The deprioritised approach to prioritised algorithms

Stephen Howe

School of Mathematics and Statistics

The University of New South Wales

Doctor of Philosophy

2008

Abstract

Randomised algorithms are an effective method of attacking computationally intractable problems. A simple and fast randomised algorithm may produce results to an accuracy sufficient for many purposes, especially in the average case. In this thesis we consider average case analyses of heuristics for certain NP-hard graph optimisation problems. In particular, we consider algorithms that find dominating sets of random regular directed graphs. As well as providing an average case analysis, our results also determine new upper bounds on domination numbers of random regular directed graphs.

The algorithms for random regular directed graphs considered in this thesis are known as prioritised algorithms. Each prioritised algorithm determines a discrete random process. This discrete process may be continuously approximated using differential equations. Under certain conditions, the solutions to these differential equations describe the behaviour of the prioritised algorithm. Applying such an analysis to prioritised algorithms directly is difficult. However, we are able to use prioritised algorithms to define new algorithms, called deprioritised algorithms, that can be analysed in this fashion.

Defining a deprioritised algorithm based on a given prioritised algorithm, and then analysing the deprioritised algorithm, is called the deprioritised approach. The initial theory describing the deprioritised approach was developed by Wormald and has been successfully applied in many cases. However not all algorithms are covered by Wormald's theory: for example, algorithms for random regular directed graphs. The main contribution of this thesis is the extension of the deprioritised approach to a larger class of prioritised algorithms. We demonstrate the new theory by applying it to two algorithms which find dominating sets of random regular directed graphs.

Originality Statement

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I have worked at UNSW or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.

....

December 19, 2008

Acknowledgments

There are many people without whom I could not have written this thesis. Most significantly, my supervisor Catherine Greenhill who suggested my research topic. I am very grateful for her considerable effort and friendship.

My family, Mum, Dad, Zoe, and Luke, have always looked out for me which has made life so much easier. Thanks also to Fiona and Mark for providing me with a place to stay when I needed it.

Many people have encouraged me in my attempt to become a mathematician. Such encouragement makes a tremendous difference and I value it highly. I would also like to thank all my friends at UNSW for making my time as a PhD student enjoyable.

Contents

Chapter 1

Introduction

Randomisation, made practical by computers, is an important tool in modern mathematics. In this thesis, randomisation appears frequently as we consider randomised algorithms, that is, algorithms that rely on random choices, and random combinatorial structures. In particular, this thesis extends a technique of analysing certain randomised algorithms. As a demonstration of this new theory, we design and analyse such algorithms for random regular directed graphs with fixed degree.

Combinatorial structures, such as graphs and directed graphs, are effective at modeling real world phenomena. For example, graphs and directed graphs are used to model the world wide web [4, 22] and social networks [52]; they also appear in operations research [32] and game theory [51, 53]. Networks that arise in practice are often modeled by random graphs: for example, social networks are often modeled by scale-free random graphs. Random graphs are also frequently used to represent an average case. Some random graphs are of interest because they have useful properties: for example, random regular graphs have high connectivity and logarithmic diameter. So computer networks based on random regular graphs have good communication properties.

We are often interested in special substructures of combinatorial structures. Algorithms are commonly used to find such substructures. However, some substructures, especially those that are optimal (in some sense), are very difficult to find. So the corresponding algorithms may be impractical to use. In such cases, randomised algorithms have proved to be effective. Of course the use of randomisation comes at a cost: such algorithms may give incorrect or suboptimal results. But typically we are able to bound the probability that the output of such an algorithm is incorrect or is too far from optimal. The increase in speed and simplicity of randomised algorithms often makes up for their drawbacks.

Some randomised algorithms, mainly for graphs and satisfiability (SAT) instances [24, Chapter 34], have been successfully analysed using the differential equations method of Wormald [60]. This method gives an asymptotic analysis by letting, for example, the number of vertices of a graph tend to infinity. Then, under certain conditions, the asymptotic behaviour of an algorithm is approximated by solutions to systems of ordinary differential equations.

Differential equations may seem an unlikely way to analyse combinatorial algorithms, but they arise quite naturally. Indeed, differential equations have a long history as continuous approximations to discrete random processes. For example, the exponential models of population growth and of radioactive decay [18]. Consider a radioactive substance, which decays by emitting radiation randomly. Experiments suggest that the decay rate is proportional to the mass of the substance. So letting $m(t)$ be the mass of a radioactive substance at time t, we have

$$\frac{\mathrm{d}m}{\mathrm{d}t} = km \tag{1.0.1}$$

for some negative constant k. Solving (1.0.1) with the initial condition $m(0) = m_0$ we approximate the discrete random process of radioactive decay with the continuous function $m(t) = m_0 e^{kt}$.

The continuous approximations of the algorithms we consider arise as follows. Each algorithm starts with an empty directed graph and adds edges one at a time until certain conditions are satisfied. Thus we obtain sequences of random directed graph processes indexed by the number of vertices. Under certain conditions, as the number of vertices tends to infinity, the behaviour of such algorithm in some sense "settles down" and we can describe this behaviour by systems of differential equations. Thus the behaviour of the algorithms can be approximated by continuous functions.

The first such use of differential equations for random graphs is due to Karp and Sipser [46] (1981) who analysed an algorithm for finding maximum matchings in sparse random graphs. Karp and Sipser used an earlier theorem of Kurtz [47] (1970) which applies to discrete random processes. Subsequently, algorithms for satisfying random k-SAT instances were studied by Chao and Franco [20] (1986), and Frieze and Suen [33] (1996). In both cases, random processes defined by the algorithms were analysed using binomial random variables. A more successful technique for analysing such random processes, based on martingales, was introduced by Wormald [58] (1995). A similar approach is used by Aronson et al. [7] (1998) to obtain an improved analysis of the algorithm of Karp and Sipser. This technique is codified in a theorem of Wormald [60, Theorem 5.1] (1999) which we will refer to as the Differential Equations Theorem. The Differential Equations Theorem and variations thereof have been applied in many papers [2, 16, 27, 37, 45, 61] and a new variation appears later in this thesis as Theorem 5.1.1. We will use Theorem 5.1.1 to prove the main result of this thesis.

The algorithms considered in this thesis, known as prioritised algorithms, are similar to those mentioned above. We now briefly describe some aspects of prioritised algorithms. Prioritised algorithms proceed via a sequence of operations. Each operation has one of a finite number of types. Prioritised algorithms are so called because the type of the next operation performed is chosen according to a prioritisation of the types. This prioritisation causes the algorithms to perform well but also makes the analysis of prioritised algorithms difficult. So related algorithms which are easier to analyse have been sought.

The introduction of deprioritised algorithms by Wormald [61] was foreshadowed by work by Zito and (independently) Achlioptas. Zito [62] (2001) introduced a new class of algorithm; these algorithms use the same operations as prioritised algorithms but avoid prioritisation by using part of the input to specify the type of each operation of the algorithm. Zito was able to analyse such algorithms more easily than the corresponding prioritised algorithms. However evidence suggests that prioritised algorithms have superior performance. Meanwhile, Achlioptas [1] (2000) defined an algorithm that combines prioritisation with a random selection to determine the type of the next operation. Wormald [61] (2003) extended this idea by defining deprioritised algorithms which avoid prioritisation altogether.

Deprioritised algorithms choose the type of the next operation to perform randomly. In particular, deprioritised algorithms specify a probability distribution, which changes during the execution of the algorithm, that is used to select the type of the next operation. We attempt to define a deprioritised algorithm so that its behaviour is similar to the behaviour of a given prioritised algorithm.

Wormald has provided a general theory [61] for analysing deprioritised algorithms based on prioritised algorithms. This theory has been successfully applied in many cases [11, 28, 29, 31]. However, prioritised algorithms for random regular directed graphs have not been considered. (By regular we mean that for some positive r each vertex has in-degree r and out-degree r). Indeed, the theory provided by Wormald does not cover many natural algorithms for random regular directed graphs. Some algorithms on graphs are also not covered [25, 56]. So this thesis extends Wormald's theory to a larger class of prioritised algorithms, including algorithms for random regular directed graphs.

