{\epsilon}n\cdot 8k\\ &\leq 12\sqrt{\epsilon}kn.\end{aligned}$$

Then, as $\epsilon k^2 < 10^{-14}$, it follows that a.a.s. $T_\ell \leq 10^{-5}n$, as desired. □

We can then use the clean-up algorithm to finish the argument for running the perfect matching algorithm by Gao, MacRury and Prałat [11] k times.

Lemma 6.6. $\tau_{\mathcal{F}_k} \leq 1.20524 \cdot k + 10^{-5}$ *for all* $k \geq 1$.

Proof. The result follows directly from [11, Theorem 1.1], Proposition 6.1, and Lemma 6.2. □

We will use this bound as a benchmark for our new method.

6.5 Issues in extending the perfect matching algorithm

Instead of repeatedly using the perfect matching algorithm developed by Gao, MacRury and Prałat [11], we aim to use its concepts to design a algorithm for general k-factors leading to stronger bounds. At first glance, it may seem like their strategy for $k = 1$ is easily extendable to general k. There are however a number of factors that make this generalisation non-trivial.

We first observe that each edge in the partial matching H_t can be used as an augmentation edge. It makes sense to consider all such edges for augmentation, as each of the endpoints is saturated. This is not generally the case when extending to general k. Namely, for $k \geq 2$, subgraph H_t may contain edges between unsaturated vertices. If an edge in H_t joins two unsaturated vertices, it is inefficient to consider it as an augmentation edge. Namely, any edge priming such an augmentation edge could instead be directly added.

Hence, we only consider edges with at least one saturated endpoint for augmentation. The next question that then arises is whether we should consider all such edges for augmentation. For each individual edge incident with a saturated vertex, it is useful to consider it as an augmentation edge when hitting said saturated vertex. However, issues arise if we consider multiple such edges simultaneously. Consider the number of augmentation edges a vertex can be incident to. In the perfect matching case, each vertex can only be incident with one augmentation edge, because it is incident with at most one edge in subgraph H_t. However, for $k > 1$, each saturated vertex may be incident with up to k augmentation edges. Suppose such a saturated vertex is hit, and a priming edge is added. We then have two options in the design of the algorithm. Either we consider all augmentation edges incident with said vertex primed, or we assign the priming edge to a specific augmentation edge.

If we consider all augmentation edges incident with the hit vertex to be primed, issues arise when an augmentation involving the priming edge occurs. Namely, to analyse the trajectory of the random variables in the process using Wormald's differential equation method, we then need to describe the expected number of augmentation edges of each degree that become unprimed. If instead the priming edge is associated to a single augmentation edge, another issue comes up. Namely, one then requires the probability that in a given round a degree-ℓ augmentation edge becomes primed. This probability is non-trivial due to the existence of both primed and unprimed augmentation edges of varying degrees.

We have not been able to define a set of random variables and accompanying expected change equations that model these choices. We run into similar issues when considering multiple priming edges assigned to a single augmentation edge. We then need to track the distribution of priming edges per type of augmentation edge to describe the number of primed augmentation edges of each degree.

To avoid these issues, we require that augmentation edges are vertex-disjoint, and that each augmentation edge is incident with at most one priming edge. Moreover, each priming edge is incident with exactly one augmentation edge, as otherwise augmentation would potentially lead to two augmenting edges sharing a common endpoint.

6.6 Algorithm for upper bound

In this section we describe our algorithm for constructing a k-factor. It generalises the perfect matching algorithm by Gao, MacRury and Prałat [11], while taking into account the issues outlined in Section 6.5.

6.6.1 Setup

We first provide an informal description of the mechanisms we use in our algorithm. As described in Section 6.3, we maintain a spanning subgraph H_t and a list E of augmentation edges. For each of these augmentation edges, we additionally track whether they are primed, and if so with what priming edge. Every round, we choose v_t based on the degree of u_t in H_t, whether u_t is an endpoint of an augmentation edge, and the status of the augmentation edge if u_t is incident with one.

We consider three classes of vertices based on their degree. Naturally, we consider saturated vertices as an important class which we treat differently than unsaturated vertices. We split the unsaturated vertices into two classes; critical vertices (those of degree $k-1$ in H_t) and those of degree at most $k-2$. We make this distinction to have more control about adding edges incident with vertices that are on the brink of becoming saturated. In our algorithm, we want to ensure that initially each saturated vertex is incident with an augmentation edge. Hence, if u_t is critical and uncovered, we want to choose v_t such that u_tv_t becomes an augmentation edge.

Generally, if u_t is uncovered, we choose v_t to be unsaturated as well and add edge u_tv_t to subgraph H_t. If u_t is critical, we additionally want to ensure that v_t is not incident with an augmentation edge, so that u_tv_t can become an augmentation edge. Note that if v_t is critical and uncovered, u_tv_t will also become an augmentation edge.

If u_t is unsaturated but incident with an augmentation edge, the strategy depends on the status of said augmentation edge. If it is unprimed, we choose v_t to be unsaturated and non-critical, and add u_tv_t to H_t. Namely, augmentation edges are a tool to make use of rounds where a saturated vertex is hit. Hence, we only prime an augmentation edge when hitting a saturated endpoint. If the augmentation edge is instead primed, we have two

options. Either we again choose v_t to be unsaturated and non-critical and add edge $u_t v_t$ to H_t, or we augment along the augmentation edge. Note that as we only prime from saturated vertices, u_t cannot be the primed vertex. We choose either option with a fixed probability.

We have a separate strategy if u_t is critical and incident with an augmentation edge. We can still choose either of the previous two options, which we again do with fixed probability. However, there is also a third option. We can choose v_t to be critical as well, and incident with another augmentation edge. Edge $u_t v_t$ is then added to H_t, and as a result both of the critical vertices become saturated. Moreover, the degree-$(k-1)$ augmentation edges they were incident with both become degree-k augmentation edges. This move is illustrated in Figure 6.2.

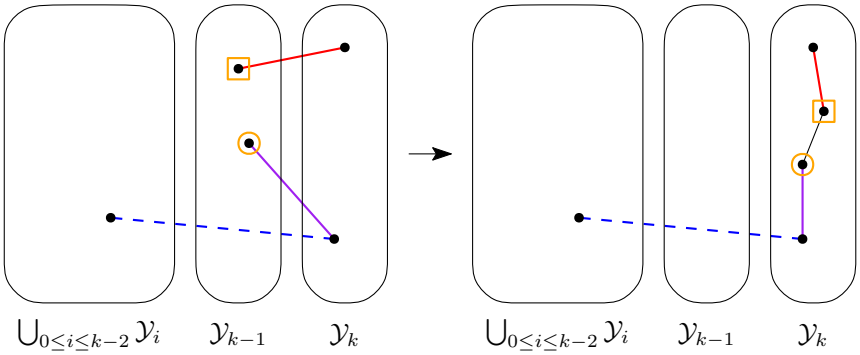

Figure 6.2: Possible move when u_t is critical and incident with an augmentation edge.

If u_t is instead saturated, the focus lies on augmentations. Namely, if u_t is not incident with an augmentation edge, there is no useful edge to be added and we consider the round a failure. For this reason, our algorithm initially ensures that all saturated vertices are contained in an augmentation edge. Moreover, if an augmentation is performed along an augmentation edge, each of the two resulting new edges in H_t is considered an augmentation edge if at least one of its endpoints is saturated.

If u_t is saturated and part of an augmentation edge, the strategy again depends on the status of the augmentation edge. If it is unprimed, we choose v_t from all uncovered vertices, and prime the augmentation edge with edge $u_t v_t$. If it is already primed, the strategy depends on the type of augmentation edge. If the augmentation edge is of degree at most $k-1$, vertex u_t must be the primed vertex. Then, as we allow at most one priming edge per augmentation edge, we also consider the round a failure. If it is instead a degree-k augmentation edge, there are two possibilities. If u_t is the primed vertex, we similarly consider the round a failure. Otherwise, we augment along the augmentation edge and choose v_t to be an uncovered vertex.

If at any point during the algorithm a vertex becomes saturated, all incident priming edges are destroyed. Therefore, each priming edge always connects a saturated vertex with an unsaturated vertex. Moreover, if due to an augmentation a vertex becomes part of an

augmentation edge, all incident priming edges are destroyed as well. This is done such that augmentation edges remain disjoint after future augmentations.

The design of our algorithm satisfies a number of such invariants. The most important features of the algorithm are the following;

- Augmentation edges are vertex-disjoint.
- Each augmentation edge is incident with at least one saturated vertex.
- Each augmentation edge is incident with at most one priming edge.
- Each priming edge is incident with exactly one augmentation edge.
- Priming edges are only incident with saturated endpoints of augmentation edges, and have exactly one saturated endpoint.
- Each vertex that becomes saturated is incident with an augmentation edge directly after.
- Each saturated vertex that is incident with an augmentation edge before an augmentation will be incident with an augmentation edge directly after said augmentation.

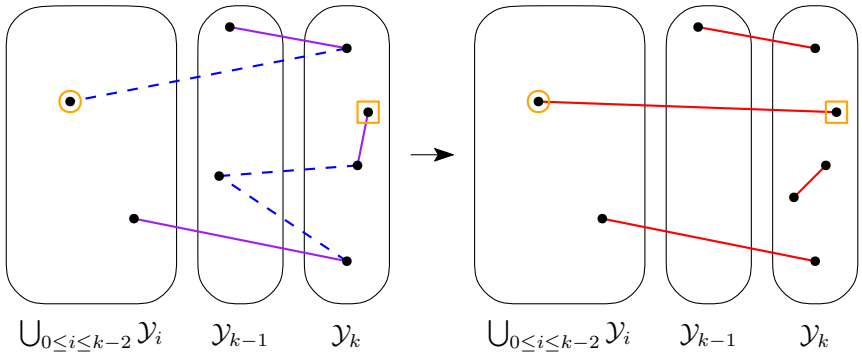

Figure 6.3: Illustration of augmentation along a degree-k augmentation edge and the resulting destruction of priming edges.

As mentioned, initially each vertex that becomes saturated is incident with an augmentation edge. Moreover, the algorithm as described thus far ensures that saturated vertices incident with an augmentation edge remain incident with an augmentation edge. However, this construction gives rise to issues near the end of the algorithm. Namely, as each augmentation edge is disjoint, the number of augmentation edges of degree at most $k-1$ is limited by the number of unsaturated vertices. However, more importantly, as priming edges cannot be incident with two augmentation edges, once the number of augmentation edges of degree at most $k-1$ equals the number of unsaturated vertices, no more augmentations can take place. Hence, once the ratio of the number of augmentation edges of degree at most $k-1$ over the number of unsaturated vertices becomes to high, we demote a number of such augmentation edges. The number of augmentation edges we demote is dependent on the ratio between the number of such augmentation edges, and the number of unsaturated vertices that are not part of an augmentation edge.

6.6.2 Formal description

Next, we describe the algorithm more formally by defining the strategy for each case based on the degree and augmentation edge status of vertex u_t. To that purpose we first define the following sets of vertices.

$\mathcal{Y}_i(t)$ for $0 \leq i \leq k-1$	All vertices $v \in V(H_t)$ such that $\deg_{H_t}(v) = i$.
$\mathcal{A}_i(t)$ for $0 \leq i \leq k-1$	All vertices $v \in V(H_t)$ incident with an unprimed degree-i augmentation edge and $\deg_{H_t}(v) = i$.
$\mathcal{A}'_i(t)$ for $0 \leq i \leq k-1$	All vertices $v \in V(H_t)$ incident with an unprimed degree-i augmentation edge and $\deg_{H_t}(v) = k$.
$\mathcal{A}_k(t)$	All vertices $v \in V(H_t)$ incident with an unprimed degree-k augmentation edge.
$\mathcal{P}_i(t)$ for $0 \leq i \leq k-1$	All vertices $v \in V(H_t)$ incident with a primed degree-i augmentation edge and $\deg_{H_t}(v) = i$.
$\mathcal{P}'_i(t)$ for $0 \leq i \leq k-1$	All vertices $v \in V(H_t)$ incident with a primed degree-i augmentation edge and $\deg_{H_t}(v) = k$.
$\mathcal{P}_k(t)$	All vertices $v \in V(H_t)$ incident with a primed degree-k augmentation edge.

Note that by the definition of degree-ℓ augmentation edges, $\mathcal{A}_0 = \mathcal{A}'_0 = \mathcal{P}_0 = \mathcal{P}'_0 = \emptyset$. We still define these variables to simplify notation in further expressions. We do not track the set of saturated vertices, as it may directly be derived from the sets of unsaturated vertices of specified degrees.

Event 1	If $\deg_{H_t}(u_t) \leq k-2$, and u_t is uncovered, we can directly add an edge to another unsaturated vertex. We choose v_t u.a.r. from $\left(\bigcup_{i=0}^{k-2} \mathcal{Y}_i(t)\right) \setminus N_{H_t}[u_t]$.
Event 2	If $\deg_{H_t}(u_t) \leq k-2$, and u_t is incident with an unprimed augmentation edge, we can again directly add an edge to another unsaturated vertex. Hence, we choose v_t u.a.r. from $\left(\bigcup_{i=0}^{k-2} \mathcal{Y}_i(t)\right) \setminus N_{H_t}[u_t]$.

Continued on next page

Event 3 If $\deg_{H_t}(u_t) \leq k-2$, and u_t is incident with a primed augmentation edge, we have two choices. Because the augmentation edge is primed, and the priming edge is incident with the saturated endpoint of the augmentation edge, we may choose to add an edge from u_t to another unsaturated vertex, and augment along the primed augmentation edge. We refer to this case as *Event 3.1*. For w the priming vertex of the augmentation edge, we then choose v_t u.a.r. in $\left(\bigcup_{i=0}^{k-1} \mathcal{Y}_i(t)\right) \setminus (N_{H_t}[u_t] \cup \{w\})$.
Alternatively, we may choose to directly add an edge to another unsaturated vertex. We refer to this case as *Event 3.2*, where we choose v_t u.a.r. from $\left(\bigcup_{i=0}^{k-2} \mathcal{Y}_i(t)\right) \setminus N_{H_t}[u_t]$. We choose for Event 3.1 with probability $\rho_{\deg_{H_t}(u)}$, and otherwise for Event 3.2.

Event 4 If $\deg_{H_t}(u_t) = k-1$, and u_t is uncovered, we create a degree-k augmentation edge by adding an edge to another vertex of degree $k-1$ that is not incident with any augmentation edge. That is, we choose v_t u.a.r. from $\mathcal{Y}_{k-1}(t) \setminus (\mathcal{A}_{k-1}(t) \cup \mathcal{P}_{k-1}(t))$.

Event 5 If $\deg_{H_t}(u_t) = k-1$, and u_t is incident with an unprimed augmentation edge, we create two degree-k augmentation edges by adding an edge to another unsaturated endpoint of a degree-$(k-1)$ augmentation edge. This gives two choices. Either we add an edge to the endpoint of an unprimed augmentation edge, or to the endpoint of a primed augmentation edge. We refer to these cases by *Event 5.1* and *Event 5.2* respectively. It may seem sensible to choose u.a.r. over all such vertices. However, this would introduce a singularity in the differential equations when applying Wormald's differential equation method. Hence, we assign a probability to each of the two events.
In case of Event 5.1, we choose v_t u.a.r. from $\mathcal{A}_{k-1}(t) \setminus N_{H_t}[u_t]$. We choose this case with probability σ_A.
In case of Event 5.2, we then choose v_t u.a.r. from $\mathcal{P}_{k-1}(t) \setminus N_{H_t}[u_t]$. We choose this case with the remaining probability $1 - \sigma_A$.

Continued on next page

Event 6 If $\deg_{H_t}(u_t) = k-1$, and u_t is incident with a primed augmentation edge, we have multiple options. Firstly, we can augment along the augmentation edge. We refer to this case as *Event 6.1*, and choose this option with probability ρ_{k-1}. We then choose v_t u.a.r. from $\left(\bigcup_{i=0}^{k-1} \mathcal{Y}_i(t)\right) \setminus (N_{H_t}[u_t] \cup \{w\})$, where w is the priming vertex of the augmentation edge.
Alternatively, we create two degree-k augmentation edges by adding an edge to another vertex of degree $k-1$ incident with an augmentation edge. This thus happens with probability $1-\rho_{k-1}$. In this case, like in Event 5, this edge may connect to a vertex incident with an unprimed or a primed augmentation edge, which we refer to by *Event 6.2.1* and *Event 6.2.2* respectively. We choose for Event 6.2.1 with probability σ_P and for Event 6.2.2 with remaining probability $1-\sigma_P$. In the case of Event 6.2.1, we choose v_t u.a.r. from $\mathcal{A}_{k-1}(t) \setminus N_{H_t}[u_t]$. Similarly, in the case of Event 6.2.2, we choose v_t u.a.r. from $\mathcal{P}_{k-1}(t) \setminus N_{H_t}[u_t]$.

Event 7 If $\deg_{H_t}(u_t) = k$, and u_t is incident with an unprimed degree-k augmentation edge, we add a priming edge by choosing v_t u.a.r. in $\bigcup_{i=0}^{k-1} (\mathcal{Y}_i(t) \setminus (\mathcal{A}_i(t) \cup \mathcal{P}_i(t))) \setminus N_{H_t}[u_t]$

Event 8 If $\deg_{H_t}(u_t) = k$, and u_t is incident with a primed degree-k augmentation edge, the strategy depends on whether u_t is the primed vertex. If u_t is the primed vertex, which we refer to as *Event 8.1*, we consider the round a failure round.
Otherwise, we augment along the augmentation edge, which we refer to as *Event 8.2*. In this case, we choose v_t u.a.r. from $\left(\bigcup_{i=0}^{k-1} (\mathcal{Y}_i(t) \setminus ((\mathcal{A}_i(t) \cup \mathcal{P}_i(t)))\right) \setminus (N_{H_t}[u_t] \cup \{w\})$ where w is the priming vertex for the augmentation edge.

Event 9 If $\deg_{H_t}(u_t) = k$, and u_t is incident with an unprimed augmentation edge of degree at most $k-1$, we add a priming edge by choosing v_t u.a.r. in $\bigcup_{i=0}^{k-1} (\mathcal{Y}_i(t) \setminus (\mathcal{A}_i(t) \cup \mathcal{P}_i(t))) \setminus N_{H_t}[u_t]$.

Event 10 If $\deg_{H_t}(u_t) = k$, and u_t is incident with a primed augmentation edge of degree at most $k-1$, we consider the round a failure round.

Event 11 If $\deg_{H_t}(u_t) = k$, and u_t is not incident with any augmentation edge, we consider the round a failure round.

If in any of these events the set where v_t is to be chosen from is empty, we consider the round a failure.

Finally, we formally define the demotion procedure. To avoid the previously described scenario of having each unsaturated vertex being incident with an augmentation edge, we probabilistically demote a number of such augmentation edges. Specifically, we demote each augmentation edge that is incident with an unsaturated vertex with probability

$\phi / \left|\bigcup_{i=0}^{k-1} \left(\mathcal{Y}_i(t) \setminus (\mathcal{A}_i(t) \cup \mathcal{P}_i(t))\right)\right|$. Observe that the denominator of this probability is exactly the number of uncovered vertices. Moreover, note that there is no point in demoting degree-k augmentation edges. The factor ϕ, which we refer to as the *demotion factor*, should be chosen sufficiently large to ensure that the expected number of demoted augmentation edges is not dominated by the expected number of new such augmentation edges when the number of remaining uncovered vertices approaches 0. In Section 6.8.1 we will consider explicit values for demotion factor ϕ.

6.7 Analysis

6.7.1 Random variables and their expected change

By design, the sets $\mathcal{Y}_i(t)$, $\mathcal{A}_i(t)$, $\mathcal{A}'_i(t)$, $\mathcal{P}_i(t)$, $\mathcal{P}'_i(t)$ for $0 \leq i \leq k-1$, and $\mathcal{A}_k(t)$ and $\mathcal{P}_k(t)$ are sufficient to completely describe the algorithm's process. Vertices within these sets are indistinguishable for the algorithm's purpose. Hence, for the analysis of the algorithm, it suffices to track the sizes of these sets.

Let $Y_i(t) = |\mathcal{Y}_i(t)|$, $A_i(t) = |\mathcal{A}_i(t)|$, and $P_i(t) = |\mathcal{P}_i(t)|$ for all $0 \leq i \leq k-1$. Additionally, let $A_k(t) = |\mathcal{A}_k(t)|/2$ and $P_k(t) = |\mathcal{P}_k(t)|/2$. As such, $A_k(t)$ and $P_k(t)$ track the number of unprimed and primed degree-k augmentation edges respectively, rather than the number of vertices incident with such edges. Because $|\mathcal{A}_i(t)| = |\mathcal{A}'_i(t)|$ and $|\mathcal{P}_i(t)| = |\mathcal{P}'_i(t)|$, this collection of variables suffices to describe the progress of the algorithm.

Initially, for $t = 0$, as the graph is edgeless, all variables except for $Y_0(0)$ are equal to 0. Naturally, $Y_0(0) = n$.

Using that the algorithm makes a number of choices u.a.r. we can then derive equations for the expected difference in value in a single round for each variable. These equations are included in Appendix B. Note that we have excluded $A_0(t)$ and $P_0(t)$, as both are equal to 0 and are unaffected by all events.

6.7.2 Differential equations

To analyse the equations for general n, we scale each of the variables accordingly. Let $y_i(s) = Y_i(sn)/n$ for $0 \leq i \leq k-1$. Similarly, let $a_i(s) = A_i(sn)/n$ and $p_i(s) = P_i(sn)/n$ for all $0 \leq i \leq k$.

By substituting these variables into the expected difference equations, and grouping these equations per variable, we obtain a set of differential equations. We have included these differential equations in Appendix C.

We claim that these differential equations accurately describe the progress of the variable using our strategy for $n \to \infty$. We will use Wormald's differential equation method to prove as much.

6.7.3 Applying Wormald's differential equation method

Let $D \subseteq \mathbb{R}^{3k+1}$ be the region defined by

$$\begin{aligned}\{(s, y_0, y_1, \ldots, y_{k-1}, a_1, a_2, \ldots, a_k, p_1, p_2, \ldots, p_k) \mid & -\epsilon_0 < s < 2k \\ & \wedge -\epsilon_0 < y_i < 1 + \epsilon_0 \text{ for all } 0 \leq i \leq k-1 \\ & \wedge -\epsilon_0 < a_i < 1/2 + \epsilon_0 \text{ for all } 1 \leq i \leq k \\ & \wedge -\epsilon_0 < p_i < 1/2 + \epsilon_0 \text{ for all } 1 \leq i \leq k \\ & \wedge \sum_{i=0}^{k-1} y_i > \epsilon_1 \\ & \wedge \sum_{i=0}^{k-2} y_i > \epsilon_2 \\ & \wedge \sum_{i=0}^{k-1} (y_i - a_i - p_i) > \epsilon_3\},\end{aligned}$$

where $\epsilon_0, \epsilon_1, \epsilon_2, \epsilon_3 > 0$. We will give explicit values for these variables in Section 6.7.4.

We then claim that our differential equations satisfy the three hypotheses for Wormald's differential equation method on region D. Firstly, we consider the boundedness hypothesis. We note that the direct change in any of the variables in a single round as a result of vertices u_t and v_t changing degree, and the resulting augmentations, is at most 2. There are two other factors that can result in variables changing in value, namely the destruction of priming edges and the demotion of augmentation edges. We note that in expectation, each uncovered vertex is incident with at most

$$\frac{\sum_{j=1}^{k} P_j(t)}{\sum_{j=0}^{k-1}(Y_j(t) - A_j(t) - P_j(t))} < \frac{(1+\epsilon_0)kn}{\epsilon_3 n} = \frac{(1+\epsilon_0)k}{\epsilon_3}$$

priming edges. Hence, using a Chernoff-Hoeffding bound for binomial variables, we find that the probability that the number of priming edges destroyed in a single round is greater than $2\log_2 n$ is at most $2^{-2\log_2 n} = n^{-2}$ for n sufficiently large such that $2\log_2 n > 2 \cdot 2e^{\frac{(1+\epsilon_0)k}{\epsilon_3}}$. Similarly, the expected number of augmentation edges to be demoted in a single round is

$$\begin{aligned}\frac{\phi \cdot \sum_{j=1}^{k-1}(A_j(t) + P_j(t))}{\sum_{j=0}^{k-1}(Y_j(t) - A_j(t) - P_j(t))} &< \frac{\phi \cdot (k-1) \cdot ((1/2+\epsilon_0)n + (1/2+\epsilon_0)n)}{\epsilon_3 n} \\ &= \frac{\phi \cdot (k-1) \cdot (1+2\epsilon_0)}{\epsilon_3}.\end{aligned}$$

Hence, again using the Chernoff-Hoeffding bound, the probability that the number of augmentation edges demoted in a single round is greater than $2\log_2 n$ is at most n^{-2} for n sufficiently large such that $2\log_2 n > 2e^{\frac{\phi \cdot (k-1) \cdot (1+2\epsilon_0)}{\epsilon_3}}$. Thus, for $\beta(n) = 2 + 4\log_2 n = \theta(\log n)$ and $\gamma(n) = n^{-2}$, the boundedness hypothesis is satisfied.

Next, we observe that by the definition of the differential equations, the trend hypothesis holds for $\lambda_1(n) = O(1/n)$.

Finally, for the Lipschitz hypothesis, we observe that all of the right-hand sides of the differential equations are rational functions. Moreover, by the boundary definitions of region D, the denominators in each of the terms of the differential equations are bounded from below by strictly positive constants, and the numerators are bounded from both above and below. Hence, each of the right-hand sides of the differential equations are bounded, and therefore, the functions $y_0(t), y_1(t), \ldots, y_{k-1}(t), a_1(t), a_2(t), \ldots, a_k(t), p_1(t), p_2(t), \ldots, p_k(t)$ are Lipschitz continuous on D, as desired.

Because all three of the hypotheses are met, Wormald's differential equation method applies. The system of differential equations thus has a unique solution in D that goes through the initial point $(0, 1, 0, 0, \ldots 0)$ and extends arbitrarily close to the boundary of D. We will denote the functions making up this solution by $y_i^\star(s)$ for $0 \leq i \leq k-1$, $a_i^\star(s)$ for $1 \leq i \leq k$, and $p_i^\star(s)$ for $1 \leq i \leq k$.

Moreover, for $\lambda(n) = \theta(n^{-1/4})$, with probability

$$\begin{aligned} 1 - O\left(n\gamma(n) + \frac{\beta(n)}{\gamma(n)} \exp\left(-\frac{n\lambda(n)^3}{\beta(n)^3}\right)\right) &= 1 - O\left(n \cdot n^{-2} + \frac{\log n}{n^{-1/4}} \exp\left(-\frac{n(n^{-1/4})^3}{(\log n)^3}\right)\right) \\ &= 1 - O\left(\frac{1}{n} + n^{1/4} \log n \exp\left(-\frac{n^{1/4}}{(\log n)^3}\right)\right), \end{aligned}$$

and therefore a.a.s.,

$$\begin{aligned} Y_i(t) &= n y_i^\star(t/n) + O(n^{3/4}) && \text{for all } 0 \leq i \leq k-1 \\ A_i(t) &= n a_i^\star(t/n) + O(n^{3/4}) && \text{for all } 1 \leq i \leq k \\ P_i(t) &= n p_i^\star(t/n) + O(n^{3/4}) && \text{for all } 1 \leq i \leq k. \end{aligned}$$

6.7.4 Boundaries and singularities

To determine the running time of the algorithm, we are interested in the time at which the solution equations get arbitrarily close to the boundary given by the inequality $\sum_{j=0}^{k-1} y_j > \epsilon_1$. To this purpose, we first need to establish that the solutions do not hit another border of region D first.

Naturally, as we are only interested in a positive number of rounds, the boundary $-\epsilon_0 < s$ does not pose an issue. Furthermore, because our algorithm is by its design at least as efficient as running the perfect matching algorithm k times, by Lemma 6.6, the solution functions will hit the desired boundary before hitting the boundary given by $s < 2k$. For the boundaries of D given by the inequalities $-\epsilon_0 < y_i < 1 + \epsilon_0$ for $0 \leq i \leq k-1$, $-\epsilon_0 < a_i < 1/2 + \epsilon_0$ for $1 \leq i \leq k$, and $-\epsilon_0 < p_i < 1/2 + \epsilon_0$ for $1 \leq i \leq k$ it follows that they are not hit before the desired boundary by the definitions of the corresponding variables.

This leaves the two boundaries given by $\sum_{i=0}^{k-2} y_i > \epsilon_2$ and $\sum_{i=0}^{k-1}(y_i - a_i - p_i) > \epsilon_3$. These two boundaries are included in the definition of D to avoid singularities in the differential equations. We will show that by choosing appropriate values for ϵ_2 and ϵ_3 relative to ϵ_1, we can ensure that these boundaries are not hit. We first observe that $\epsilon_2, \epsilon_3 < \epsilon_1$ must hold, as otherwise the desired boundary is redundant.

Assume that $2\epsilon_3 < \epsilon_0$. Suppose that $\sum_{i=0}^{k-1}(y_i(s) - a_i(s) - p_i(s)) < 2\epsilon_3$ and $\sum_{i=0}^{k-1} y_i(s) > \epsilon_1$ for all $s > 0$ in D. It then follows that $\sum_{i=0}^{k-1}(a_i(s) + p_i(s)) > \epsilon_1 - 2\epsilon_3$. However, we note that if we additionally assume that $\sum_{i=0}^{k-2} y_i(s) > \epsilon_2$, it follows that

$$\lim_{\left(\sum_{i=0}^{k-1}(y_i(s)-a_i(s)-p_i(s))\right)\downarrow 0} \left(\sum_{i=0}^{k-1}(a_i'(s) + p_i'(s))\right)$$
$$\leq c + \lim_{\left(\sum_{i=0}^{k-1}(y_i(s)-a_i(s)-p_i(s))\right)\downarrow 0} \left(-\frac{\phi \cdot \sum_{i=0}^{k-1}(a_i(s) + p_i(s))}{\sum_{i=0}^{k-1}(y_i(s) - a_i(s) - p_i(s))}\right),$$

where c is a constant dependent on ϵ_1 and ϵ_2. Then, for $\phi > 0$,

$$\lim_{\left(\sum_{i=0}^{k-1}(y_i(s)-a_i(s)-p_i(s))\right)\downarrow 0} \left(\sum_{i=0}^{k-1}(a_i'(s) + p_i'(s))\right) = -\infty.$$

Thus, for ϵ_3 sufficiently small, $\sum_{i=0}^{k-1}(a_i(s)+p_i(s)) > \epsilon_1 - 2\epsilon_3$ cannot hold. Hence, the desired boundary must be hit before the boundary given by $\sum_{i=0}^{k-1}(y_i(s) - a_i(s) - p_i(s)) > \epsilon_3$.