While the deprioritised approach is now a standard technique, prioritised algorithms are still analysed directly. For example, Bohman and Frieze [15] use martingales to analyse the Karp-Sipser algorithm for random graphs with a fixed degree sequence. Their method of analysis is similar to that used by Wormald to prove the Differential Equations Theorem. However, they also derive analytic properties of the corresponding differential equations. Using these properties, they analyse the algorithm until very near its end (nearer than is possible using the Differential Equations Theorem directly). Bohman and Frieze also avoid the need for numerical approximations. So while deprioritisation has proved effective, prioritisation is still important, notably for algorithms that either succeed or fail. For example, algorithms for satisfying random k-SAT instances [45] and colouring algorithms [55, 56] have used prioritisation. The latter algorithms, due to Shi and Wormald, are partially deprioritised (like the algorithm of Achlioptas mentioned above) in the sense that some operation types are chosen randomly and others are not.

We demonstrate the new theory presented in this thesis by applying it to algorithms for finding dominating sets of random regular directed graphs. Domination in graphs is a large and important subject area [40, 41]. One major application of dominating sets is in computer science: identifying a dominating set of a computer network enables efficient communication within that network [19, 48]. Previously the differential equations method has been used to find small dominating sets of random regular graphs [30, 31]. However, dominating sets of directed graphs have been studied little, despite their many applications (an example in game theory is discussed in Section 2.2). So it is natural to now apply the differential equations method to algorithms for finding (small) dominating sets of random regular directed graphs. The author has previously consider the case for 2-in 2-out digraphs [42].

We finish the introduction with an outline of the thesis. In the next chapter we introduce directed graphs and dominating sets of directed graphs; we also provide some relevant previous results. Chapter 2 provides the context in which we develop the theory of this thesis, however the main results are presented more generally. The third chapter introduces random directed graph models and describes prioritised and deprioritised algorithms. Chapter 3 also describes how differential equations arise in the analysis of prioritised and deprioritised algorithms. This chapter provides a overview of the more technical work that follows in Chapter 4 and Chapter 5.

In Chapter 4 and Chapter 5 we present the main theory of this thesis. Chapter 4 introduces prioritised and deprioritised algorithms in a general setting. The differential equations and their solutions, which describe the deprioritised algorithm, are also defined. Chapter 5 presents the theorem that allows us to analyse the deprioritised algorithms given in Chapter 4. Some time is also devoted to showing how this theorem may be applied. Quite a bit a notation is required for Chapter 4 and Chapter 5, so a symbol index is included after the final chapter. Applications of the theory in Chapter 4 and Chapter 5 are given in Chapter 6, where we analyse algorithms for finding certain types of dominating sets of random regular directed graphs. Chapter 7 concludes the thesis and discusses possibilities for future work.

Chapter 2

Directed Graphs and Dominating Sets

The theory presented in this thesis was developed to solve certain domination problems in directed graphs. So we now introduce some basic definitions and results relating to dominating sets of digraphs. These and further definitions and results can be found in a number of books on graphs and digraphs [8, 26, 35, 39], unless otherwise noted. The definitions of this chapter do not include random models of digraphs; although a particular model, that of random d-in d-out digraphs, is used throughout this thesis. Random d-in d-out digraphs and other related definitions are covered in the next chapter.

The content of this chapter divides into two parts. First we provide the definitions needed to understand the major results of this thesis (presented in Chapter 4 and Chapter 5) and the motivation leading to these results (given in Chapter 3). These definitions include digraphs, regular digraphs, and dominating sets. In the second part of this chapter we present definitions and results that are either used in applications of the theory of this thesis or are relevant and interesting in their own right. For example, we give new bounds on the minimum size of a dominating set of a digraph. We also define 2-path dominating sets, which are considered in Chapter 6. We conclude with a discussion of the complexity of finding dominating sets of digraphs.

2.1 Basic Definitions

Directed graphs often arise naturally in modeling real world phenomena. For example, directed graphs appear as models of web graphs [22] and in operations research [32]. Indeed, directed graphs arise naturally in the theory developed in this thesis, see Chapter 4. We consider only directed graphs. Definitions similar to those that follow can also be made for undirected graphs.

A *simple directed graph* or *digraph* G is a set $V = V(G)$ of vertices and a set $E = E(G)$ of (directed) edges where each edge is an element of

$$\{(u, v) \, : \, u, v \in V \text{ with } u \neq v\}.$$

Let $e = (u, v) \in E(G)$ be an edge of G. Then we say that e is an *edge from u to v*, that u and v are *adjacent*, and that u and v are *incident* with e. Notice that in a simple digraph no vertex is adjacent to itself and each edge occurs at most once. By removing these restrictions, that is, by allowing $E(G)$ to be multi-subset of $\{(u, v) \, : \, u, v \in V\}$ we obtain a *multi-digraph*. In a multi-digraph, edges of the form (u, u) are called loops while an edge that occurs more than once is called a *multiple edge* (or, more specifically, a *double edge* or a *triple edge* and so on).

When referring to simple digraphs we usually drop the word simple; that is, by digraph we always mean a simple digraph. The vertices of the graphs we consider are always labeled; usually 1 through n where n is the number of vertices. The definitions that follow all hold for both simple and multi-digraphs.

In order to define and analyse the algorithms considered in this thesis, we classify vertices of digraphs using the following definitions. For a vertex $u \in V$, the *in-neighbours* of u are vertices in the set

$$N_{\text{in}}(u) = \{v \in V \, : \, (v, u) \in E\},$$

while the *out-neighbours* of u are vertices in the set

$$N_{\text{out}}(u) = \{v \in V \, : \, (u, v) \in E\}.$$

We then define the *in-degree* of u to be $|N_{\text{in}}(u)|$ and the *out-degree* of u to be $|N_{\text{out}}(u)|$. We denote the in-degree and the out-degree of u by in-deg(u) and out-deg(u) respectively. The pair whose first component is the in-degree of u and whose second component is the out-degree of u is called the *degree pair* of u. We denote that degree pair of u by deg-pair(u); so deg-pair(u) $=$ (in-deg(u), out-deg(u)). If the degree pair of u is $(0, 0)$ then we say that u is *isolated*. Figure 2.1.1(a) shows the in-neighbours, out-neighbours, and degree pair of a vertex in a digraph. We can now define the class of digraphs in which we are most interested. For fixed positive integer d, a digraph G is *d-in d-out* or *regular* (of degree d) if every vertex of G has degree pair (d, d).

We now consider sequences of edges of digraphs. A *directed walk* is a sequence $v_1, e_1, v_2, e_2, \ldots, e_{m-1}, v_m$ of vertices v_i and edges e_i such that e_i is an edge from v_i to v_{i+1} (for $i = 1, \ldots, m-1$). Walks may contain repeated edges and vertices. The number of edges in a walk W is called the *length* of W. A walk for which each vertex is distinct is called a *directed path*, while a directed walk such that no edge is repeated and the only repeated vertex is $v_1 = v_m$ is called a *directed cycle*. A directed cycle that contains each vertex of $V(G)$ is called a *directed Hamilton cycle*. Any digraph containing a directed Hamilton cycle is called *Hamiltonian*. Next we consider special subsets of the vertices of a digraph.

2.1.1 Domination in digraphs

For a digraph G, a *dominating set* of G is a subset D of the vertices $V(G)$ such that, for each vertex $u \notin D$ there is an edge from some vertex in D to u. In general, for a vertex v and a set of vertices D, if there is an edge from some vertex in D to v then we say the D *dominates* v. So D is a dominating set if D dominates each vertex in $V \backslash D$. For undirected graphs it is natural to consider independent dominating sets, that is, dominating sets in which no two vertices are adjacent. However, some digraphs do not have an independent dominating set (see Figure 2.1.1(b)), so we do not consider independent dominating sets of digraphs.