Next we consider the boundary given by $\sum_{i=0}^{k-2} y_i(s) > \epsilon_2$. Assume that $\sum_{i=0}^{k-1} y_i(s) > \epsilon_1$ and $\sum_{i=0}^{k-1}(y_i(s) - a_i(s) - p_i(s)) > \epsilon_3$ hold. Let $c > 0$ be a constant. We note that by the design of the algorithm, $\sum_{i=0}^{k-2} y_i(s)$ is strictly decreasing. Thus, because $\sum_{i=0}^{k-2} y_i(0) = 1$, and as by the design of the algorithm $\sum_{i=0}^{k-2} y_i(s)$ tends to 0, there exists a unique time $\hat{s} > 0$ such that $\sum_{i=0}^{k-2} y_i(\hat{s}) = c\epsilon_1$. Then, because $0 \leq y_{k-1}(s) \leq 1$ and $c\epsilon_1 \leq \sum_{i=0}^{k-2} y_i \leq 1$, the maximum ratio $\hat{r} = \max_{0\leq s\leq \hat{s}} \frac{y_{k-1}(s)}{\sum_{i=0}^{k-2} y_i(s)}$ is bounded and well-defined. Let $\epsilon_2 = \frac{c\epsilon_1}{2\hat{r}}$.

We then consider the derivative of $(y_{k-1}(s))/(\sum_{i=0}^{k-2} y_i(s))$ for $s > \hat{s}$. By the quotient rule,

$$\frac{d}{ds}\frac{y_{k-1}(s)}{\sum_{i=0}^{k-2} y_i(s)} = \frac{y_{k-1}'(s) \cdot \left(\sum_{i=0}^{k-2} y_i(s)\right) + y_{k-1}(s) \cdot \left(\sum_{i=0}^{k-2} y_i'(s)\right)}{\left(\sum_{i=0}^{k-2} y_i(s)\right)^2}.$$

We observe that for c sufficiently small, $y_{k-1}'(s) < 0$ for $s > \hat{s}$. But then, as also $\sum_{i=0}^{k-2} y_i(s)$ is strictly decreasing, it follows that

$$\frac{d}{ds}\frac{y_{k-1}(s)}{\sum_{i=0}^{k-2} y_i(s)} < 0,$$

and hence $(y_{k-1}(s))/(\sum_{i=0}^{k-2} y_i(s)) \leq \hat{r}$ for all $s \geq 0$ in D. Therefore, our choice of ϵ_2 suffices to ensure that the desired boundary is hit.

Finally, the case that the solution approaches both of the boundaries defined by $\sum_{i=0}^{k-1}(y_i - a_i - p_i) > \epsilon_3$ and $\sum_{i=0}^{k-2} y_i > \epsilon_2$ simultaneously before approaching the desired boundary follows from combining the arguments used for the two boundaries individually.

Then, by setting $\epsilon_1 \leq \frac{10^{-14}}{2k^2}$, and setting the other constants ϵ_2 and ϵ_3 accordingly, the solutions extend to the point where $\sum_{i=0}^{k-1} y_i(s) = 2\epsilon_1$. Then, by the Clean-up Lemma (Lemma 6.2), we obtain the following result.

Theorem 6.7. *For each $k \geq 2$, $\tau_{\mathcal{F}_k} \leq \beta + 10^{-5}$ where β is derived from a system of differential equations.*

6.8 Numerical results

Due to the complexity of the system of differential equations, we do not have analytical solutions. Instead, we use Maple to obtain numerical solutions. Specifically, we call a seventh-eighth order continuous Runge-Kutta method, which allows for both fast computation and error control. See Appendix D for the code used.

6.8.1 Parameter tuning

The algorithm as described in Section 6.6.2 contains constants for which the exact values were left open. It concerns the parameters ρ_i for all $0 \leq i \leq k-1$, σ_A, σ_B, and ϕ. We first experimentally tune these parameters.

Interestingly, we consistently obtain the best results with $\rho_i = 0$ for $0 \leq i \leq k-1$ and $\sigma_A = \sigma_B = 0$. This is not entirely unexpected. The parameters ρ_i define the probability of choosing to augment along an augmentation edge when there is also the option of directly adding an edge between two unsaturated vertices. An augmentation along an augmentation edge is inherently less efficient than directly adding an edge in such a fashion. In some cases it may be more effective to augment to create new augmentation edges, but clearly this potential benefit is outweighed by the overall cost in efficiency. For σ_A and σ_B, the setting of $\sigma_A = \sigma_B = 0$ most closely mimics the behaviour of choosing v_t u.a.r. from all degree-$(k-1)$ vertices incident with an augmentation edge in events 5 and 6.2, which was the natural choice we avoided to not introduce a singularity in the differential equations.

Thus, only demotion factor ϕ remains to be optimised. Due to its important role in regulating the number of augmentation edges near the end of the process, this parameter has a strong effect on the final solution. By extensive experimentation, we determined the optimal first two significant digits for all $2 \leq k \leq 20$. We then use a linear regression fit on the relation between k and $1/\phi$ to determine likely good values for ϕ for larger k. Table 6.1 includes the values for ϕ we use to generate the bounds for $\tau_{\mathcal{F}_k}$ in Section 6.8.2.

The trajectories of the variables as shown in Figure 6.4 suggest further explanations for the limited effect of some of the discussed parameters. Namely, they show that there are only few augmentation edges of degree less than $k-1$. As such, the parameters and features

k	ϕ	k	ϕ	k	ϕ
2	0.70	10	0.39	18	0.27
3	0.64	11	0.37	19	0.25
4	0.59	12	0.35	20	0.24
5	0.55	13	0.34	50	0.12
6	0.51	14	0.32	100	0.06
7	0.48	15	0.30	250	0.03
8	0.45	16	0.29	500	0.01
9	0.42	17	0.28		

Table 6.1: Values used for ϕ.

of the algorithm controlling these lower-degree augmentation edges are limited in effect as well.

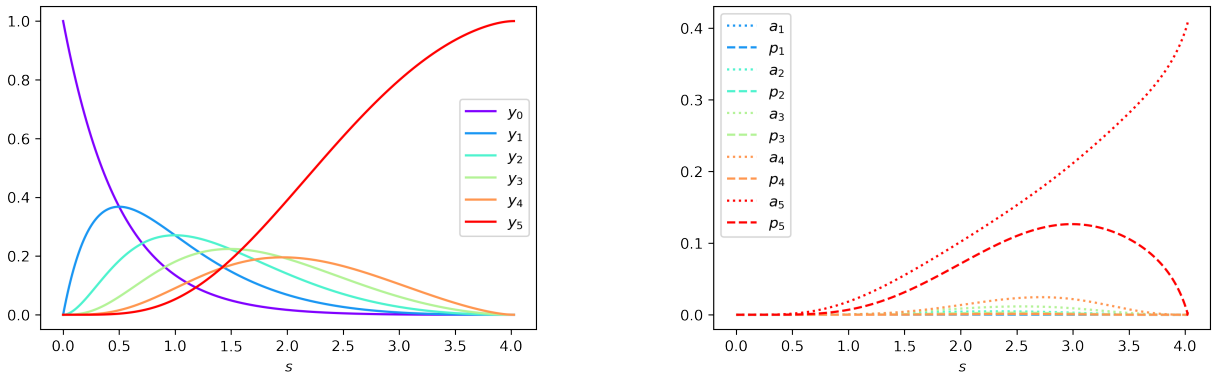

Figure 6.4: Trajectories of the variables for $k = 5$.

6.8.2 Results for small k

In Table 6.2 we give an overview of the performance of the upper bound given by our strategy using the values for the parameters as determined in Section 6.8.1 as compared to the other known upper bounds for $\tau_{\mathcal{F}_k}$. We first observe that for $k \geq 3$, the bound given by our Theorem 6.7 outperforms all other known bounds we know of, including running the perfect matching algorithm by Gao, MacRury and Prałat [11] k times. Interestingly however, for $k = 2$, the bound for τ_{HAM} by Frieze and Sorkin [8] is significantly better. We note that their algorithm (and the algorithm it is based upon by Gao, MacRury and Prałat [12]) allows for each edge in H_t to be an augmentation edge. Moreover, when a priming edge is added, all incident augmentation edges are considered primed. These notions are not easily analysable when extended to larger k as discussed in Section 6.5.

k	Theorem 6.7	Lemma 6.6	[8, Theorem 1.1]	[3, Theorem 1.3]
2	2.01382	2.41049	1.85	3.0
3	2.70352	3.61573	-	4.5
4	3.37106	4.82097	-	6.0
5	4.02486	6.02621	-	7.5
6	4.66894	7.23145	-	9.0
7	5.30562	8.43669	-	10.5
8	5.93640	9.64193	-	12.0
9	6.56228	10.84717	-	13.5
10	7.18403	12.05241	-	15.0
11	7.80218	13.25765	-	16.5
12	8.41720	14.46289	-	18.0
13	9.02943	15.66813	-	19.5
14	9.63914	16.87337	-	21.0
15	10.24659	18.07861	-	22.5
16	10.85195	19.28385	-	24.0
17	11.45542	20.48909	-	25.5
18	12.05714	21.69433	-	27.0
19	12.65721	22.89957	-	28.5
20	13.25576	24.10481	-	30.0
50	30.78334	60.26201	-	75.0
100	59.14394	120.52401	-	150.0
250	142.58268	301.31001	-	375.0
500	279.07176	602.62001	-	750.0

Table 6.2: Comparison of upper bounds on $\tau_{\mathcal{F}_k}$

6.9 Possible improvements

We are confident that our bounds are not tight. Additionally, we do not think that the strategy and analysis used can be tweaked sufficiently to obtain tight bounds for $\tau_{\mathcal{F}_k}$. Nevertheless, there are a number of potential improvements to our algorithm that we discuss here.

It makes intuitive sense to prioritise increasing the degrees of vertices of low degree over increasing the degree of vertices that are already almost saturated. One could consider a phased approach where it each phase ensures that the minimum degree of the graph is raised, similar to the k-min process. Alternatively, one could increase the probability of choosing a low-degree vertex for v_t by using a weighted distribution over the unsaturated vertices where the weight is inversely proportional to the degree of the vertex. Both of these suggested approaches would be difficult to analyse with Wormald's differential equation method.

As found in Section 6.8.1, most parameters in our algorithm are simply set to 0 for the best results. Potentially, the features of the algorithm these parameters control only have a positive impact during a small portion of the process, which is outweighed by the negative impact over the remainder of the process. Should this be the case, better results could potentially be obtained by making the parameters time-dependent. That is, the parameters would be functions of time s rather than constants. This would allow for much finer control of when to apply their effects. This could be especially relevant for the demotion factor, as demotion of augmentation edges is only required near the end of the process.

The main limitations of our method are the requirements that augmentation edges are vertex-disjoint and that each priming edge is associated with only one augmentation edge. While it is straightforward to design strategies that avoid these limitations, their analysis proves difficult, as discussed in Section 6.5.

6.10 Large k

As previously noted, Ben-Eliezer, Gishboliner, Hefetz and Krivelevich [3] showed that $\tau_{\mathcal{F}_k} \leq k/2 + o(k)$ for $k = \omega(\log(n))$. We note that as a k-factor on n vertices contains $kn/2$ edges, and as a single edge is added in each round of the semi-random graph process, this bound is asymptotically tight. It is natural to consider whether the same asymptotic behaviour occurs for $k \to \infty$ independent of n. That is, whether $\tau_{\mathcal{F}_k} = k/2 + o(k)$. Such behaviour seems to be suggested by the results of Theorem 6.7, as included in Table 6.2. The following result confirms this.

Lemma 6.8. $\tau_{\mathcal{F}_k} = \frac{k}{2} + o(k)$.

Proof. Consider the Erdős-Rényi random graph $G(n,p)$ on n vertices where each potential edge is included with probability $p = kn/\left(2 \cdot \binom{n}{2}\right)$. By a result of Riordan and Selby [17, Theorem 2.1], using $b = 1/k^3$, and as $k < n$, it follows that a.a.s.

$$\begin{aligned}
\Delta(G(n,p)) &\leq pn + \frac{1}{k^3}\sqrt{np(1-p)} \\
&= \frac{kn}{n-1} + \frac{1}{k^3}\sqrt{\frac{kn}{n-1}\left(1 - \frac{k}{n-1}\right)} \\
&= k + \mathcal{O}\left(\frac{\sqrt{k}}{k^3}\right).
\end{aligned}$$

Similarly, a.a.s.

$$\begin{aligned}\delta(G(n,p)) &\geq 1-\left((1-p)n+\frac{1}{k^3}\sqrt{np(1-p)}\right)\\ &= \frac{kn}{n-1}-\frac{1}{k^3}\sqrt{\frac{kn}{n-1}\left(1-\frac{k}{n-1}\right)}\\ &= k-\mathcal{O}\left(\frac{\sqrt{k}}{k^3}\right).\end{aligned}$$

Let $G(n,m)$ be the Erdős-Rényi random graph where m edges are chosen uniformly at random. Then, for $m=\binom{n}{2}p=kn/2$, a.a.s. it also holds that $\Delta(G(n,m))\leq k+\mathcal{O}(\sqrt{k}/k^3)$ and $\delta(G(n,m))\geq k-\mathcal{O}(\sqrt{k}/k^3)$.

Moreover, by [4, Proposition 1.1], we can obtain a graph G_t in the semi-random graph process in $(1+o(1))\cdot m$ rounds such that there exists a spanning subgraph $H_t\subseteq G_t$ such that $H_t\sim G(n,m)$. Let H_t be this subgraph for $m=kn/2$. Thus, a.a.s. $\Delta(H_t)\leq k+\mathcal{O}(\sqrt{k}/k^3)$ and $\delta(H_t)\geq k-\mathcal{O}(\sqrt{k}/k^3)$.

Then let subgraph H_t' be obtained from H_t by removing edges incident with vertices of degree greater than k in H_t until each vertex has degree at most k. We note that a.a.s.

$$\Lambda_k(H_t')\leq 2\cdot n\cdot\mathcal{O}\left(\frac{\sqrt{k}}{k^3}\right)=\mathcal{O}\left(n\cdot\frac{\sqrt{k}}{k^3}\right).$$

Hence, by Lemma 6.2, for k sufficiently large, there exists a strategy to construct a k-factor from H_t' a.a.s. in $10^{-5}\cdot n$ rounds. There thus exists a strategy to construct a k-factor a.a.s. in $(1+o(1))\cdot m+10^{-5}\cdot n=kn/2+o(kn)$ rounds. Therefore, $\tau_{\mathcal{F}_k}\leq k/2+o(k)$. Note that as a k-factor contains exactly $kn/2$ edges, this is tight, as desired. □

References

[1] Yossi Azar, Andrei Z. Broder, Anna R. Karlin, and Eli Upfal. Balanced allocations. In *Proceedings of the twenty-sixth annual ACM symposium on theory of computing*, pages 593–602, 1994.

[2] Natalie C. Behague, Trent G. Marbach, Paweł Prałat, and Andrzej Rucinski. Subgraph Games in the Semi-Random Graph Process and Its Generalization to Hypergraphs. *arXiv preprint arXiv:2105.07034*, 2021.

[3] Omri Ben-Eliezer, Lior Gishboliner, Dan Hefetz, and Michael Krivelevich. Very fast construction of bounded-degree spanning graphs via the semi-random graph process. *Random Structures & Algorithms*, 57(4):892–919, 2020.

[4] Omri Ben-Eliezer, Dan Hefetz, Gal Kronenberg, Olaf Parczyk, Clara Shikhelman, and Miloš Stojaković. Semi-random graph process. *Random Structures & Algorithms*, 56(3):648–675, 5 2020.

[5] Tom Bohman and Alan Frieze. Avoiding a giant component. *Random Structures & Algorithms*, 19(1):75–85, 2001.

[6] Devdatt P. Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 6 2009.