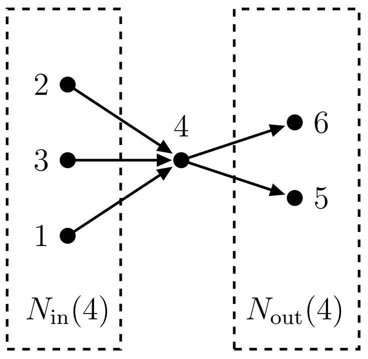
The most interesting dominating sets are those of minimum cardinality. A minimum dominating set of a random 2-in 2-out digraph is shown in Figure 2.1.1(c). In particular we are interested in the size of such dominating sets. For a digraph G we denote the minimum size of a dominating set of G by $\overrightarrow{\gamma}(G)$ and call $\overrightarrow{\gamma}(G)$ the *domination number* of the digraph G. Determining domination numbers exactly is difficult so we usually look for upper and lower bounds. In Chapter 6 we use the theory developed in this thesis to determine upper bounds on the domination numbers of random d-in d-out digraphs. These bounds will be determined by letting the number of vertices n of such digraphs tend to infinity; so next we introduce the required asymptotic notation. Note that we always take d to be fixed, that is, d never depends on n. So we do not consider d-in d-out digraphs where d tends to infinity with n, for example.

2.1.2 Asymptotic notation

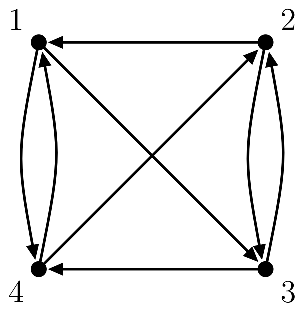
We use n to denote the number of vertices of a given digraph. To describe properties of digraphs as the number of vertices tends to infinity, we use the usual big-O and little-o notation. Let $\{a_n\}$, $\{b_n\}$, and $\{c_n\}$ be sequences of real numbers such that each c_n is positive. Then we write

- $a_n = O(c_n)$ if there exists positive numbers C and N such that $|a_n| \leq Cc_n$ for all $n > N$,
- $a_n = \Omega(c_n)$ if there exists positive numbers C and N such that $|a_n| \geq Cc_n$ for all $n > N$, and
- $a_n = o(c_n)$ if $\lim_{n\to\infty} a_n/c_n = 0$.

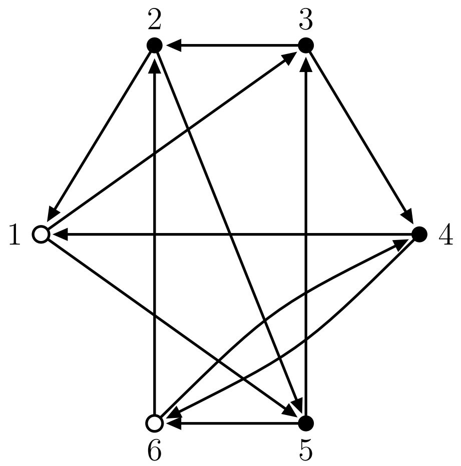
Occasionally we consider equations of the form $a_n = b_n + o(c_n)$. By such an equation we mean that there exists a positive sequence $\{f_n\}$ such that $|a_n - b_n| \leq f_n$ and that $f_n = o(c_n)$. Graham et al. [36, Chapter 9] give a detailed introduction to asymptotic notation.



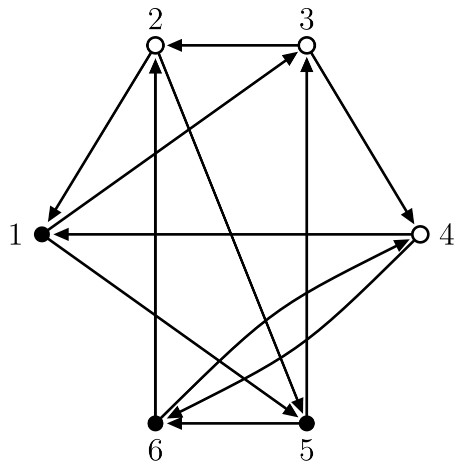
(a) A digraph in which vertex 4 has degree pair $(3, 2)$.



(b) A digraph with no independent dominating set.



(c) The set $\{1, 6\}$ is a minimum dominating set of the above 2-in 2-out digraph.



(d) The set $\{2, 3, 4\}$ is a minimum 2-path dominating set of the above 2-in 2-out digraph.

Figure 2.1.1: Examples of directed graphs

2.2 More Definitions

We now present some more definitions relating to, and some basic results about, domination in digraphs. However, domination is not the focus of this thesis and the definitions and results that follow are not required to understand the most important parts of this thesis, namely Chapter 4 and Chapter 5.

The definition of dominating sets for digraphs naturally suggests some related definitions. In particular, if we reverse the direction of the required edge we obtain the definition of an *absorbent set.* That is, a subset $A \subseteq V(G)$ of the vertices of a digraph G is *absorbent* if for each vertex $u \notin A$, there is an edge from u to some vertex in A. The minimum size of an absorbent set of a digraph G is called the *absorption number* and denoted by $\overleftarrow{\gamma}(G)$.

It is natural to consider a set B that is both dominating and absorbent. Figure 2.1.1(d) shows such a set for a random 2-in 2-out digraph. Notice that each vertex not in B lies on a directed path (or cycle) of length two between vertices of B. So we call B a *2-path dominating* set. As far as we are aware, this definition has not previously been made. The minimum size of a 2-path dominating set is called the *2-path domination number* and denoted by $\overrightarrow{\gamma}_{\text{path}}(G)$. We focus mainly on the domination number and 2-path dominating number.

Absorbent sets and dominating sets are closely related. As suggested by the symmetry of the definitions, results on the domination number tranfer easily to the absorption number. For a digraph G, the *reversal* of G is the digraph G^{-1} with the same vertex set as G and the edges $\{(u, v) : (v, u) \in G\}$. If D is a dominating set of G then D is an absorbent set of G^{-1}, and vice-versa. Moreover, if D is a minimum dominating set of G then D is a minimum absorbent set of G^{-1}. Clearly, G and G^{-1} have exactly the same 2-path dominating sets.

We now briefly describe an application of domination in digraphs. Digraphs occur naturally in game theory with independent absorbent sets (usually called *kernels*) playing an important role. Many papers have been published on this topic [14, 51, 53]. We now, as an example, consider a two player game (with players Alice and Bob) in which both players can make the same moves and the last player to move wins. We may represent such a game by a digraph.

We define a digraph G on a vertex set where each vertex represents a possible state of the game. Then the edges of G are such that there is an edge from vertex u to vertex v if a player can change the state of the game from u to v in one move. Playing a game corresponds to a directed walk on G; we call this walk the *game walk*. Any vertex with out-degree 0 is called *terminal* and these vertices represent game states from which no move is possible. So a player wins if they extend the game walk to a terminal vertex.

Now let K be an independent absorbent set of G. Then every terminal vertex lies in K. Assume that Alice extends the game walk to a vertex of K. Then Alice need not lose. Since K is independent, Bob must extend the game walk to a vertex not in K. This vertex cannot be a terminal vertex and Bob cannot win with this move. Now, since K is absorbent, Alice can extend the game walk back to a vertex in K. Thus Bob can never win the game. Because of such applications, kernels in digraphs have received much attention [35, Section 15.4].

Next we consider some basic results on dominating sets, absorbent sets, and 2-path dominating sets.

2.2.1 Basic results on domination in digraphs

The following results bound the domination, absorption, and 2-path domination numbers using in-degrees and out-degrees. Throughout this section we let δ_{in} and δ_{out} be the minimum in-degree and out-degree of a given digraph, respectively. Likewise, we let Δ_{in} and Δ_{out} be the maximum in-degree and out-degree, respectively. The first result is due to Lee [49].

Theorem 2.2.1 ([49, Theorem 1.2.6]). *Let G be a digraph on n vertices with minimum in-degree δ_{in} at least one. Then*

$$1 \leq \overrightarrow{\gamma}(G) \leq \frac{\delta_{\text{in}} + 1}{2\delta_{\text{in}} + 1} n.$$

Notice that the above bound tends to $n/2$ as δ_{in} tends to infinity. The next theorem, again by Lee [49], gives an upper bound that is $o(n)$ as δ_{in} tends to infinity and improves upon Theorem 2.2.1 except when $\delta_{\text{in}} = 1$ or $\delta_{\text{in}} = 2$.

Theorem 2.2.2 ([49, Theorem 1.2.1]). *Let G be a digraph on n vertices with minimum in-degree δ_{in} at least one. Then*

$$\overrightarrow{\gamma}(G) \leq \left(1 - \left(\frac{1}{1+\delta_{\text{in}}}\right)^{\frac{1}{\delta_{\text{in}}}} + \left(\frac{1}{1+\delta_{\text{in}}}\right)^{\frac{1+\delta_{\text{in}}}{\delta_{\text{in}}}}\right) n.$$

The following result, due to Ghoshal et al. [35], gives bounds for all digraphs based on the maximum out-degree.

Theorem 2.2.3 ([35, Theorem 15.57]). *Let G be a digraph on n vertices. Then*

$$\frac{n}{1 + \Delta_{\text{out}}} \leq \overrightarrow{\gamma}(G) \leq n - \Delta_{\text{out}}.$$

The above bounds are sharp. Let $K_{a,b}$ (for integers $a, b \geq 1$) be the digraph on the vertices $\{1, \ldots, a + b\}$ with the edge set

$$\{(u, v) : 1 \leq u \leq a,\ a + 1 \leq v \leq a + b\}.$$

Then $\Delta_{\text{out}}(K_{a,b}) = b$ and the vertices $\{1, \ldots, a\}$ are a minimum dominating set for $K_{a,b}$. Hence $\overrightarrow{\gamma}(K_{n-b,b}) = n - \Delta_{\text{out}}(K_{n-b,b}) = n - b$ achieves the upper bound and

$$\overrightarrow{\gamma}(K_{1,b}) = \frac{1 + b}{1 + \Delta_{\text{out}}(K_{1,b})} = 1$$

achieves the lower bound of Theorem 2.2.3.

Clearly a digraph must satisfy some strong conditions to have small dominating sets. For a connected undirected graph, the domination number is at most half the number of vertices. To obtain a similar bound for a digraph G we require G to be *strongly connected*; that is, we require that for every pair of distinct vertices u and v of G, there exists a directed path from u to v and from v to u. Then we have the following theorem, again due to Lee [49].

Theorem 2.2.4 ([49, Corollary 1.2.11]). *Let G be a strongly connected digraph on n vertices. Then*

$$\overrightarrow{\gamma}(G) \leq \left\lceil \frac{n}{2} \right\rceil.$$

So, for instance, the bound in Theorem 2.2.4 applies if G is Hamiltonian since Hamiltonian digraphs are strongly connected. These are the most important (for us) of the known results for the domination number of a directed graph. We next consider equivalent bounds for the absorption number.

As mentioned above, a minimum absorbent set of G is a minimum dominating set of the reversal of G. Thus we may apply the theorems above to obtain bounds on $\overleftarrow{\gamma}(G)$. In particular we have the following two theorems.

Theorem 2.2.5. *Let G be a digraph on n vertices with minimum out-degree δ_{out} at least one. Then*

$$1 \leq \overleftarrow{\gamma}(G) \leq \min\left\{\frac{\delta_{\text{out}}+1}{2\delta_{\text{out}}+1}, 1 - \left(\frac{1}{1+\delta_{\text{out}}}\right)^{\frac{1}{\delta_{\text{out}}}} + \left(\frac{1}{1+\delta_{\text{out}}}\right)^{\frac{1+\delta_{\text{out}}}{\delta_{\text{out}}}}\right\} n.$$

Theorem 2.2.6. *Let G be a digraph on n vertices. Then*

$$\frac{n}{1+\Delta_{\text{in}}} \leq \overleftarrow{\gamma}(G) \leq n - \Delta_{\text{in}}.$$

Of course, the reversal of a strongly connected digraph is also strongly connected. So if a digraph G is Hamiltonian (or just strongly connected) then $\overleftarrow{\gamma}(G) \leq \left\lceil \frac{n}{2} \right\rceil$.

We now present some original bounds on the 2-path domination number. The lower bounds given above for the domination and absorption number also hold for the 2-path domination number. For upper bounds we obtain the following result by considering directed paths. The proof is straightforward and so is omitted.

Lemma 2.2.7. *Let G be a digraph on n vertices containing a directed path or cycle of length L. Then*

$$\overrightarrow{\gamma}_{\text{path}}(G) \le n - \left\lfloor \frac{L}{2} \right\rfloor .$$

In particular, if G is Hamiltonian, then

$$\overrightarrow{\gamma}_{\text{path}}(G) \le \left\lceil \frac{n}{2} \right\rceil .$$

To obtain good upper bounds on the 2-path dominating set, we look for a result similar to Theorem 2.2.2. We only consider d-in d-out digraphs since in other cases the generalisation of Theorem 2.2.2 is quite difficult. Fortunately d-in d-out digraphs are the digraphs in which are most interested.

Theorem 2.2.8. *Let G be a d-in d-out digraph with $d \ge 1$. Then*

$$\overrightarrow{\gamma}_{\text{path}}(G) \le \left(1 - \left(\frac{1}{2d+1}\right)^{\frac{1}{d}} + 2\left(\frac{1}{2d+1}\right)^{\frac{d+1}{d}} - \left(\frac{1}{2d+1}\right)^{\frac{2d+1}{d}}\right) n.$$

Proof. We use a common technique known as the probabilistic method [5, 6, 49, 54]. Fix p with $0 < p < 1$. Define a set $S \subseteq V$ such that for each vertex $v \in V$ we have $v \in S$ with probability p independently of the other vertices. Let T be the set of vertices not in S with either no in-neighbour in S or no out-neighbour in S. Then $S \cup T$ is a 2-path dominating set of G.

We now determine an upper bound on, $\mathbb{E}(|T|)$, the expected size of T. Note that we use $\mathbb{E}(\cdot)$ to denote expectation and $\mathbb{P}(\cdot)$ to denote probability. Let χ_v be the indicator variable for $v \in T$. Then

$$\mathbb{E}(|T|) = \sum_{v \in V} \mathbb{E}(\chi_v) = \sum_{v \in V} \mathbb{P}(v \in T).$$

Now $v \in T$ if and only if each of the following are satisfied:

- $v \notin S$,
- $N_{\text{in}}(v) \cap N_{\text{out}}(v) \cap S = \emptyset$, and
- $(N_{\text{in}}(v) \backslash N_{\text{out}}(v) \cap S = \emptyset)$ or $(N_{\text{out}}(v) \backslash N_{\text{in}}(v) \cap S = \emptyset)$.

d	2	3	4	5	6
upper bound	$0.71379n$	$0.61594n$	$0.54383n$	$0.48840n$	$0.44433n$

Table 2.2.1: Upper bounds on the 2-path domination number of a d-in d-out digraph on n vertices.

So let $|N_{\text{in}}(v) \cap N_{\text{out}}(v)| = j$. Then

$$\begin{aligned}\mathbb{P}(v \in T) &= (1-p)^{j+1}\left(2(1-p)^{d-j} - (1-p)^{2(d-j)}\right)\\ &= 2(1-p)^{d+1} - (1-p)^{2d+1}\left(\frac{1}{1-p}\right)^{j}\\ &\leq 2(1-p)^{d+1} - (1-p)^{2d+1}\end{aligned}$$

and so

$$\mathbb{E}(|S \cup T|) \leq \left(p + 2(1-p)^{d+1} - (1-p)^{2d+1}\right) n. \tag{2.2.1}$$

The right hand side of (2.2.1) is minimised by taking

$$p = 1 - \left(\frac{1}{2d+1}\right)^{\frac{1}{d}}.$$

Substituting this value for p into (2.2.1) we obtain the desired result. □

Table 2.2.1 gives upper bounds on the 2-path domination number of a d-in d-out digraph obtained from Theorem 2.2.8. This completes our survey of bounds on the domination, absorption, and 2-path domination numbers of directed graphs. Next we consider how regular digraphs are related to regular bipartite undirected graphs.