[7] Paul Erdős and Alfréd Rényi. On random graphs i. *Publicationes Mathematicae Debrecen*, 6:290–297, 1959.

[8] Alan Frieze and Gregory B. Sorkin. Hamilton cycles in a semi-random graph model, 2022.

[9] Tibor Gallai. Elementare Relationen bezuglich der Glieder und trennenden Punkte von Graphen. *A Magyar Tudományos Akadémia Matematikai Kutató Intézetének közleményei*, 9:235–236, 1964.

[10] Pu Gao, Bogumił Kamiński, Calum MacRury, and Paweł Prałat. Hamilton cycles in the semi-random graph process. *European Journal of Combinatorics*, 99:103423, 2022.

[11] Pu Gao, Calum MacRury, and Paweł Prałat. Perfect Matchings in the Semi-random Graph Process. *arXiv preprint arXiv:2105.13455*, 2021.

[12] Pu Gao, Calum MacRury, and Paweł Prałat. A Fully Adaptive Strategy for Hamiltonian Cycles in the Semi-Random Graph Process, 2022.

[13] Frank Harary and Geert Prins. The block-cutpoint-tree of a graph. *Publicationes Mathematicae Debrecen*, 13:103–107, 1966.

[14] Mihyun Kang, Youngmee Koh, Sangwook Ree, and Tomasz Łuczak. The connectivity threshold for the min-degree random graph process. *Random Structures and Algorithms*, 29(1):105–120, 8 2006.

[15] Dénes Kőnig. *Theorie der Endlichen und Unendlichen Graphen: Kombinatorische Topologie der Streckenkomplexe*. Akademische Verlagsgesellschaft, Leipzig, 1936.

[16] Michael Mitzenmacher. The power of two choices in randomized load balancing. *IEEE Transactions on Parallel and Distributed Systems*, 12(10):1094–1104, 2001.

[17] Oliver Riordan and Alex Selby. The maximum degree of a random graph. *Combinatorics, Probability and Computing*, 9(6):549–572, 2000.

[18] Nicholas C. Wormald. Differential Equations for Random Processes and Random Graphs. *The Annals of Applied Probability*, 5(4):1217–1235, 1995.

[19] Nicholas C. Wormald. The differential equation method for random graph processes and greedy algorithms. *Lectures on approximation and randomized algorithms*, 73:73–155, 1999.

APPENDICES

Appendix A

Wormald's differential equation method theorem

Theorem A.1 (Wormald's differential equation method, [19, Theorem 5.1])**.** *Let $\mathfrak{H}_t$ denote the set of all possible histories $\mathcal{H}_t$. Then, for all $1 \leq l \leq a$ where a is fixed, let $y_l : \mathfrak{H}_t \to \mathbb{R}$ and $f_l : \mathbb{R}^{a+1}$ be functions such that for some constant C_0 and all $1 \leq l \leq a$, $|y_l(\mathcal{H}_t)| < C_0 n$ for each $\mathcal{H}_t \in \mathfrak{H}_t$ for all n. Let $Y_l(t)$ for $1 \leq l \leq a$ denote the random counterpart of y_l, that is, for a single history $\mathcal{H}_t$. Additionally, let $D \subseteq \mathbb{R}^{a+1}$ be a bounded connected open set containing the closure of the possible initial points*

$$\{(0, z_1, z_2, \ldots, z_a) \,|\, \mathbb{P}\left(Y_l(0) = z_l n, 1 \leq l \leq a\right) > 0 \text{ for some } n \geq 1\}.$$

Let stopping time T_D be the minimum time t such that $(t/n, Y_1(t)/n, Y_2(t)/n, \ldots, Y_a(t)/n) \notin D$. Moreover, suppose that the following three hypotheses are satisfied.

(i) (Boundedness hypothesis.) For some functions $\beta(n) \geq 1$ and $\gamma(n)$,

$$\mathbb{P}\left(\max_{1 \leq l \leq a} |Y_l(t+1) - Y_l(t)| \leq \beta(n) \;\mid\; \mathcal{H}_t\right) \geq 1 - \gamma(n)$$

for all $t < T_D$.

(ii) (Trend hypothesis.) For some function $\lambda_1(n) = o(1)$, and for all $1 \leq l \leq a$,

$$|\mathbb{E}[Y_l(t+1) - Y_l(t) \;\mid\; \mathcal{H}_t] - f_l(t/n, Y_1(t)/n, Y_2(t)/n, \ldots, Y_a(t)/n| \leq \lambda(n)$$

for all $t < T_D$.

(iii) (Lipschitz hypothesis.) Each function f_l is continuous and satisfies a Lipschitz condition on the set

$$D \cap \{(t, z_1, z_2, \ldots, z_a) \,|\, t \geq 0\}.$$

Then, the following two statements are true.

(i) For an initial point $(0, \hat{z}_1, \hat{z}_2, \ldots, \hat{z}_a) \in D$*, the system of differential equations*

$$\begin{aligned}
\frac{dz_1}{dx} &= f_1(x, z_1, z_2, \ldots, z_a) \\
\frac{dz_2}{dx} &= f_2(x, z_1, z_2, \ldots, z_a) \\
&\vdots \\
\frac{dz_a}{dx} &= f_a(x, z_1, z_2, \ldots, z_a)
\end{aligned}$$

has a unique solution in D *for* $z_l : \mathbb{R} \to \mathbb{R}$ *for all* $1 \leq l \leq a$ *such that* $z_l(0) = \hat{z}_l$*, and which extends to points arbitrarily close to the boundary of* D*.*

(ii) Let $\lambda(n) = o(1)$ *be a function such that* $\lambda(n) > \lambda_1(n) + C_0 n\gamma$*. For* C *a sufficiently large constant, it holds uniformly for* $0 \leq t \leq \sigma(n)n$ *that with probability* $1 - O(n\gamma(n) + \frac{\beta(n)}{\lambda(n)} \exp(-\frac{n\lambda(n)^3}{\beta(n)^3})$*,*

$$Y_l(t) = nz_l(t/n) + O(\lambda(n)n)$$

for all $1 \leq l \leq a$*, where* $z_l(x)$ *is the solution in* (i) *with initial point* $(0, Y_1(0)/n, Y_2(0)/n, \ldots, Y_a(0)/n)$*, and function* $\sigma(n)$ *is the supremum of those points* x *to which the solution can be extended before reaching within* ℓ^∞*-distance* $C\lambda(n)$ *of the boundary of* D*.*

Appendix B

Expected change in variables for the k-factor algorithm

We will use Y_i' as shorthand notation for $\mathbb{E}\left[Y_i(t+1) - Y_i(t) \mid \mathcal{H}_t\right]$. Moreover, in these tables, we will use Y_i to denote $Y_i(t)$, and analogously for A_i and P_i.

Event 1	
Y'_0	$-\frac{Y_0-A_0-P_0}{n} - \frac{\sum_{j=0}^{k-2}\left(Y_j-A_j-P_j\right)}{n} \cdot \frac{Y_0}{\sum_{j=0}^{k-2} Y_j} + O(1/n)$
Y'_i for $1 \le i \le k-2$	$-\frac{Y_i-A_i-P_i}{n} - \frac{\sum_{j=0}^{k-2}\left(Y_j-A_j-P_j\right)}{n} \cdot \frac{Y_i}{\sum_{j=0}^{k-2} Y_j} + \frac{Y_{i-1}-A_{i-1}-P_{i-1}}{n} + \frac{\sum_{j=0}^{k-2}\left(Y_j-A_j-P_j\right)}{n} \cdot \frac{Y_{i-1}}{\sum_{j=0}^{k-2} Y_j} + O(1/n)$
Y'_{k-1}	$\frac{Y_{k-2}-A_{k-2}-P_{k-2}}{n} + \frac{\sum_{j=0}^{k-2}\left(Y_j-A_j-P_j\right)}{n} \cdot \frac{Y_{k-2}}{\sum_{j=0}^{k-2} Y_j} + O(1/n)$
A'_i for $1 \le i \le k-2$	$-\frac{\sum_{j=0}^{k-2}\left(Y_j-A_j-P_j\right)}{n} \cdot \frac{A_i}{\sum_{j=0}^{k-2} Y_j} + \frac{\sum_{j=0}^{k-2}\left(Y_j-A_j-P_j\right)}{n} \cdot \frac{A_{i-1}}{\sum_{j=0}^{k-2} Y_j} + O(1/n)$
A'_{k-1}	$\frac{\sum_{j=0}^{k-2}\left(Y_j-A_j-P_j\right)}{n} \cdot \frac{A_{k-2}}{\sum_{j=0}^{k-2} Y_j} + O(1/n)$
A'_k	0
P'_i for $1 \le i \le k-2$	$-\frac{\sum_{j=0}^{k-2}\left(Y_j-A_j-P_j\right)}{n} \cdot \frac{P_i}{\sum_{j=0}^{k-2} Y_j} + \frac{\sum_{j=0}^{k-2}\left(Y_j-A_j-P_j\right)}{n} \cdot \frac{P_{i-1}}{\sum_{j=0}^{k-2} Y_j} + O(1/n)$
P'_{k-1}	$\frac{\sum_{j=0}^{k-2}\left(Y_j-A_j-P_j\right)}{n} \cdot \frac{P_{k-2}}{\sum_{j=0}^{k-2} Y_j} + O(1/n)$
P'_k	0

Legend	
	Contribution to expected change due to increase in degree of vertex u_t Contribution to expected change due to increase in degree of vertex v_t

Event 2	
Y'_0	$-\frac{A_0}{n} - \frac{\sum_{j=0}^{k-2} A_j}{n} \cdot \frac{Y_0}{\sum_{j=0}^{k-2} Y_j} + O(1/n)$
Y'_i for $1 \le i \le k-2$	$-\frac{A_i}{n} - \frac{\sum_{j=0}^{k-2} A_j}{n} \cdot \frac{Y_i}{\sum_{j=0}^{k-2} Y_j} + \frac{A_{i-1}}{n} + \frac{\sum_{j=0}^{k-2} A_j}{n} \cdot \frac{Y_{i-1}}{\sum_{j=0}^{k-2} Y_j} + O(1/n)$
Y'_{k-1}	$\frac{A_{k-2}}{n} + \frac{\sum_{j=0}^{k-2} A_j}{n} \cdot \frac{Y_{k-2}}{\sum_{j=0}^{k-2} Y_j} + O(1/n)$
A'_i for $1 \le i \le k-2$	$-\frac{A_i}{n} - \frac{\sum_{j=0}^{k-2} A_j}{n} \cdot \frac{A_i}{\sum_{j=0}^{k-2} Y_j} + \frac{A_{i-1}}{n} + \frac{\sum_{j=0}^{k-2} A_j}{n} \cdot \frac{A_{i-1}}{\sum_{j=0}^{k-2} Y_j} + O(1/n)$
A'_{k-1}	$\frac{A_{k-2}}{n} + \frac{\sum_{j=0}^{k-2} A_j}{n} \cdot \frac{A_{k-2}}{\sum_{j=0}^{k-2} Y_j} + O(1/n)$
A'_k	0
P'_i for $1 \le i \le k-2$	$-\frac{\sum_{j=0}^{k-2} A_j}{n} \cdot \frac{P_i}{\sum_{j=0}^{k-2} Y_j} + \frac{\sum_{j=0}^{k-2} A_j}{n} \cdot \frac{P_{i-1}}{\sum_{j=0}^{k-2} Y_j} + O(1/n)$
P'_{k-1}	$\frac{\sum_{j=0}^{k-2} A_j}{n} \cdot \frac{P_{k-2}}{\sum_{j=0}^{k-2} Y_j} + O(1/n)$
P'_k	0
Legend	Contribution to expected change due to increase in degree of vertex u_t Contribution to expected change due to increase in degree of vertex v_t