Domination in regular bipartite graphs

A *bipartite graph* is an undirected graph whose vertices can be partitioned into two independent sets, called *parts*. If a bipartite graph is regular, then its parts must be the same cardinality. Regular bipartite graphs are closely related to regular directed graphs. In particular, some bounds on the domination number of d-in d-out digraphs obtained in the previous section also determine bounds on the domination number of regular bipartite graphs.

Consider a bipartite graph B, regular of degree $(d+1)$, with two parts S and T, each of size n. Regular bipartite graphs necessarily have a perfect matching [26, Corollary 2.1.3], so let M be a perfect matching of B. Direct all the edges of B from S to T and then contract the edges in M. The result is a (simple) d-in d-out digraph $G(B)$ on n vertices. Note that B is a graph on $2n$ vertices.

Let P be a 2-path domination set of the digraph $G(B)$. Each vertex in P corresponds to two vertices in B via the perfect matching M. Let D be the set of vertices of B that correspond to the vertices in P; so $|D| = 2\,|P|$. Then D is a dominating set of B. Hence we have the following corollary to Theorem 2.2.8.

Corollary 2.2.9. *Let B be a $(d+1)$-regular bipartite graph on $2n$ vertices. Then the domination number $\gamma(B)$ of B satisfies*

$$\gamma(B) \leq \left(1 - \left(\frac{1}{2d+1}\right)^{\frac{1}{d}} + 2\left(\frac{1}{2d+1}\right)^{\frac{d+1}{d}} - \left(\frac{1}{2d+1}\right)^{\frac{2d+1}{d}}\right) n.$$

So Table 2.2.1 also gives upper bounds for the domination number of $(d+1)$-regular bipartite graphs on $2n$ vertices. Further results should also be possible, but we do not pursue them here. Instead we finish the chapter by considering the complexity of finding a dominating set of minimum size.

2.2.2 Complexity results

Finding a dominating set of minimum size in an undirected graph is NP-hard [34]. Of course any graph can be represented as a digraph by replacing each edge $\{u, v\}$ by the two edges (u, v) and (v, u). Hence the problems of finding minimum dominating, absorbent, and 2-path dominating sets of directed graphs are also NP-hard problems.

In general we expect the the complexity results in digraphs to be similar to those in undirected graphs. However Chlebík and Chlebíková [21] note a significant difference. While every undirected graph has an independent dominating set, it is NP-complete to determine whether an arbitrary digraph has an independent dominating set. Some approximation results are also given by Chlebík and Chlebíková

[21]. Complexity results for some special classes of dominating sets have been obtained by Barkauskas and Host [10] and Bar-Yehuda and Vishkin [9].

This completes our introduction to digraphs and dominating sets. In the next chapter we consider random directed graphs and dominating sets; we also describe prioritised and deprioritised algorithms, and how differential equations arise in their analysis.

Chapter 3

Random Digraphs and Algorithms

In this chapter we introduce the uniform model of random d-in d-out digraphs. We call a uniformly distributed random d-in d-out digraph simply a *random d-in d-out digraph* (that is, we do not explicitly mention the distribution). We always take d to be some fixed integer greater than one. To determine properties of random d-in d-out digraphs, a related random model called the pairing model is often used. We demonstrate the pairing model by determining lower bounds on the domination numbers of random d-in d-out digraphs for some d. However, the main focus of this chapter is the design and analysis of randomised algorithms for random d-in d-out digraphs. As an example we consider an algorithm that finds small dominating sets of random 2-in 2-out digraphs.

3.1 Random Regular Digraphs

There are many models of random d-in d-out digraphs but we are mainly interested in the uniform model. Consider the set of all labeled d-in d-out digraphs on the vertex set $\{1, \dots, n\}$. By choosing a digraph randomly from this set, with each digraph having an equal chance of being selected, we obtain a uniformly distributed random d-in d-out digraph. We denote this model by $\mathcal{DG}_{n,d}$ and call $\mathcal{DG}_{n,d}$ the *random d-in d-out digraph* (on the vertex set $\{1, \dots, n\}$).

Since we are working with random digraphs, the results we obtain will nearly always involve probabilities. We denote the probability of an event A by $\mathbb{P}(A)$. In this context the underlying sample space is the (finite) set of d-in d-out digraphs on the vertex set $\{1, \ldots, n\}$. Hence an event is a set of digraphs and, as the model is uniform, the probability of an event is just the proportion of digraphs which belong to that event. However we prefer the probabilistic notation. We often define random variables on random digraphs. Mostly we are interested in the expected values of random variables, which we denote by $\mathbb{E}(\cdot)$. Note that each event and random variable is indexed by n, the number of vertices, though usually we do not make this explicit. Finally, we often randomly select an element from a given set using the uniform distribution. When we make such a selection, we say that we are selecting *uniformly at random* (u.a.r.).

There are many properties of digraphs which are not satisfied by every digraph (on the vertex set $\{1, \ldots, n\}$), but the proportion of digraphs satisfying the property tends to 1 as the number of vertices tends to infinity. In this case we say that the property holds *asymptotically almost surely* (a.a.s.). In probabilistic notation, let A_n be the event that a random d-in d-out digraph $G \in \mathcal{DG}_{n,d}$ has some property P_n. For example, P_n could be the property that G has a directed Hamilton cycle. Then a random d-in d-out digraph has property P_n asymptotically almost surely if $\mathbb{P}(A_n) \to 1$ as $n \to \infty$. In our example, a random d-in d-out digraph a.a.s. has a directed Hamilton cycle when $d \geq 3$ but when $d = 2$ a.a.s. does not [23]. Later in this thesis we determine a.a.s. bounds on the domination number of random d-in d-out digraphs.

There are many techniques for studying random d-in d-out digraphs, including switchings, direct expectation arguments, randomised algorithms, and pairing models. These techniques may also be applied to random regular graphs and Wormald [59] has provided a survey of many such applications. Pairing models are closely related to uniform models of random d-in d-out digraphs and we introduce them next.

3.1.1 Pairing models

Pairing models, although not always by that name, have a long history in the study of random regular graphs and digraphs. They first appeared implicitly [12, 13, 57], before being given explicitly by Bollobás [17], who called them *configuration models*. We use the name *pairing model*, another common name. Pairing models provide an indirect way of studying random d-regular graphs and digraphs on n vertices. We define the pairing model for $\mathcal{DG}_{n,d}$ only; the pairing model for $\mathcal{G}_{n,d}$ is similar and a detailed explanation is given by Wormald [59].

The pairing model for $\mathcal{DG}_{n,d}$ has previously appeared [44, 50], with varying terminology. We take two sets of nd points; points of one set are called *in-points* and points of the other set are called *out-points*. With each vertex in the set $\{1, \dots, n\}$, we associate d in-points and d out-points so that each point is associated with exactly one vertex. A *pairing* is then a partition of the set of points (both in and out) into nd blocks of size 2 such that each block contains exactly one in-point and exactly one out-point. Each block of the pairing is called a *pair*.

For each pairing P there is a corresponding d-in d-out multi-digraph $DG(P)$. The digraph $DG(P)$ is the digraph such that, for every pair $\{p_{\text{in}}, p_{\text{out}}\}$ of P (where p_{in} is the in-point and p_{out} is the out-point), $DG(P)$ contains an edge from the vertex associated with p_{out} to the vertex associated with p_{in}. Distinct pairs may correspond to the same edge and in this case $DG(P)$ has a multiple edge. If p_{in} and p_{out} are associated with the same vertex then $DG(P)$ has a loop. Thus $DG(P)$ may be a multi-digraph. Of course, distinct pairings may correspond to the same multi-digraph; in particular, there are exactly $(d!)^{2n}$ pairings corresponding to each simple d-in d-out digraph. So from a pairing we obtain a d-in d-out multi-digraph.

By choosing a random pairing we obtain a random d-in d-out multi-digraph. Let $\mathcal{DP}_{n,d}$ denote the uniform probability space on the set of all pairings for a d-in d-out digraph on n vertices. Then for a random pairing $P \in \mathcal{DP}_{n,d}$, the corresponding digraph $DG(P)$ is a random d-in d-out multi-digraph distributed non-uniformly.

However, conditioning on $DG(P)$ having no loops and no multiple edges we obtain the uniform model $\mathcal{DG}_{n,d}$. (If we condition on $DG(P)$ having no multiple edges only, then we also obtain a uniform model. However, digraphs with multiple edges obtained via the pairing model are not distributed uniformly.) So to study the uniform model $\mathcal{DG}_{n,d}$ we often use the pairing model $\mathcal{DP}_{n,d}$.

Results for pairing models can translate to results for random d-in d-out digraphs. Let Σ_n be the set of pairings corresponding to the simple d-in d-out digraphs on the vertex set $\{1, \ldots, n\}$. Applying a theorem of McKay [50, Theorem 4.6] we find that, for fixed d, we have

$$\mathbb{P}_{\mathcal{DP}_{n,d}}(\Sigma_n) = \exp\left(-\frac{d^2+1}{2} + O\left(\frac{1}{n}\right)\right).$$

That is, the proportion of pairings corresponding to simple d-in d-out digraphs is bounded below by a positive constant. Thus any property that holds a.a.s. in the pairing model $\mathcal{DP}_{n,d}$, also holds a.a.s. for random d-in d-out digraphs.

The pairing model for regular digraphs is very similar to the pairing model for regular bipartite graphs. The two models only differ in how points are associated with vertices: for regular digraphs each vertex is associated with two sets of points, the in-points and the out-points; while for regular bipartite graphs there are twice as many vertices and each vertex associated with only one set of points. Thus we expect results, similar to Corollary 2.2.9, relating random regular digraphs and random regular bipartite graphs.

For the rest of this thesis, we work solely with the pairing model. In particular, we determine a.a.s. upper and lower bounds on the domination numbers of random d-in d-out digraphs (for some d) by showing that these bounds hold a.a.s. in the corresponding pairing model. As an introduction to the pairing model, we determine some lower bounds in the next section. We then consider randomised algorithms for random d-in d-out digraphs.

3.1.2 Lower bounds on domination numbers

We now determine a.a.s. lower bounds on domination numbers of random d-in d-out digraphs, for fixed d, using the pairing model $\mathcal{DP}_{n,d}$. Following an approach of Zito [62], we use direct expectation arguments and generating functions. Let $N(k)$ be the number of dominating sets of size k of a random d-in d-out digraph. Note that any set of vertices containing a dominating set is also a dominating set. Hence the probability that a random d-in d-out digraph has a dominating set of size at most k is

$$\mathbb{P}\left(\left[\sum\nolimits_{k^\star \leq k} N(k^\star)\right] \geq 1\right) = \mathbb{P}(N(k) \geq 1) \leq \mathbb{E}(N(k)),$$

by Markov's inequality. Thus any k with $\mathbb{E}(N(k)) = o(1)$ is an a.a.s. lower bound on the domination number.

Consider a subset D of the vertices with size k. Let P be a pairing and let O be the out-points associated with the vertices in D. Define

$$I = \{p_{\text{in}} : \{p_{\text{in}}, p_{\text{out}}\} \in P \text{ for some } p_{\text{out}} \in O\}.$$

Then D is a dominating set if, for each vertex $v \notin D$, some in-point associated with v is in I.

We use generating functions to determine the number of pairings for which D is a dominating set. First $(1+x)^d$ is the generating function for choosing subsets of d in-points and $(1+x)^d - 1$ is the generating function for choosing non-empty subsets. Each of the $n-k$ vertices not in D must have at least one of their associated in-points in I. The vertices in D may have any number of their associated in-points in I. So the generating function for the number of ways to choose the set I of in-points such that D is a dominating set is

$$((1+x)^d - 1)^{n-k}(1+x)^{dk}. \tag{3.1.1}$$

The vertices of D are associated with dk in-points, so we are interested in the coefficient of x^{dk} in (3.1.1). This coefficient is no greater than

$$\frac{((1+x)^d - 1)^{n-k}(1+x)^{dk}}{x^{dk}}, \quad \text{for all } x > 0.$$

d	2	3	4	5	6
lower bound	$0.34960n$	$0.27602n$	$0.23115n$	$0.20049n$	$0.17801n$

Table 3.1.1: A.a.s. lower bounds on the domination number of a random d-in d-out digraph.

Now, having chosen the in-points of I there are $(dk)!$ ways to put the points of O and I into pairs. The remaining $d(n-k)$ in-points and out-points can be arranged in pairs in $(d(n-k))!$ ways. Finally there are $\binom{n}{k}$ ways to choose D and $(dn)!$ possible pairings. Hence

$$\mathbb{E}(N(k)) \leq \binom{n}{k}\binom{dn}{dk}^{-1}\frac{((1+x)^d-1)^{n-k}(1+x)^{dk}}{x^{dk}} \tag{3.1.2}$$

for all $x > 0$.

Applying Stirling's approximation to the right hand side of (3.1.2), and letting $\phi(w) = w^w$ and $k = \kappa n$, a.a.s. we have

$$\mathbb{E}(N(k))^{1/n} \leq [\phi(\kappa)\phi(1-\kappa)]^{d-1}\frac{((1+x)^d-1)^{1-\kappa}(1+x)^{d\kappa}}{x^{d\kappa}} \tag{3.1.3}$$

for all $x > 0$. For fixed κ, we evaluate the right hand side of (3.1.3) by choosing $x > 0$ so as to minimise

$$\frac{((1+x)^d-1)^{1-\kappa}(1+x)^{d\kappa}}{x^{d\kappa}}.$$

By finding κ and x for which the right hand side of (3.1.3) is less than 1, we obtain the lower bounds given in Table 3.1.2.

Next we describe a class of randomised algorithms for random d-in d-out digraphs.

3.2 Algorithms and the Pairing Process

We now describe how we define and analyse algorithms for random d-in d-out digraphs and give a specific example of an algorithm that finds dominating sets. Each algorithm we consider is defined using the pairing model given in Section 3.1.1.

Instead of selecting a pairing uniformly at random in one go, we now construct a pairing one pair at a time. By constructing the pairing in an appropriate way, we still obtain a pairing that is uniformly distributed. The random process that constructs a uniformly distributed random pairing is called the *pairing process*, which we define next.

The pairing process proceeds as follows. We start with the empty set P_0. At each step t, to obtain P_{t+1} we add a pair $\{p_{\text{in}}, p_{\text{out}}\}$ to P_t, such that neither p_{in} nor p_{out} is already present in a pair of P_t. The process ends at step F when P_F is a pairing. During the process, if a point does not occur in a pair of P_t we say the point is *free*; otherwise we say the point has been *exposed.* Adding the pair $\{p_{\text{in}}, p_{\text{out}}\}$ to P_t is called *exposing* p_{in} (or p_{out}, or the pair $\{p_{\text{in}}, p_{\text{out}}\}$, or the edge to which $\{p_{\text{in}}, p_{\text{out}}\}$ corresponds). To complete the definition of the pairing process we next describe how $\{p_{\text{in}}, p_{\text{out}}\}$ may be chosen.

The key to the pairing process is the manner in which the next pair is chosen. There are two allowable ways: either the in-point p_{in} is chosen arbitrarily and the out-point p_{out} is chosen u.a.r. from the set of free out-points; or the out-point is chosen arbitrarily and the in-point is chosen u.a.r. from the set of free in-points. By choosing the next pair in one of these two ways, the final pairing P_F will be distributed uniformly.