Event 3.1	
Y_0'	$-\frac{\sum_{j=1}^{k-2}\left(\rho_j \cdot P_j\right)}{n} \cdot \frac{Y_0}{\sum_{j=0}^{k-1} Y_j} - \frac{\sum_{j=1}^{k-2}\left(\rho_j \cdot P_j\right)}{n} \cdot \frac{Y_0 - A_0 - P_0}{\sum_{j=0}^{k-1}\left(Y_j - A_j - P_j\right)} + O(1/n)$
Y_i' for $1 \le i \le k-2$	$-\frac{\sum_{j=1}^{k-2}\left(\rho_j \cdot P_j\right)}{n} \cdot \frac{Y_i}{\sum_{j=0}^{k-1} Y_j} + \frac{\sum_{j=1}^{k-2}\left(\rho_j \cdot P_j\right)}{n} \cdot \frac{Y_{i-1}}{\sum_{j=0}^{k-1} Y_j} - \frac{\sum_{j=1}^{k-2}\left(\rho_j \cdot P_j\right)}{n} \cdot \frac{Y_i - A_i - P_i}{\sum_{j=0}^{k-1}\left(Y_j - A_j - P_j\right)} + \frac{\sum_{j=1}^{k-2}\left(\rho_j \cdot P_j\right)}{n} \cdot \frac{Y_{i-1} - A_{i-1} - P_{i-1}}{\sum_{j=0}^{k-1}\left(Y_j - A_j - P_j\right)} + O(1/n)$
Y_{k-1}'	$-\frac{\sum_{j=1}^{k-2}\left(\rho_j \cdot P_j\right)}{n} \cdot \frac{Y_{k-1}}{\sum_{j=0}^{k-1} Y_j} + \frac{\sum_{j=1}^{k-2}\left(\rho_j \cdot P_j\right)}{n} \cdot \frac{Y_{k-2}}{\sum_{j=0}^{k-1} Y_j} - \frac{\sum_{j=1}^{k-2}\left(\rho_j \cdot P_j\right)}{n} \cdot \frac{Y_{k-1} - A_{k-1} - P_{k-1}}{\sum_{j=0}^{k-1}\left(Y_j - A_j - P_j\right)} + \frac{\sum_{j=1}^{k-2}\left(\rho_j \cdot P_j\right)}{n} \cdot \frac{Y_{k-2} - A_{k-2} - P_{k-2}}{\sum_{j=0}^{k-1}\left(Y_j - A_j - P_j\right)} + O(1/n)$
A_i' for $1 \le i \le k-2$	$-\frac{\sum_{j=1}^{k-2}\left(\rho_j \cdot P_j\right)}{n} \cdot \frac{A_i}{\sum_{j=0}^{k-1} Y_j} + \frac{\sum_{j=1}^{k-2}\left(\rho_j \cdot P_j\right)}{n} \cdot \frac{A_{i-1}}{\sum_{j=0}^{k-1} Y_j} + \frac{\sum_{j=1}^{k-2}\left(\rho_j \cdot P_j\right)}{n} \cdot \frac{Y_{i-1} - A_{i-1} - P_{i-1}}{\sum_{j=0}^{k-1}\left(Y_j - A_j - P_j\right)} + \frac{\rho_i \cdot P_i}{n} \cdot \frac{Y_{k-1} - A_{k-1} - P_{k-1}}{\sum_{j=0}^{k-1} Y_j}$ $+\frac{\sum_{j=1}^{k-2}\left(\rho_j \cdot P_j\right)}{n} \cdot \left(1 + \frac{Y_{k-1} - A_{k-1} - P_{k-1}}{\sum_{j=0}^{k-1} Y_j}\right) \cdot \frac{P_i}{\sum_{j=0}^{k-1}\left(Y_j - A_j - P_j\right)} + O(1/n)$
A_{k-1}'	$-\frac{\sum_{j=1}^{k-2}\left(\rho_j \cdot P_j\right)}{n} \cdot \frac{A_{k-1}}{\sum_{j=0}^{k-1} Y_j} + \frac{\sum_{j=1}^{k-2}\left(\rho_j \cdot P_j\right)}{n} \cdot \frac{A_{k-2}}{\sum_{j=0}^{k-1} Y_j} + \frac{\sum_{j=1}^{k-2}\left(\rho_j \cdot P_j\right)}{n} \cdot \frac{Y_{k-2} - A_{k-2} - P_{k-2}}{\sum_{j=0}^{k-1}\left(Y_j - A_j - P_j\right)}$ $+\frac{\sum_{j=1}^{k-2}\left(\rho_j \cdot P_j\right)}{n} \cdot \left(1 + \frac{Y_{k-1} - A_{k-1} - P_{k-1}}{\sum_{j=0}^{k-1} Y_j}\right) \cdot \frac{P_{k-1}}{\sum_{j=0}^{k-1}\left(Y_j - A_j - P_j\right)} + O(1/n)$
A_k'	$\frac{\sum_{j=1}^{k-2}\left(\rho_j \cdot P_j\right)}{n} \cdot \frac{A_{k-1}}{\sum_{j=0}^{k-1} Y_j} + \frac{\sum_{j=1}^{k-2}\left(\rho_j \cdot P_j\right)}{n} \cdot \frac{Y_{k-1} - A_{k-1} - P_{k-1}}{\sum_{j=0}^{k-1}\left(Y_j - A_j - P_j\right)} + \frac{\sum_{j=1}^{k-2}\left(\rho_j \cdot P_j\right)}{n} \cdot \left(1 + \frac{Y_{k-1} - A_{k-1} - P_{k-1}}{\sum_{j=0}^{k-1} Y_j}\right) \cdot \frac{P_k}{\sum_{j=0}^{k-1}\left(Y_j - A_j - P_j\right)} + O(1/n)$
P_i' for $1 \le i \le k-2$	$-\frac{\sum_{j=1}^{k-2}\left(\rho_j \cdot P_j\right)}{n} \cdot \frac{P_i}{\sum_{j=0}^{k-1} Y_j} + \frac{\sum_{j=1}^{k-2}\left(\rho_j \cdot P_j\right)}{n} \cdot \frac{P_{i-1}}{\sum_{j=0}^{k-1} Y_j} - \frac{\rho_i \cdot P_i}{n} - \frac{\sum_{j=1}^{k-2}\left(\rho_j \cdot P_j\right)}{n} \cdot \left(1 + \frac{Y_{k-1} - A_{k-1} - P_{k-1}}{\sum_{j=0}^{k-1} Y_j}\right) \cdot \frac{P_i}{\sum_{j=0}^{k-1}\left(Y_j - A_j - P_j\right)} + O(1/n)$
P_{k-1}'	$-\frac{\sum_{j=1}^{k-2}\left(\rho_j \cdot P_j\right)}{n} \cdot \frac{P_{k-1}}{\sum_{j=0}^{k-1} Y_j} + \frac{\sum_{j=1}^{k-2}\left(\rho_j \cdot P_j\right)}{n} \cdot \frac{P_{k-2}}{\sum_{j=0}^{k-1} Y_j} - \frac{\sum_{j=1}^{k-2}\left(\rho_j \cdot P_j\right)}{n} \cdot \left(1 + \frac{Y_{k-1} - A_{k-1} - P_{k-1}}{\sum_{j=0}^{k-1} Y_j}\right) \cdot \frac{P_{k-1}}{\sum_{j=0}^{k-1}\left(Y_j - A_j - P_j\right)} + O(1/n)$
P_k'	$\frac{\sum_{j=1}^{k-2}\left(\rho_j \cdot P_j\right)}{n} \cdot \frac{P_{k-1}}{\sum_{j=0}^{k-1} Y_j} - \frac{\sum_{j=1}^{k-2}\left(\rho_j \cdot P_j\right)}{n} \cdot \left(1 + \frac{Y_{k-1} - A_{k-1} - P_{k-1}}{\sum_{j=0}^{k-1} Y_j}\right) \cdot \frac{P_k}{\sum_{j=0}^{k-1}\left(Y_j - A_j - P_j\right)} + O(1/n)$

Legend	
	Contribution to expected change due to increase in degree of the priming vertex
	Contribution to expected change due to increase in degree of vertex v_t
	Contribution to expected change due to creation of augmentation edge by augmentation
	Contribution to expected change due to creation of additional augmentation edge if $v_t \in \mathcal{Y}_{k-1}(t) \setminus (\mathcal{A}_{k-1}(t) \cup \mathcal{P}_{k-1}(t))$
	Contribution to expected change due to destruction of priming edges

Event 3.2	
Y_0'	$-\left(\sum_{j=1}^{k-2}\left((1-\rho_j)\cdot\frac{P_j}{n}\right)\right)\cdot\frac{Y_0}{\sum_{j=0}^{k-2}Y_j}+O(1/n)$
Y_i' for $1\le i\le k-2$	$-(1-\rho_i)\cdot\frac{P_i}{n}+(1-\rho_{i-1})\cdot\frac{P_{i-1}}{n}-\left(\sum_{j=1}^{k-2}\left((1-\rho_j)\cdot\frac{P_j}{n}\right)\right)\cdot\frac{Y_i}{\sum_{j=0}^{k-2}Y_j}+\left(\sum_{j=1}^{k-2}\left((1-\rho_j)\cdot\frac{P_j}{n}\right)\right)\cdot\frac{Y_{i-1}}{\sum_{j=0}^{k-2}Y_j}+O(1/n)$
Y_{k-1}'	$(1-\rho_{k-2})\cdot\frac{P_{k-2}}{n}+\left(\sum_{j=1}^{k-2}\left((1-\rho_j)\cdot\frac{P_j}{n}\right)\right)\cdot\frac{Y_{k-2}}{\sum_{j=0}^{k-2}Y_j}+O(1/n)$
A_i' for $1\le i\le k-2$	$-\left(\sum_{j=1}^{k-2}\left((1-\rho_j)\cdot\frac{P_j}{n}\right)\right)\cdot\frac{A_i}{\sum_{j=0}^{k-2}Y_j}+\left(\sum_{j=1}^{k-2}\left((1-\rho_j)\cdot\frac{P_j}{n}\right)\right)\cdot\frac{A_{i-1}}{\sum_{j=0}^{k-2}Y_j}+O(1/n)$
A_{k-1}'	$\left(\sum_{j=1}^{k-2}\left((1-\rho_j)\cdot\frac{P_j}{n}\right)\right)\cdot\frac{A_{k-2}}{\sum_{j=0}^{k-2}Y_j}+O(1/n)$
A_k'	0
P_i' for $1\le i\le k-2$	$-(1-\rho_i)\cdot\frac{P_i}{n}+(1-\rho_{i-1})\cdot\frac{P_{i-1}}{n}-\left(\sum_{j=1}^{k-2}\left((1-\rho_j)\cdot\frac{P_j}{n}\right)\right)\cdot\frac{P_i}{\sum_{j=0}^{k-2}Y_j}+\left(\sum_{j=1}^{k-2}\left((1-\rho_j)\cdot\frac{P_j}{n}\right)\right)\cdot\frac{P_{i-1}}{\sum_{j=0}^{k-2}Y_j}+O(1/n)$
P_{k-1}'	$(1-\rho_{k-2})\cdot\frac{P_{k-2}}{n}+\left(\sum_{j=1}^{k-2}\left((1-\rho_j)\cdot\frac{P_j}{n}\right)\right)\cdot\frac{P_{k-2}}{\sum_{j=0}^{k-2}Y_j}+O(1/n)$
P_k'	0

Legend	
	Contribution to expected change due to increase in degree of vertex u_t Contribution to expected change due to increase in degree of vertex v_t

Event 4	
Y'_0	0
Y'_i for $1 \le i \le k-2$	0
Y'_{k-1}	$-2 \cdot \frac{Y_{k-1}-A_{k-1}-P_{k-1}}{n}$
A'_i for $1 \le i \le k-2$	$\frac{Y_{k-1}-A_{k-1}-P_{k-1}}{n} \cdot \frac{2}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)} \cdot P_i$
A'_{k-1}	$\frac{Y_{k-1}-A_{k-1}-P_{k-1}}{n} \cdot \frac{2}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)} \cdot P_{k-1}$
A'_k	$\frac{Y_{k-1}-A_{k-1}-P_{k-1}}{n} + \frac{Y_{k-1}-A_{k-1}-P_{k-1}}{n} \cdot \frac{2}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)} \cdot P_k$
P'_i for $1 \le i \le k-2$	$-\frac{Y_{k-1}-A_{k-1}-P_{k-1}}{n} \cdot \frac{2}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)} \cdot P_i$
P'_{k-1}	$-\frac{Y_{k-1}-A_{k-1}-P_{k-1}}{n} \cdot \frac{2}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)} \cdot P_{k-1}$
P'_k	$-\frac{Y_{k-1}-A_{k-1}-P_{k-1}}{n} \cdot \frac{2}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)} \cdot P_k$

Legend	
	Contribution to expected change due to creation of new degree-k augmentation edge Contribution to expected change due to destruction of priming edges

Event 5.1 and Event 5.2	
Y'_0	0
Y'_i for $1 \le i \le k-2$	0
Y'_{k-1}	$-\frac{A_{k-1}}{n} - \frac{A_{k-1}}{n}$
A'_i for $1 \le i \le k-2$	0
A'_{k-1}	$-\frac{A_{k-1}}{n} - \frac{A_{k-1}}{n} \cdot \sigma_A + O(1/n)$
A'_k	$\frac{A_{k-1}}{n} + \frac{A_{k-1}}{n} \cdot \sigma_A + O(1/n)$
P'_i for $1 \le i \le k-2$	0
P'_{k-1}	$-\frac{A_{k-1}}{n} \cdot (1-\sigma_A) + O(1/n)$
P'_k	$\frac{A_{k-1}}{n} \cdot (1-\sigma_A) + O(1/n)$

Legend	
	Contribution to expected change due to increase in degree of vertex u_t Contribution to expected change due to increase in degree of vertex v_t

Event 6.1	
Y_0'	$-\rho_{k-1}\cdot\frac{P_{k-1}}{n}\cdot\frac{Y_0}{\sum_{j=0}^{k-1}Y_j}$ $-\rho_{k-1}\cdot\frac{P_{k-1}}{n}\cdot\frac{Y_0-A_0-P_0}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)}+O(1/n)$
Y_i' for $1\le i\le k-2$	$-\rho_{k-1}\cdot\frac{P_{k-1}}{n}\cdot\frac{Y_i}{\sum_{j=0}^{k-1}Y_j}$ $-\rho_{k-1}\cdot\frac{P_{k-1}}{n}\cdot\frac{Y_i-A_i-P_i}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)}$ $+\rho_{k-1}\cdot\frac{P_{k-1}}{n}\cdot\frac{Y_{i-1}}{\sum_{j=0}^{k-1}Y_j}$ $+\rho_{k-1}\cdot\frac{P_{k-1}}{n}\cdot\frac{Y_{i-1}-A_{i-1}-P_{i-1}}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)}+O(1/n)$
Y_{k-1}'	$-\rho_{k-1}\cdot\frac{P_{k-1}}{n}\cdot\frac{Y_{k-1}}{\sum_{j=0}^{k-1}Y_j}$ $-\rho_{k-1}\cdot\frac{P_{k-1}}{n}\cdot\frac{Y_{k-1}-A_{k-1}-P_{k-1}}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)}$ $+\rho_{k-1}\cdot\frac{P_{k-1}}{n}\cdot\frac{Y_{k-2}}{\sum_{j=0}^{k-1}Y_j}$ $+\rho_{k-1}\cdot\frac{P_{k-1}}{n}\cdot\frac{Y_{k-2}-A_{k-2}-P_{k-2}}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)}+O(1/n)$
A_i' for $1\le i\le k-2$	$\rho_{k-1}\cdot\frac{P_{k-1}}{n}\cdot\frac{Y_{i-1}-A_{i-1}-P_{i-1}}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)}$ $-\rho_{k-1}\cdot\frac{P_{k-1}}{n}\cdot\frac{A_i}{\sum_{j=0}^{k-1}Y_j}+\rho_{k-1}\cdot\frac{P_{k-1}}{n}\cdot\frac{A_{i-1}}{\sum_{j=0}^{k-1}Y_j}$ $+\rho_{k-1}\cdot\frac{P_{k-1}}{n}\cdot\frac{Y_{k-1}-A_{k-1}-P_{k-1}}{\sum_{j=0}^{k-1}Y_j}\cdot\frac{P_i}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)}$ $+\rho_{k-1}\cdot\frac{P_{k-1}}{n}\cdot\frac{P_i}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)}+O(1/n)$
A_{k-1}'	$\rho_{k-1}\cdot\frac{P_{k-1}}{n}\cdot\frac{Y_{k-2}-A_{k-2}-P_{k-2}}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)}$ $-\rho_{k-1}\cdot\frac{P_{k-1}}{n}\cdot\frac{A_{k-1}}{\sum_{j=0}^{k-1}Y_j}+\rho_{k-1}\cdot\frac{P_{k-1}}{n}\cdot\frac{A_{k-2}}{\sum_{j=0}^{k-1}Y_j}$ $+\frac{\rho_{k-1}\cdot P_{k-1}}{n}\cdot\frac{Y_{k-1}-A_{k-1}-P_{k-1}}{\sum_{j=0}^{k-1}Y_j}$ $+\rho_{k-1}\cdot\frac{P_{k-1}}{n}\cdot\frac{Y_{k-1}-A_{k-1}-P_{k-1}}{\sum_{j=0}^{k-1}Y_j}\cdot\frac{P_{k-1}}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)}$ $+\rho_{k-1}\cdot\frac{P_{k-1}}{n}\cdot\frac{P_{k-1}}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)}+O(1/n)$
A_k'	$\rho_{k-1}\cdot\frac{P_{k-1}}{n}\cdot\frac{Y_{k-1}-A_{k-1}-P_{k-1}}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)}$ $+\rho_{k-1}\cdot\frac{P_{k-1}}{n}\cdot\frac{A_{k-1}}{\sum_{j=0}^{k-1}Y_j}$ $+\rho_{k-1}\cdot\frac{P_{k-1}}{n}\cdot\frac{Y_{k-1}-A_{k-1}-P_{k-1}}{\sum_{j=0}^{k-1}Y_j}\cdot\frac{P_k}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)}$ $+\rho_{k-1}\cdot\frac{P_{k-1}}{n}\cdot\frac{P_k}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)}+O(1/n)$
P_i' for $1\le i\le k-2$	$-\rho_{k-1}\cdot\frac{P_{k-1}}{n}\cdot\frac{P_i}{\sum_{j=0}^{k-1}Y_j}+\rho_{k-1}\cdot\frac{P_{k-1}}{n}\cdot\frac{P_{i-1}}{\sum_{j=0}^{k-1}Y_j}$ $-\rho_{k-1}\cdot\frac{P_{k-1}}{n}\cdot\frac{Y_{k-1}-A_{k-1}-P_{k-1}}{\sum_{j=0}^{k-1}Y_j}\cdot\frac{P_i}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)}$ $-\rho_{k-1}\cdot\frac{P_{k-1}}{n}\cdot\frac{P_i}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)}$ $+O(1/n)$
P_{k-1}'	$-\rho_{k-1}\cdot\frac{P_{k-1}}{n}$ $-\rho_{k-1}\cdot\frac{P_{k-1}}{n}\cdot\frac{P_{k-1}}{\sum_{j=0}^{k-1}Y_j}+\rho_{k-1}\cdot\frac{P_{k-1}}{n}\cdot\frac{P_{k-2}}{\sum_{j=0}^{k-1}Y_j}$ $-\rho_{k-1}\cdot\frac{P_{k-1}}{n}\cdot\frac{Y_{k-1}-A_{k-1}-P_{k-1}}{\sum_{j=0}^{k-1}Y_j}\cdot\frac{P_{k-1}}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)}$ $-\rho_{k-1}\cdot\frac{P_{k-1}}{n}\cdot\frac{P_{k-1}}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)}+O(1/n)$
P_k'	$\rho_{k-1}\cdot\frac{P_{k-1}}{n}\cdot\frac{P_{k-1}}{\sum_{j=0}^{k-1}Y_j}$ $-\rho_{k-1}\cdot\frac{P_{k-1}}{n}\cdot\frac{Y_{k-1}-A_{k-1}-P_{k-1}}{\sum_{j=0}^{k-1}Y_j}\cdot\frac{P_k}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)}$ $-\rho_{k-1}\cdot\frac{P_{k-1}}{n}\cdot\frac{P_k}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)}+O(1/n)$