We use the pairing process to define algorithms on random d-in d-out digraphs. By specifying either the in-point or the out-point of the next pair to be exposed we obtain an algorithm that generates a random pairing and thus a random d-in d-out multi-digraph. Choosing our specification carefully we obtain an algorithm that generates a random d-in d-out multi-digraph G and, at the same time, identifies a dominating set for G. Of course, many other special subsets of the vertices and edges can also be identified.

These algorithms can seem unnatural. However they can be easily modified to obtain algorithms that find a dominating set of a given d-in d-out digraph. In this case the digraph is part of the input to the algorithm, rather than part of the output.

Instead of adding pairs, such an algorithm proceeds by removing edges of the input graph. In this context, analysing the algorithm on a random d-in d-out digraph can be considered an average case analysis of the algorithm on the non-random digraph.

Before considering an example algorithm, we need some further definitions relating to the pairing process. The pairing process defines a random sequence of sets of pairs $P_0, \dots, P_F$, starting with the empty set P_0 and ending with a pairing P_F. For each P_t we obtain a digraph $G_t = DG(P_t)$ (as in Section 3.1.1). So the pairing process also defines a random sequence of multi-digraphs $G_0, \dots, G_F$ where G_0 is the empty digraph and G_F is a d-in d-out multi-digraph.

Algorithms for random d-in d-out digraphs are analysed by considering the degree pairs of the vertices during the algorithm. So we define $V_{(i,j)}(t) = V_{(i,j)}(G_t)$ to be the set of vertices of degree pair (i, j) in G_t. Note that if a vertex v of G_t has degree pair (i, j) then, in P_t, of the in-points associated with v exactly i are exposed and $d - i$ are free, and of the out-points associated with v exactly j are exposed and $d - j$ are free. If all points associated with a vertex v are exposed, and so v has degree pair (d, d) in G_t, then we say that the vertex is *saturated*. If all the points associated with v are free, and so v has degree pair $(0, 0)$ in G_t, then we say that v is *isolated*. We also define $Y_{(i,j)}(t) = \left|V_{(i,j)}(t)\right|$ to be the random variable counting the numbers of vertices of degree pair (i, j). These random variables will be used for every algorithm we analyse. The definitions of this section are used for the algorithms defined in the next section and in Chapter 6.

3.2.1 An example algorithm

Algorithms 1 and 2 define the algorithm DominatingSet2 which finds a dominating set of a random 2-in 2-out digraph (note that DominatingSet2 has previously been defined and analysed in preliminary work [42] for this thesis). DominatingSet2 starts with empty sets P_0 and D which become a pairing and a dominating set (for the digraph corresponding to the pairing) respectively when the algorithm finishes.

DominatingSet2 achieves this by repeatedly performing the following three steps: it selects a vertex u u.a.r. from all vertices of a given degree pair; exposes the free points of u and the new out-neighbours of u (determined by exposing the out-points of u); adds u and any vertices, other than the out-neighbours of u, that become saturated during the previous step to D (vertices other than u that are added to D at this stage are called *accidental saturates*). The three steps outlined above are called an *operation.* Notice that after each operation every saturated vertex is either in D or is an out-neighbour of a vertex in D. This invariant ensures that DominatingSet2 returns a dominating set as expected. To complete the description of DominatingSet2 we must look at the first step of an operation in more detail.

In the first step of each operation, excluding the first operation, a vertex of degree pair $(1, 0)$, $(2, 0)$, or $(2, 1)$ is selected. The first operation, which occurs before the while loop, selects a vertex of degree pair $(0, 0)$. DominatingSet2 proceeds via a sequence of operations until there are no vertices of degree pairs $(1, 0)$, $(2, 0)$, or $(2, 1)$. At this point, as the sum of the in-degrees must equal the sum of the out-degrees, we see that there are vertices of degree pairs $(0, 0)$, $(1, 1)$, and $(2, 2)$ only. The last two steps of DominatingSet2 ensure a dominating set is returned by adding all unsaturated vertices to D and complete the pairing by exposing the remaining free points. In practice it seems that the last two steps have little influence on the behaviour of DominatingSet2.

To continue our discussion of DominatingSet2, we need a little more terminology. Although each operation is different, some operations are sufficiently similar that we say they have the same *type.* In fact, the type of an operation depends solely on the degree pair of the vertex chosen in that operations first step. For now we take this degree pair to be the type of the operation; later we will use integers for types. So when considering an algorithm as a sequence of operations, we classify the operations according to their type.

Algorithm 1 DominatingSet2

```
Set $P := \emptyset$;
Set $D := \emptyset$;
Choose $u \in V_{(0,0)}$ uniformly at random;
$D := D \cup \{u\}$;
Saturate($u$);
while $Y_{(1,0)} + Y_{(2,0)} + Y_{(2,1)} \neq 0$ do
  if $Y_{(2,1)} \neq 0$ then
    Choose $u \in V_{(2,1)}$ uniformly at random;
  else if $Y_{(2,0)} \neq 0$ then
    Choose $u \in V_{(2,0)}$ uniformly at random;
  else
    Choose $u \in V_{(1,0)}$ uniformly at random;
  end if
  $D := D \cup \{u\}$;
  Saturate($u$);
end while
Add all unsaturated vertices to $D$;
Expose all remaining free points;
return $D$, $P$;
```

Algorithm 2 Saturate

```
# When we expose points we add the corresponding pairs to $P$
Expose the free points associated with $u$;
Expose the free points associated with the out-neighbours of $u$;
Add accidental saturates to $D$;
```

At step t of DominatingSet2, the type of the next operation depends on the degree pairs of the vertices in the digraph G_t. From the definition of an operation, an operation of type (p, q) can only take place when there is a vertex of degree pair (p, q) in G_t. So when $Y_{(p,q)} > 0$ we say that an operation of type (p, q) is *permissible.* We next explain the most important aspect of DominatingSet2 and the reason that DominatingSet2 is known as a prioritised algorithm.

3.2.2 Prioritised algorithms

Consider how DominatingSet2 determines the type of the next operation. Notice that an operation of type $(2, 1)$ is performed whenever an operation of type $(2, 1)$ is permissible. If an operation of type $(2, 1)$ is not permissible but an operation of type $(2, 0)$ is, then the type of the next operation is $(2, 0)$. If the only permissible operation type is $(1, 0)$, then an operation of type $(1, 0)$ is performed. If there are no operation types permissible then the algorithm has finished. So DominatingSet2 prioritises operations of type $(2, 1)$ over operations of type $(2, 0)$ and $(1, 0)$, and prioritises operations of type $(2, 0)$ over operations of type $(1, 0)$. (Note that although the first operation of DominatingSet2 has type $(0, 0)$, we do not consider the type $(0, 0)$ to have a priority as there is only one type $(0, 0)$ operation; however, in general the type $(0, 0)$ does have a priority). This is the sense in which DominatingSet2 is a prioritised algorithm. All the algorithms we consider are prioritised in this way.

Finally, before describing how algorithms such as DominatingSet2 are analysed, we consider the algorithm DominatingSet2Det (given in Algorithms 3 and 4). DominatingSet2Det is a modification of DominatingSet2 which determines a dominating set for a 2-in 2-out digraph, given as input. Note that in Algorithm 3 we have $V_{(i,j)} = V_{(i,j)}(G)$ and $Y_{(i,j)} = Y_{(i,j)}(G)$ where G is the current state of the input graph. The analysis of DominatingSet2 [42] can be viewed as an average case analysis of DominatingSet2Det. Average case analysis has been the motivation of some of the early work [46] on analysing graph algorithms with differential equations. We leave it to the reader to notice the parallels between the two algorithms.