Legend	
	Contribution to expected change due to increase in degree of the priming vertex
	Contribution to expected change due to increase in degree of vertex v_t
	Contribution to expected change due to creation of augmentation edge by augmentation
	Contribution to expected change due to creation of additional augmentation edge if $v_t\in\mathcal{Y}_{k-1}(t)\setminus(\mathcal{A}_{k-1}(t)\cup\mathcal{P}_{k-1}(t))$
	Contribution to expected change due to destruction of priming edges

Event 6.2.1 and Event 6.2.2	
Y'_0	0
Y'_i for $1 \le i \le k-2$	0
Y'_{k-1}	$-\frac{P_{k-1}}{n} \cdot (1-\rho_{k-1}) - \frac{P_{k-1}}{n} \cdot (1-\rho_{k-1})$
A'_i for $1 \le i \le k-2$	0
A'_{k-1}	$-\frac{P_{k-1}}{n} \cdot (1-\rho_{k-1}) \cdot \sigma_P + O(1/n)$
A'_k	$\frac{P_{k-1}}{n} \cdot (1-\rho_{k-1}) \cdot \sigma_P + O(1/n)$
P'_i for $1 \le i \le k-2$	0
P'_{k-1}	$-\frac{P_{k-1}}{n} \cdot (1-\rho_{k-1}) - \frac{P_{k-1}}{n} \cdot (1-\rho_{k-1}) \cdot (1-\sigma_P) + O(1/n)$
P'_k	$\frac{P_{k-1}}{n} \cdot (1-\rho_{k-1}) + \frac{P_{k-1}}{n} \cdot (1-\rho_{k-1}) \cdot (1-\sigma_P) + O(1/n)$
Legend	Contribution to expected change due to increase in degree of vertex u_t Contribution to expected change due to increase in degree of vertex v_t

Event 7	
Y'_0	0
Y'_i for $1 \le i \le k-2$	0
Y'_{k-1}	0
A'_i for $1 \le i \le k-2$	0
A'_{k-1}	0
A'_k	$-\frac{2 \cdot A_k}{n}$
P'_i for $1 \le i \le k-2$	0
P'_{k-1}	0
P'_k	$\frac{2 \cdot A_k}{n}$
Legend	Contribution to expected change due to priming of degree-k augmentation edge

Event 8.1	
Y'_0	0
Y'_i for $1 \le i \le k-2$	0
Y'_{k-1}	0
A'_i for $1 \le i \le k-2$	0
A'_{k-1}	0
A'_k	0
P'_i for $1 \le i \le k-2$	0
P'_{k-1}	0
P'_k	0

Event 8.2	
Y_0'	$-\frac{P_k}{n}\cdot\frac{Y_0-A_0-P_0}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)}-\frac{P_k}{n}\cdot\frac{Y_0-A_0-P_0}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)}+O(1/n)$
Y_i' for $1\le i\le k-2$	$-\frac{P_k}{n}\cdot\frac{Y_i-A_i-P_i}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)}-\frac{P_k}{n}\cdot\frac{Y_i-A_i-P_i}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)}+\frac{P_k}{n}\cdot\frac{Y_{i-1}-A_{i-1}-P_{i-1}}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)}+\frac{P_k}{n}\cdot\frac{Y_{i-1}-A_{i-1}-P_{i-1}}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)}+O(1/n)$
Y_{k-1}'	$-\frac{P_k}{n}\cdot\frac{Y_{k-1}-A_{k-1}-P_{k-1}}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)}-\frac{P_k}{n}\cdot\frac{Y_{k-1}-A_{k-1}-P_{k-1}}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)}+\frac{P_k}{n}\cdot\frac{Y_{k-2}-A_{k-2}-P_{k-2}}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)}+\frac{P_k}{n}\cdot\frac{Y_{k-2}-A_{k-2}-P_{k-2}}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)}+O(1/n)$
A_i' for $1\le i\le k-2$	$\frac{P_k}{n}\cdot 2\cdot\frac{Y_{i-1}-A_{i-1}-P_{i-1}}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)}+\frac{P_k}{n}\cdot 2\cdot\frac{P_i}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)}+O(1/n)$
A_{k-1}'	$\frac{P_k}{n}\cdot 2\cdot\frac{Y_{k-2}-A_{k-2}-P_{k-2}}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)}+\frac{P_k}{n}\cdot 2\cdot\frac{P_{k-1}}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)}+O(1/n)$
A_k'	$\frac{P_k}{n}\cdot 2\cdot\frac{Y_{k-1}-A_{k-1}-P_{k-1}}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)}+\frac{P_k}{n}\cdot 2\cdot\frac{P_k}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)}+O(1/n)$
P_i' for $1\le i\le k-2$	$-\frac{P_k}{n}\cdot 2\cdot\frac{P_i}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)}+O(1/n)$
P_{k-1}'	$-\frac{P_k}{n}\cdot 2\cdot\frac{P_{k-1}}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)}+O(1/n)$
P_k'	$-\frac{P_k}{n}-\frac{P_k}{n}\cdot 2\cdot\frac{P_k}{\sum_{j=0}^{k-1}\left(Y_j-A_j-P_j\right)}+O(1/n)$

Legend	
	Contribution to expected change due to increase in degree of the priming vertex Contribution to expected change due to increase in degree of vertex v_t Contribution to expected change due to creation of augmentation edge by augmentation Contribution to expected change due to destruction of priming edges

Event 9	
Y'_0	0
Y'_i for $1 \le i \le k-2$	0
Y'_{k-1}	0
A'_i for $1 \le i \le k-2$	$-\frac{A_i}{n}$
A'_{k-1}	$-\frac{A_{k-1}}{n}$
A'_k	0
P'_i for $1 \le i \le k-2$	$\frac{A_i}{n}$
P'_{k-1}	$\frac{A_{k-1}}{n}$
P'_k	0
Legend	Contribution to expected change due to priming of degree-i augmentation edge for $1 \le i < k$

Event 10	
Y'_0	0
Y'_i for $1 \le i \le k-2$	0
Y'_{k-1}	0
A'_i for $1 \le i \le k-2$	0
A'_{k-1}	0
A'_k	0
P'_i for $1 \le i \le k-2$	0
P'_{k-1}	0
P'_k	0

Event 11	
Y'_0	0
Y'_i for $1 \le i \le k-2$	0
Y'_{k-1}	0
A'_i for $1 \le i \le k-2$	0
A'_{k-1}	0
A'_k	0
P'_i for $1 \le i \le k-2$	0
P'_{k-1}	0
P'_k	0

Demotion event	
Y'_0	0
Y'_i for $1 \leq i \leq k-2$	0
Y'_{k-1}	0
A'_i for $1 \leq i \leq k-2$	$-\phi \cdot \frac{A_i}{\sum_{j=0}^{k-1}\left(Y_j - A_j - P_j\right)} + O(1/n)$
A'_{k-1}	$-\phi \cdot \frac{A_{k-1}}{\sum_{j=0}^{k-1}\left(Y_j - A_j - P_j\right)} + O(1/n)$
A'_k	0
P'_i for $1 \leq i \leq k-2$	$-\phi \cdot \frac{P_i}{\sum_{j=0}^{k-1}\left(Y_j - A_j - P_j\right)} + O(1/n)$
P'_{k-1}	$-\phi \cdot \frac{P_{k-1}}{\sum_{j=0}^{k-1}\left(Y_j - A_j - P_j\right)} + O(1/n)$
P'_k	0
Legend	Contribution to expected change due to demotion of augmentation edges

Appendix C

Differential equations for the k-factor algorithm

We use y_i, a_i, and p_i to denote the functions $y_i(s)$, $a_i(s)$, and $p_i(s)$ respectively.

$$
\begin{aligned}
y_0' = {} & y_0 + p_0 \\
& - \frac{\left(\sum_{j=0}^{k-2} (y_j - \rho_j \cdot p_j)\right) \cdot y_0}{\sum_{j=0}^{k-2} y_j} \\
& - \frac{\left(\sum_{j=1}^{k-1} (\rho_j \cdot p_j)\right) \cdot y_0}{\sum_{j=0}^{k-1} y_j} \\
& - \frac{\left(\left(\sum_{j=1}^{k-1} (\rho_j \cdot p_j)\right) + 2p_k\right) \cdot (y_0 - a_0 - p_0)}{\sum_{j=0}^{k-1} (y_j - a_j - p_j)}
\end{aligned}
$$

For all $1 \leq i \leq k-2$,

$$\begin{aligned}
y_i' = &- y_i + \rho_i \cdot p_i + y_{i-1} - \rho_{i-1} \cdot p_{i-1} \\
&+ \frac{\left(\sum_{j=1}^{k-1}(\rho_j \cdot p_j)\right) \cdot (-y_i + y_{i-1})}{\sum_{j=0}^{k-1} y_j} \\
&+ \frac{\left(\sum_{j=0}^{k-2}(y_j - \rho_j \cdot p_j)\right) \cdot (-y_i + y_{i-1})}{\sum_{j=0}^{k-2} y_j} \\
&+ \frac{\left(\left(\sum_{j=1}^{k-1}(\rho_j \cdot p_j)\right) + 2p_k\right) \cdot (-y_i + a_i + p_i + y_{i-1} - a_{i-1} - p_{i-1})}{\sum_{j=0}^{k-1}(y_j - a_j - p_j)}
\end{aligned}$$

$$\begin{aligned}
y_{k-1}' = &\ y_{k-2} - \rho_{k-2} \cdot p_{k-2} - 2y_{k-1} + 2\rho_{k-1} \cdot p_{k-1} \\
&+ \frac{\left(\sum_{j=1}^{k-1}(\rho_j \cdot p_j)\right) \cdot (-y_{k-1} + y_{k-2})}{\sum_{j=0}^{k-1} y_j} \\
&+ \frac{\left(\sum_{j=0}^{k-2}(y_j - \rho_j \cdot p_j)\right) \cdot y_{k-2}}{\sum_{j=0}^{k-2} y_j} \\
&+ \frac{\left(\left(\sum_{j=1}^{k-1}(\rho_j \cdot p_j)\right) + 2p_k\right) \cdot (-y_{k-1} + a_{k-1} + p_{k-1} + y_{k-2} - a_{k-2} - p_{k-2})}{\sum_{j=0}^{k-1}(y_j - a_j - p_j)}
\end{aligned}$$

For all $1 \leq i \leq k-2$,

$$
\begin{aligned}
a_i' = & -2a_i + a_{i-1} \\
& + \frac{\left(\sum_{j=0}^{k-2}(y_j - \rho_j \cdot p_j)\right) \cdot (-a_i + a_{i-1})}{\sum_{j=0}^{k-2} y_j} \\
& + \frac{\left(\sum_{j=1}^{k-1}(\rho_j \cdot p_j)\right) \cdot (-a_i + a_{i-1}) + \rho_i \cdot p_i \cdot (y_{k-1} - a_{k-1} - p_{k-1})}{\sum_{j=0}^{k-1} y_j} \\
& + \frac{\left(\sum_{j=1}^{k-1}(\rho_j \cdot p_j)\right) \cdot (y_{i-1} - a_{i-1} - p_{i-1} + p_i) - (y_{k-1} - a_{k-1} - p_{k-1}) \cdot 2p_i + (y_{i-1} - a_{i-1} - p_{i-1} + p_i) \cdot 2p_k - \phi \cdot a_i}{\sum_{j=0}^{k-1} (y_j - a_j - p_j)} \\
& + \frac{\left(\sum_{j=1}^{k-1} (\rho_j \cdot p_j)\right) \cdot (y_{k-1} - a_{k-1} - p_{k-1}) \cdot p_i}{\left(\sum_{j=0}^{k-1} y_j\right) \cdot \left(\sum_{j=0}^{k-1} (y_j - a_j - p_j)\right)}
\end{aligned}
$$

$$
\begin{aligned}
a'_{k-1} = {} & a_{k-2} - 2a_{k-1} - a_{k-1} \cdot \sigma_A - p_{k-1} \cdot (1 - \rho_{k-1}) \cdot \sigma_P \\
& + \frac{\left(\sum_{j=0}^{k-2} (y_j - \rho_j \cdot p_j)\right) \cdot a_{k-2}}{\sum_{j=0}^{k-2} y_j} \\
& + \frac{\left(\sum_{j=1}^{k-1} (\rho_j \cdot p_j)\right) \cdot (-a_{k-1} + a_{k-2}) + \rho_{k-1} \cdot p_{k-1} \cdot (y_{k-1} - a_{k-1} - p_{k-1})}{\sum_{j=0}^{k-1} y_j} \\
& + \frac{\left(\sum_{j=1}^{k-1} (\rho_j \cdot p_j)\right) \cdot (y_{k-2} - a_{k-2} - p_{k-2} + p_{k-1}) + (y_{k-1} - a_{k-1} - p_{k-1}) \cdot 2p_{k-1}}{\sum_{j=0}^{k-1} (y_j - a_j - p_j)} \\
& + \frac{(y_{k-2} - a_{k-2} - p_{k-2} + p_{k-1}) \cdot 2p_k - \phi \cdot a_{k-1}}{\sum_{j=0}^{k-1} (y_j - a_j - p_j)} \\
& + \frac{\left(\sum_{j=1}^{k-1} (\rho_j \cdot p_j)\right) \cdot (y_{k-1} - a_{k-1} - p_{k-1}) \cdot p_{k-1}}{\left(\sum_{j=0}^{k-1} y_j\right) \cdot \left(\sum_{j=0}^{k-1} (y_j - a_j - p_j)\right)} \\
a'_k = {} & y_{k-1} - p_{k-1} + a_{k-1} \cdot \sigma_A - 2a_k + p_{k-1} \cdot (1 - \rho_{k-1}) \cdot \sigma_P \\
& + \frac{\left(\sum_{j=1}^{k-1} (\rho_j \cdot p_j)\right) \cdot a_{k-1}}{\sum_{j=0}^{k-1} y_j} \\
& + \frac{\left(\sum_{j=1}^{k-1} (\rho_j \cdot p_j)\right) \cdot (y_{k-1} - a_{k-1} - p_{k-1} + p_k) + (y_{k-1} - a_{k-1} - p_{k-1}) \cdot 4p_k + 2p_k^2}{\sum_{j=0}^{k-1} (y_j - a_j - p_j)} \\
& + \frac{\left(\sum_{j=1}^{k-1} (\rho_j \cdot p_j)\right) \cdot (y_{k-1} - a_{k-1} - p_{k-1}) \cdot p_k}{\left(\sum_{j=0}^{k-1} y_j\right) \cdot \left(\sum_{j=0}^{k-1} (y_j - a_j - p_j)\right)}
\end{aligned}
$$

For all $1 \leq i \leq k-2$,

$$
\begin{aligned}
p_i' = & -(1-\rho_i)\cdot p_i + (1-\rho_{i-1})\cdot p_{i-1} - \rho_i \cdot p_i + a_i \\
& + \frac{\left(\sum_{j=0}^{k-2}(y_j - \rho_j \cdot p_j)\right)\cdot(-p_i + p_{i-1})}{\sum_{j=0}^{k-2} y_j} \\
& + \frac{\left(\sum_{j=1}^{k-1}(\rho_j \cdot p_j)\right)\cdot(-p_i + p_{i-1})}{\sum_{j=0}^{k-1} y_j} \\
& + \frac{-\left(\sum_{j=1}^{k-1}(\rho_j \cdot p_j)\right)\cdot p_i - (y_{k-1} - a_{k-1} - p_{k-1} + p_k)\cdot 2p_i - \phi \cdot p_i}{\sum_{j=0}^{k-1}(y_j - a_j - p_j)} \\
& + \frac{-\left(\sum_{j=1}^{k-1}(\rho_j \cdot p_j)\right)\cdot(y_{k-1} - a_{k-1} - p_{k-1})\cdot p_i}{\left(\sum_{j=0}^{k-1} y_j\right)\cdot\left(\sum_{j=0}^{k-1}(y_j - a_j - p_j)\right)} \\
p_{k-1}' = & (1-\rho_{k-2})\cdot p_{k-2} - \rho_{k-1}\cdot p_{k-1} - p_{k-1}\cdot(1-\rho_{k-1}) - a_{k-1}\cdot(1-\sigma_A) - p_{k-1}\cdot(1-\rho_{k-1})\cdot(1-\sigma_P) + a_{k-1} \\
& + \frac{\left(\sum_{j=0}^{k-2}(y_j - \rho_j \cdot p_j)\right)\cdot p_{k-2}}{\sum_{j=0}^{k-2} y_j} \\
& + \frac{\left(\sum_{j=1}^{k-1}(\rho_j \cdot p_j)\right)\cdot(-p_{k-1} + p_{k-2})}{\sum_{j=0}^{k-1} y_j} \\
& + \frac{-\left(\sum_{j=1}^{k-1}(\rho_j \cdot p_j)\right)\cdot p_{k-1} - (y_{k-1} - a_{k-1} - p_{k-1} + p_k)\cdot 2p_{k-1} - \phi \cdot p_{k-1}}{\sum_{j=0}^{k-1}(y_j - a_j - p_j)} \\
& + \frac{-\left(\sum_{j=1}^{k-1}(\rho_j \cdot p_j)\right)\cdot(y_{k-1} - a_{k-1} - p_{k-1})\cdot p_{k-1}}{\left(\sum_{j=0}^{k-1} y_j\right)\cdot\left(\sum_{j=0}^{k-1}(y_j - a_j - p_j)\right)}
\end{aligned}
$$

$$
\begin{aligned}
p'_k = {} & a_{k-1} \cdot (1 - \sigma_A) + p_{k-1} \cdot (1 - \rho_{k-1}) \cdot (2 - \sigma_P) + 2a_k - p_k \\
& + \frac{\left(\sum_{j=1}^{k-1} (\rho_j \cdot p_j)\right) \cdot p_{k-1}}{\sum_{j=0}^{k-1} y_j} \\
& + \frac{\left(\sum_{j=1}^{k-1} (\rho_j \cdot p_j)\right) \cdot p_k - (y_{k-1} - a_{k-1} - p_{k-1} + p_k) \cdot 2p_k}{\sum_{j=0}^{k-1} (y_j - a_j - p_j)} \\
& + \frac{-\left(\sum_{j=1}^{k-1} (\rho_j \cdot p_j)\right) \cdot (y_{k-1} - a_{k-1} - p_{k-1}) \cdot p_k}{\left(\sum_{j=0}^{k-1} y_j\right) \cdot \left(\sum_{j=0}^{k-1} (y_j - a_j - p_j)\right)}
\end{aligned}
$$

Appendix D

Maple code

```
printlevel := 0;
with(ArrayTools);
with(ListTools);
_EnvTry = 'hard';

k := 5;
varsy_1based := [y || (0 .. k)(t)];
varsa_1based := [a || (0 .. k)(t)];
varsp_1based := [p || (0 .. k)(t)];
varsy := Array(0 .. k, i -> varsy_1based[i + 1]);
varsa := Array(0 .. k, i -> varsa_1based[i + 1]);
varsp := Array(0 .. k, i -> varsp_1based[i + 1]);

rhsy := Array(0 .. k, i -> 0);
rhsa := Array(0 .. k, i -> 0);
rhsp := Array(0 .. k, i -> 0);

for i from 0 to k do
        PAD || i(t) := 0;
end do;
PAD := Array(0 .. k, i -> PAD || i(t));
SigmaA := t -> 0;
SigmaP := t -> 0;

demotion_factor := 0.55;

(* EVENT 1 *)
rhsy[0] += -varsy[0] + varsa[0] + varsp[0];
rhsy[0] += -add(varsy[j] - varsa[j] - varsp[j], j = 0 .. k - 2)*varsy[0]/add
    (varsy[j], j = 0 .. k - 2);
for i to k - 2 do
    rhsy[i] += -varsy[i] + varsa[i] + varsp[i];
    rhsy[i] += -add(varsy[j] - varsa[j] - varsp[j], j = 0 .. k - 2)*varsy[i
    ]/add(varsy[j], j = 0 .. k - 2);
    rhsy[i] += varsy[i - 1] - varsa[i - 1] - varsp[i - 1];
```

```
34      rhsy[i] += add(varsy[j] - varsa[j] - varsp[j], j = 0 .. k - 2)*varsy[i -
        1]/add(varsy[j], j = 0 .. k - 2);
35  end do;
36  rhsy[k - 1] += varsy[k - 2] - varsa[k - 2] - varsp[k - 2];
37  rhsy[k - 1] += add(varsy[j] - varsa[j] - varsp[j], j = 0 .. k - 2)*varsy[k -
        2]/add(varsy[j], j = 0 .. k - 2);
38  for i to k - 2 do
39      rhsa[i] += -add(varsy[j] - varsa[j] - varsp[j], j = 0 .. k - 2)*varsa[i
        ]/add(varsy[j], j = 0 .. k - 2);
40      rhsa[i] += add(varsy[j] - varsa[j] - varsp[j], j = 0 .. k - 2)*varsa[i -
        1]/add(varsy[j], j = 0 .. k - 2);
41  end do;
42  rhsa[k - 1] += add(varsy[j] - varsa[j] - varsp[j], j = 0 .. k - 2)*varsa[k -
        2]/add(varsy[j], j = 0 .. k - 2);
43  rhsa[k] += 0;
44  for i to k - 2 do
45      rhsp[i] += -add(varsy[j] - varsa[j] - varsp[j], j = 0 .. k - 2)*varsp[i
        ]/add(varsy[j], j = 0 .. k - 2);
46      rhsp[i] += add(varsy[j] - varsa[j] - varsp[j], j = 0 .. k - 2)*varsp[i -
        1]/add(varsy[j], j = 0 .. k - 2);
47  end do;
48  rhsp[k - 1] += add(varsy[j] - varsa[j] - varsp[j], j = 0 .. k - 2)*varsp[k -
        2]/add(varsy[j], j = 0 .. k - 2);
49  rhsp[k] += 0;
50  print("Event 1 processed");
51
52  (* EVENT 2 *)
53  rhsy[0] += -varsa[0] - add(varsa[j], j = 0 .. k - 2)*varsy[0]/add(varsy[j],
        j = 0 .. k - 2);
54  for i to k - 2 do
55      rhsy[i] += -varsa[i] - add(varsa[j], j = 0 .. k - 2)*varsy[i]/add(varsy[
        j], j = 0 .. k - 2);
56      rhsy[i] += varsa[i - 1] + add(varsa[j], j = 0 .. k - 2)*varsy[i - 1]/add
        (varsy[j], j = 0 .. k - 2);
57  end do;
58  rhsy[k - 1] += varsa[k - 2] + add(varsa[j], j = 0 .. k - 2)*varsy[k - 2]/add
        (varsy[j], j = 0 .. k - 2);
59  for i to k - 2 do
60      rhsa[i] += -varsa[i] - add(varsa[j], j = 0 .. k - 2)*varsa[i]/add(varsy[
        j], j = 0 .. k - 2);
61      rhsa[i] += varsa[i - 1] + add(varsa[j], j = 0 .. k - 2)*varsa[i - 1]/add
        (varsy[j], j = 0 .. k - 2);
62  end do;
63  rhsa[k - 1] += varsa[k - 2] + add(varsa[j], j = 0 .. k - 2)*varsa[k - 2]/add
        (varsy[j], j = 0 .. k - 2);
64  for i to k - 2 do
65      rhsp[i] += -add(varsa[j], j = 0 .. k - 2)*varsp[i]/add(varsy[j], j = 0
        .. k - 2);
66      rhsp[i] += add(varsa[j], j = 0 .. k - 2)*varsp[i - 1]/add(varsy[j], j =
        0 .. k - 2);
67  end do;
```

```
68 rhsp[k - 1] += add(varsa[j], j = 0 .. k - 2)*varsp[k - 2]/add(varsy[j], j =
       0 .. k - 2);
69 print("Event 2 processed");
70
71 (* EVENT 3.1 *)
72 rhsy[0] += -add(PAD[j]*varsp[j], j = 1 .. k - 2)*varsy[0]/add(varsy[j], j =
       0 .. k - 1);
73 rhsy[0] += -add(PAD[j]*varsp[j], j = 1 .. k - 2)*(varsy[0] - varsa[0] -
       varsp[0])/add(varsy[j] - varsa[j] - varsp[j], j = 0 .. k - 1);
74 for i to k - 1 do
75     rhsy[i] += -add(PAD[j]*varsp[j], j = 1 .. k - 2)*varsy[i]/add(varsy[j],
       j = 0 .. k - 1);
76     rhsy[i] += add(PAD[j]*varsp[j], j = 1 .. k - 2)*varsy[i - 1]/add(varsy[j
       ], j = 0 .. k - 1);
77     rhsy[i] += -add(PAD[j]*varsp[j], j = 1 .. k - 2)*(varsy[i] - varsa[i] -
       varsp[i])/add(varsy[j] - varsa[j] - varsp[j], j = 0 .. k - 1);
78     rhsy[i] += add(PAD[j]*varsp[j], j = 1 .. k - 2)*(varsy[i - 1] - varsa[i
       - 1] - varsp[i - 1])/add(varsy[j] - varsa[j] - varsp[j], j = 0 .. k - 1);
79 end do;
80 rhsy[k] += add(PAD[j]*varsp[j], j = 0 .. k - 2)*varsy[k - 1]/add(varsy[j], j
       = 0 .. k - 1);
81 rhsy[k] += add(PAD[j]*varsp[j], j = 0 .. k - 2)*(varsy[k - 1] - varsa[k - 1]
       - varsp[k - 1])/add(varsy[j] - varsa[j] - varsp[j], j = 0 .. k - 1);
82 for i to k - 2 do
83     rhsa[i] += -add(PAD[j]*varsp[j], j = 1 .. k - 2)*varsa[i]/add(varsy[j],
       j = 0 .. k - 1);
84     rhsa[i] += add(PAD[j]*varsp[j], j = 1 .. k - 2)*varsa[i - 1]/add(varsy[j
       ], j = 0 .. k - 1);
85     rhsa[i] += add(PAD[j]*varsp[j], j = 1 .. k - 2)*(varsy[i - 1] - varsa[i
       - 1] - varsp[i - 1])/add(varsy[j] - varsa[j] - varsp[j], j = 0 .. k - 1);
86     rhsa[i] += PAD[i]*varsp[i]*(varsy[k - 1] - varsa[k - 1] - varsp[k - 1])/
       add(varsy[j], j = 0 .. k - 1);
87     rhsa[i] += add(PAD[j]*varsp[j], j = 1 .. k - 2)*(1 + (varsy[k - 1] -
       varsa[k - 1] - varsp[k - 1])/add(varsy[j], j = 0 .. k - 1))*varsp[i]/add(
       varsy[j] - varsa[j] - varsp[j], j = 0 .. k - 1);
88 end do;
89 rhsa[k - 1] += -add(PAD[j]*varsp[j], j = 1 .. k - 2)*varsa[k - 1]/add(varsy[
       j], j = 0 .. k - 1);
90 rhsa[k - 1] += add(PAD[j]*varsp[j], j = 1 .. k - 2)*varsa[k - 2]/add(varsy[j
       ], j = 0 .. k - 1);
91 rhsa[k - 1] += add(PAD[j]*varsp[j], j = 1 .. k - 2)*(varsy[k - 2] - varsa[k
       - 2] - varsp[k - 2])/add(varsy[j] - varsa[j] - varsp[j], j = 0 .. k - 1);
92 rhsa[k - 1] += add(PAD[j]*varsp[j], j = 1 .. k - 2)*(1 + (varsy[k - 1] -
       varsa[k - 1] - varsp[k - 1])/add(varsy[j], j = 0 .. k - 1))*varsp[k - 1]/
       add(varsy[j] - varsa[j] - varsp[j], j = 0 .. k - 1);
93 rhsa[k] += add(PAD[j]*varsp[j], j = 1 .. k - 2)*varsa[k - 1]/add(varsy[j], j
       = 0 .. k - 1);
94 rhsa[k] += add(PAD[j]*varsp[j], j = 1 .. k - 2)*(varsy[k - 1] - varsa[k - 1]
       - varsp[k - 1])/add(varsy[j] - varsa[j] - varsp[j], j = 0 .. k - 1);
95 rhsa[k] += add(PAD[j]*varsp[j], j = 1 .. k - 2)*(1 + (varsy[k - 1] - varsa[k
       - 1] - varsp[k - 1])/add(varsy[j], j = 0 .. k - 1))*varsp[k]/add(varsy[j
       ] - varsa[j] - varsp[j], j = 0 .. k - 1);
```

```
96  for i to k - 1 do
97      rhsp[i] += -add(PAD[j]*varsp[j], j = 1 .. k - 2)*varsp[i]/add(varsy[j],
        j = 0 .. k - 1);
98      rhsp[i] += add(PAD[j]*varsp[j], j = 1 .. k - 2)*varsp[i - 1]/add(varsy[j
        ], j = 0 .. k - 1);
99      rhsp[i] += -PAD[i]*varsp[i];
100     rhsp[i] += -add(PAD[j]*varsp[j], j = 1 .. k - 2)*(1 + (varsy[k - 1] -
        varsa[k - 1] - varsp[k - 1])/add(varsy[j], j = 0 .. k - 1))*varsp[i]/add(
        varsy[j] - varsa[j] - varsp[j], j = 0 .. k - 1);
101 end do;
102 rhsp[k - 1] += -add(PAD[j]*varsp[j], j = 1 .. k - 2)*varsp[k - 1]/add(varsy[
        j], j = 0 .. k - 1);
103 rhsp[k - 1] += add(PAD[j]*varsp[j], j = 1 .. k - 2)*varsp[k - 2]/add(varsy[j
        ], j = 0 .. k - 1);
104 rhsp[k - 1] += -add(PAD[j]*varsp[j], j = 1 .. k - 2)*(1 + (varsy[k - 1] -
        varsa[k - 1] - varsp[k - 1])/add(varsy[j], j = 0 .. k - 1))*varsp[k - 1]/
        add(varsy[j] - varsa[j] - varsp[j], j = 0 .. k - 1);
105 rhsp[k] += add(PAD[j]*varsp[j], j = 1 .. k - 2)*varsp[k - 1]/add(varsy[j], j
     = 0 .. k - 1);
106 rhsp[k] += -add(PAD[j]*varsp[j], j = 1 .. k - 2)*(1 + (varsy[k - 1] - varsa[
        k - 1] - varsp[k - 1])/add(varsy[j], j = 0 .. k - 1))*varsp[k]/add(varsy[
        j] - varsa[j] - varsp[j], j = 0 .. k - 1);
107 print("Event 3.1 processed");
108
109 (* EVENT 3.2 *)
110 rhsy[0] += -add((1 - PAD[j])*varsp[j], j = 1 .. k - 2)*varsy[0]/add(varsy[j
        ], j = 0 .. k - 2);
111 for i to k - 2 do
112     rhsy[i] += -(1 - PAD[i])*varsp[i];
113     rhsy[i] += (1 - PAD[i - 1])*varsp[i - 1];
114     rhsy[i] += -add((1 - PAD[j])*varsp[j], j = 1 .. k - 2)*varsy[i]/add(
        varsy[j], j = 0 .. k - 2);
115     rhsy[i] += add((1 - PAD[j])*varsp[j], j = 1 .. k - 2)*varsy[i - 1]/add(
        varsy[j], j = 0 .. k - 2);
116 end do;
117 rhsy[k - 1] += (1 - PAD[k - 2])*varsp[k - 2];
118 rhsy[k - 1] += add((1 - PAD[j])*varsp[j], j = 1 .. k - 2)*varsy[k - 2]/add(
        varsy[j], j = 0 .. k - 2);
119 rhsy[k] += 0;
120 for i to k - 2 do
121     rhsa[i] += -add((1 - PAD[j])*varsp[j], j = 1 .. k - 2)*varsa[i]/add(
        varsy[j], j = 0 .. k - 2);
122     rhsa[i] += add((1 - PAD[j])*varsp[j], j = 1 .. k - 2)*varsa[i - 1]/add(
        varsy[j], j = 0 .. k - 2);
123 end do;
124 rhsa[k - 1] += add((1 - PAD[j])*varsp[j], j = 1 .. k - 2)*varsa[k - 2]/add(
        varsy[j], j = 0 .. k - 2);
125 rhsa[k] += 0;
126 for i to k - 2 do
127     rhsp[i] += -(1 - PAD[i])*varsp[i];
128     rhsp[i] += (1 - PAD[i - 1])*varsp[i - 1];
```

```
129     rhsp[i] += -add((1 - PAD[j])*varsp[j], j = 1 .. k - 2)*varsp[i]/add(
        varsy[j], j = 0 .. k - 2);
130     rhsp[i] += add((1 - PAD[j])*varsp[j], j = 1 .. k - 2)*varsp[i - 1]/add(
        varsy[j], j = 0 .. k - 2);
131 end do;
132 rhsp[k - 1] += (1 - PAD[k - 2])*varsp[k - 2];
133 rhsp[k - 1] += add((1 - PAD[j])*varsp[j], j = 1 .. k - 2)*varsp[k - 2]/add(
        varsy[j], j = 0 .. k - 2);
134 rhsp[k] += 0;
135 print("Event 3.2 processed");
136
137 (* EVENT 4 *)
138 rhsy[k - 1] += -2*(varsy[k - 1] - varsa[k - 1] - varsp[k - 1]);
139 rhsy[k] += 2*(varsy[k - 1] - varsa[k - 1] - varsp[k - 1]);
140 for i to k do
141     rhsa[i] += (varsy[k - 1] - varsa[k - 1] - varsp[k - 1])*2*varsp[i]/add(
        varsy[j] - varsa[j] - varsp[j], j = 0 .. k - 1);
142     rhsp[i] += -(varsy[k - 1] - varsa[k - 1] - varsp[k - 1])*2*varsp[i]/add(
        varsy[j] - varsa[j] - varsp[j], j = 0 .. k - 1);
143 end do;
144 rhsa[k] += varsy[k - 1] - varsa[k - 1] - varsp[k - 1];
145 print("Event 4 processed");
146
147 (* EVENTS 5.1 and 5.2 *)
148 rhsy[k - 1] += -2*varsa[k - 1];
149 rhsy[k] += 2*varsa[k - 1];
150 rhsa[k - 1] += -varsa[k - 1] - varsa[k - 1]*SigmaA(t);
151 rhsa[k] += varsa[k - 1] + varsa[k - 1]*SigmaA(t);
152 rhsp[k - 1] += -varsa[k - 1]*(1 - SigmaA(t));
153 rhsp[k] += varsa[k - 1]*(1 - SigmaA(t));
154 print("Events 5.1 and 5.2 processed");
155
156 (* EVENT 6.1 *)
157 rhsy[0] += -PAD[k - 1]*varsp[k - 1]*varsy[0]/add(varsy[j], j = 0 .. k - 1);
158 rhsy[0] += -PAD[k - 1]*varsp[k - 1]*(varsy[0] - varsa[0] - varsp[0])/add(
        varsy[j] - varsa[j] - varsp[j], j = 0 .. k - 1);
159 for i to k - 1 do
160     rhsy[i] += -PAD[k - 1]*varsp[k - 1]*varsy[i]/add(varsy[j], j = 0 .. k -
        1);
161     rhsy[i] += PAD[k - 1]*varsp[k - 1]*varsy[i - 1]/add(varsy[j], j = 0 .. k
        - 1);
162     rhsy[i] += -PAD[k - 1]*varsp[k - 1]*(varsy[i] - varsa[i] - varsp[i])/add
        (varsy[j] - varsa[j] - varsp[j], j = 0 .. k - 1);
163     rhsy[i] += PAD[k - 1]*varsp[k - 1]*(varsy[i - 1] - varsa[i - 1] - varsp[
        i - 1])/add(varsy[j] - varsa[j] - varsp[j], j = 0 .. k - 1);
164 end do;
165 rhsy[k] += varsp[k - 1]*PAD[k - 1]*(varsy[k - 1] - varsa[k - 1] - varsp[k -
        1])/add(varsy[j] - varsa[j] - varsp[j], j = 0 .. k - 1);
166 rhsy[k] += varsp[k - 1]*PAD[k - 1]*varsy[k - 1]/add(varsy[j], j = 0 .. k -
        1);
167 for i to k - 1 do
```

```
168      rhsa[i] += PAD[k - 1]*varsp[k - 1]*(varsy[i - 1] - varsa[i - 1] - varsp[
         i - 1])/add(varsy[j] - varsa[j] - varsp[j], j = 0 .. k - 1);
169      rhsa[i] += -PAD[k - 1]*varsp[k - 1]*varsa[i]/add(varsy[j], j = 0 .. k -
         1);
170      rhsa[i] += PAD[k - 1]*varsp[k - 1]*varsa[i - 1]/add(varsy[j], j = 0 .. k
          - 1);
171      rhsa[i] += PAD[k - 1]*varsp[k - 1]*(varsy[k - 1] - varsa[k - 1] - varsp[
         k - 1])*varsp[i]/(add(varsy[j], j = 0 .. k - 1)*add(varsy[j] - varsa[j] -
          varsp[j], j = 0 .. k - 1));
172      rhsa[i] += PAD[k - 1]*varsp[k - 1]*varsp[i]/add(varsy[j] - varsa[j] -
         varsp[j], j = 0 .. k - 1);
173  end do;
174  rhsa[k - 1] += PAD[k - 1]*varsp[k - 1]*(varsy[k - 1] - varsa[k - 1] - varsp[
         k - 1])/add(varsy[j], j = 0 .. k - 1);
175  rhsa[k] += varsp[k - 1]*PAD[k - 1]*(varsy[k - 1] - varsa[k - 1] - varsp[k -
         1])/add(varsy[j] - varsa[j] - varsp[j], j = 0 .. k - 1);
176  rhsa[k] += PAD[k - 1]*varsp[k - 1]*varsa[k - 1]/add(varsy[j], j = 0 .. k -
         1);
177  rhsa[k] += PAD[k - 1]*varsp[k - 1]*(varsy[k - 1] - varsa[k - 1] - varsp[k -
         1])*varsp[k]/(add(varsy[j], j = 0 .. k - 1)*add(varsy[j] - varsa[j] -
         varsp[j], j = 0 .. k - 1));
178  rhsa[k] += PAD[k - 1]*varsp[k - 1]*varsp[k]/add(varsy[j] - varsa[j] - varsp[
         j], j = 0 .. k - 1);
179  for i to k - 2 do
180      rhsp[i] += -PAD[k - 1]*varsp[k - 1]*varsp[i]/add(varsy[j], j = 0 .. k -
         1);
181      rhsp[i] += PAD[k - 1]*varsp[k - 1]*varsp[i - 1]/add(varsy[j], j = 0 .. k
          - 1);
182      rhsp[i] += -PAD[k - 1]*varsp[k - 1]*(varsy[k - 1] - varsa[k - 1] - varsp
         [k - 1])*varsp[i]/(add(varsy[j], j = 0 .. k - 1)*add(varsy[j] - varsa[j]
         - varsp[j], j = 0 .. k - 1));
183      rhsp[i] += -PAD[k - 1]*varsp[k - 1]*varsp[i]/add(varsy[j] - varsa[j] -
         varsp[j], j = 0 .. k - 1);
184  end do;
185  rhsp[k - 1] += -PAD[k - 1]*varsp[k - 1];
186  rhsp[k - 1] += -PAD[k - 1]*varsp[k - 1]*varsp[k - 1]/add(varsy[j], j = 0 ..
         k - 1);
187  rhsp[k - 1] += PAD[k - 1]*varsp[k - 1]*varsp[k - 2]/add(varsy[j], j = 0 .. k
          - 1);
188  rhsp[k - 1] += -PAD[k - 1]*varsp[k - 1]*(varsy[k - 1] - varsa[k - 1] - varsp
         [k - 1])/add(varsy[j], j = 0 .. k - 1)*varsp[k - 1]/add(varsy[j] - varsa[
         j] - varsp[j], j = 0 .. k - 1);
189  rhsp[k - 1] += -PAD[k - 1]*varsp[k - 1]*varsp[k - 1]/add(varsy[j] - varsa[j]
          - varsp[j], j = 0 .. k - 1);
190  rhsp[k] += PAD[k - 1]*varsp[k - 1]*varsp[k - 1]/add(varsy[j], j = 0 .. k -
         1);
191  rhsp[k] += -PAD[k - 1]*varsp[k - 1]*(varsy[k - 1] - varsa[k - 1] - varsp[k -
          1])*varsp[k]/(add(varsy[j], j = 0 .. k - 1)*add(varsy[j] - varsa[j] -
         varsp[j], j = 0 .. k - 1));
192  rhsp[k] += -PAD[k - 1]*varsp[k - 1]*varsp[k]/add(varsy[j] - varsa[j] - varsp
         [j], j = 0 .. k - 1);
193  print("Event 6.1 processed");
```

```
194
195 (* EVENTS 6.2.1 and 6.2.2 *)
196 rhsy[k - 1] += -2*varsp[k - 1]*(1 - PAD[k - 1]);
197 rhsy[k] += 2*varsp[k - 1]*(1 - PAD[k - 1]);
198 rhsa[k - 1] += -varsp[k - 1]*(1 - PAD[k - 1])*SigmaP(t);
199 rhsa[k] += varsp[k - 1]*(1 - PAD[k - 1])*SigmaP(t);
200 rhsp[k - 1] += -varsp[k - 1]*(1 - PAD[k - 1]) - varsp[k - 1]*(1 - PAD[k -
    1])*(1 - SigmaP(t));
201 rhsp[k] += varsp[k - 1]*(1 - PAD[k - 1]) + varsp[k - 1]*(1 - PAD[k - 1])*(1
    - SigmaP(t));
202 print("Events 6.2.1 and 6.2.2 processed");
203
204 (* EVENT 7 *)
205 rhsa[k] += -2*varsa[k];
206 rhsp[k] += 2*varsa[k];
207 print("Event 7 processed");
208
209 (* EVENT 8.1 *)
210 print("Event 8.1 processed");
211
212 (* EVENT 8.2 *)
213 rhsy[0] += -2*varsp[k]*(varsy[0] - varsa[0] - varsp[0])/add(varsy[j] - varsa
    [j] - varsp[j], j = 0 .. k - 1);
214 for i to k - 1 do
215     rhsy[i] += -2*varsp[k]*(varsy[i] - varsa[i] - varsp[i])/add(varsy[j] -
    varsa[j] - varsp[j], j = 0 .. k - 1);
216     rhsy[i] += 2*varsp[k]*(varsy[i - 1] - varsa[i - 1] - varsp[i - 1])/add(
    varsy[j] - varsa[j] - varsp[j], j = 0 .. k - 1);
217 end do;
218 rhsy[k] += 2*varsp[k]*(varsy[k - 1] - varsa[k - 1] - varsp[k - 1])/add(varsy
    [j] - varsa[j] - varsp[j], j = 0 .. k - 1);
219 for i to k do
220     rhsa[i] += 2*varsp[k]*(varsy[i - 1] - varsa[i - 1] - varsp[i - 1])/add(
    varsy[j] - varsa[j] - varsp[j], j = 0 .. k - 1);
221     rhsa[i] += 2*varsp[k]*varsp[i]/add(varsy[j] - varsa[j] - varsp[j], j = 0
     .. k - 1);
222 end do;
223 for i to k - 1 do
224     rhsp[i] += -2*varsp[k]*varsp[i]/add(varsy[j] - varsa[j] - varsp[j], j =
    0 .. k - 1);
225 end do;
226 rhsp[k] += -varsp[k];
227 rhsp[k] += -(2*varsp[k])*varsp[k]/add(varsy[j] - varsa[j] - varsp[j], j = 0
    .. k - 1);
228 print("Event 8.2 processed");
229
230 (* EVENT 9 *)
231 for i to k - 1 do
232         rhsa[i] += -varsa[i];
233         rhsp[i] += varsa[i];
234 end do;
235 print("Event 9 processed");
```

```
236
237 (* EVENT 10 *)
238 print("Event 10 processed");
239
240 (* EVENT 11 *)
241 print("Event 11 processed");
242
243 (* DEMOTION EVENT *)
244 for i to k - 1 do
245         rhsa[i] += -demotion_factor*varsa[i]/add(varsy[j] - varsa[j] - varsp
    [j], j = 0 .. k - 1);
246         rhsp[i] += -demotion_factor*varsp[i]/add(varsy[j] - varsa[j] - varsp
    [j], j = 0 .. k - 1);
247 end do;
248 print("Demotion Event processed");
249
250 (* Setting up ODEs *)
251 print("Setting up ODEs");
252 odesy := seq(diff(y || i(t), t) = rhsy[i], i = 0 .. k);
253 odesa := seq(diff(a || i(t), t) = rhsa[i], i = 0 .. k);
254 odesp := seq(diff(p || i(t), t) = rhsp[i], i = 0 .. k);
255 initial_vals_y := y0(0) = 1, seq(y || i(0) = 0, i = 1 .. k);
256 initial_vals_a := seq(a || i(0) = 0, i = 0 .. k);
257 initial_vals_p := seq(p || i(0) = 0, i = 0 .. k);
258
259 (* Solving ODE system *)
260 sol_system := dsolve([odesy, odesa, odesp, initial_vals_y, initial_vals_a,
    initial_vals_p], numeric, method = dverk78, maxfun = 0, abserr =
    0.1*10^(-7), relerr = 0.1*10^(-7));
261 with(plots);
262 plot_vars := [seq([t, y || i(t)], i = 0 .. k), seq([t, a || i(t)], i = 1 ..
    k), seq([t, p || i(t)], i = 1 .. k)];
263 odeplot(sol_system, plot_vars, 0 .. 2*k);
264 sol_system(2*k);
```