Algorithm 3 DominatingSet2Det

```
Require: $G$ is a 2-in 2-out digraph
  Set $D := \emptyset$;
  Choose $u \in V_{(2,2)}$ uniformly at random;
  $D := D \cup \{u\}$;
  Isolate($u$);
  while $Y_{(1,2)} + Y_{(0,2)} + Y_{(0,1)} \neq 0$ do
    if $Y_{(0,1)} \neq 0$ then
      Choose $u \in V_{(0,1)}$ uniformly at random;
    else if $Y_{(0,2)} \neq 0$ then
      Choose $u \in V_{(0,2)}$ uniformly at random;
    else
      Choose $u \in V_{(1,2)}$ uniformly at random;
    end if
    $D := D \cup \{u\}$;
    Isolate($u$);
  end while
  Add any non-isolated vertices to $D$;
  return $D$;
```

Algorithm 4 Isolate

```
  Remove the edges incident with the out-neighbours of $u$ from $G$;
  Remove the edges incident with $u$ from $G$;
  Add accidental isolates to $D$;
  # Accidental isolates are vertices, other than $u$ and its (former) out-neighbours,
  # that have all their remaining edges removed by the first two steps.
```

3.2.3 Analysing prioritised algorithms

Wormald [60] gives a detailed introduction to analysing prioritised algorithms on random regular graphs. Algorithms for random d-in d-out digraphs are analysed using the random variables $Y_{(i,j)}$ $(0 \le i, j \le 2)$, as well as any other random variables Z_i $(1 \le i \le \ell)$ of interest; for example, when analysing DominatingSet2 we also consider the random variable $Z(t) = |D(t)|$. Under certain conditions, the behaviour of the random variables 'settles down' as the number of vertices n tends to infinity. Of course, as n increases the ranges of the random variables increase (since the random variables count vertices) and the length of the process increases (since there are more edges to expose). So we scale the random variables in time and value by defining

$$y_{(i,j)}(t/n) = Y_{(i,j)}(t)/n \quad \text{and} \quad z_i(t/n) = Z_i(t)/n.$$

Then under certain conditions the scaled random variables approach deterministic functions as n tends to infinity.

These deterministic functions are the solutions to a system of ordinary differential equations. In particular, we group the sequence of operations defined by a prioritised algorithm into contiguous subsequences called clutches (the details are given below). Then we treat the expected changes in the random variables $Y_{(i,j)}$ and Z_i due to a clutch as a derivative. Writing these expected changes in terms of the scaled random variables suggests functions which are used to define a system of differential equations. Under certain conditions, the scaled random variables approach the solution to this system of differential equations as n tends to infinity.

The expected change in the random variables depends on the type of the operations performed in a clutch. Figure 3.2.1 shows the sequence of operation types from a run of an algorithm similar to DominatingSet2. Notice that the behaviour of the algorithm falls into three distinct phases and that in each phase, the set of types of operations performed is different. Thus each phase is analysed separately. Also during a phase, most operations are of a single type (as indicated by the horizontal lines). So we categorise phases by types as we do operations, with the type of a phase being the type of lowest priority amongst all the types of operations performed. The expected changes of the random variables, and the differential equations they suggest, depend on the phase of the algorithm.

Prioritised algorithms cannot be analysed operation by operation. The expected change in a random variable due to an operation depends on the type of the operation, but this type is not determined by the scaled random variables. So we group the sequence of operations defined by an algorithm in the following way. A *clutch* during a phase of type τ is a subsequence $\mathrm{op}_t, \mathrm{op}_{t+1}, \dots, \mathrm{op}_{t+k}$ of the operations performed by the algorithm, starting with an operation of type τ and ending at the last operation to have a type with higher priority than that of type τ (or the last operation of the algorithm). So op_t is an operation of type τ, the operations $\mathrm{op}_{t+1}, \dots, \mathrm{op}_{t+k}$ have types of higher priority than operations of type τ, and op_{t+k+1} is an operation of a type with priority not higher than the priority of type τ. In Figure 3.2.1, the vertical lines indicate clutches of length greater than one. Clutches of the same phase have very similar structure (although their lengths vary) and the scaled random variables do determine the expected change in the random variables due to a clutch. Therefore we are able to analyse prioritised algorithms clutch by clutch.

Unfortunately, analysing prioritised algorithms in this way is very difficult. There are two main problems. First we need to know that a clutch is not too long. If clutches are long then the change in the random variables will be large and the (scaled) random variables will be hard to approximate. Previously clutches have been shown to be sufficiently short using large deviation arguments [55, 60]. But such arguments are hard to apply in general. The other main problem arises because each phase is analysed separately. To obtain an analysis of the entire algorithm, the analyses of each phase need to be combined; so each change of phase must be detected and proved. In response to both of these problems, Wormald [61] introduced a new class of algorithms, called deprioritised algorithms.

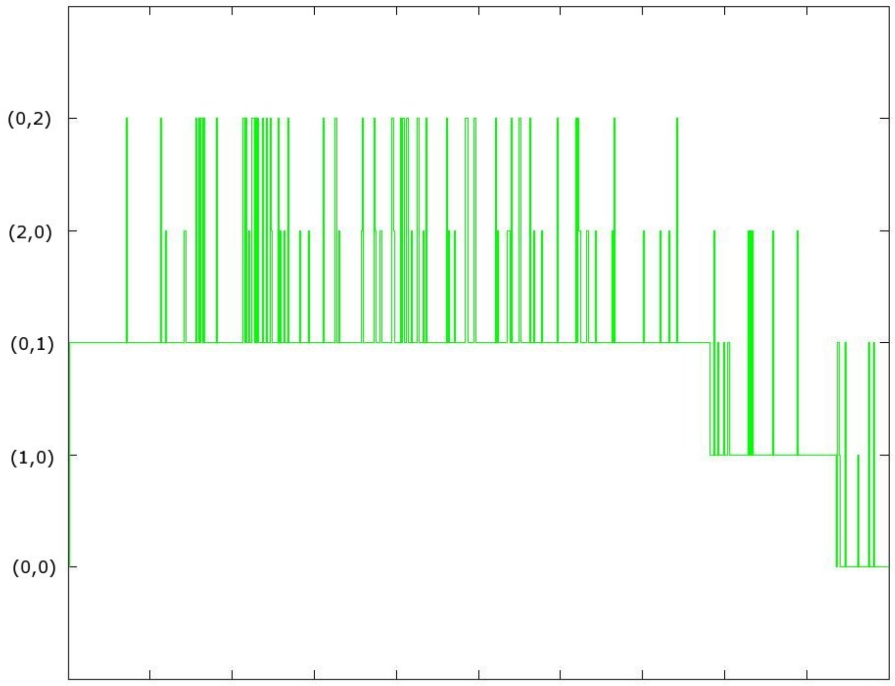


Figure 3.2.1: The sequence of operation types from a run of a prioritised algorithm. The types appear on the vertical axis in order of ascending priority.

3.2.4 Deprioritised algorithms

Deprioritised algorithms are a new class of algorithms which are easier to analyse than prioritised algorithms. Like prioritised algorithms, deprioritised algorithms proceed via a sequence of operations; but instead of selecting the type of the next operation according to a priority list, the type is selected according to a probability distribution. We call this distribution the *type distribution.* Note that the type distribution changes over the course of the algorithm. Thus we deprioritise the selection of the operation types.

In order to define a deprioritised algorithm, we need to define its type distribution. Because prioritised algorithms perform well, we would like to choose a type distribution so that the deprioritised algorithm approximates a given prioritised algorithm (in this case, both algorithms must use the same operations). In particular, we would like the probability that the next operation in the deprioritised algorithm has type r to approximate the proportion of type r operations at a similar stage of the prioritised algorithm (by a similar stage, we mean when the values of the random variables are similar). Unfortunately, it is not possible to be precise about the proportion of type r operations in the prioritised algorithm. However, we still define a type distribution with this goal in mind.

By selecting the type of the next operation using a probability distribution we solve the problems of prioritised algorithms mentioned above. We may now analyse the algorithm operation by operation: since operations (usually) have only a small affect on the random variables, it is easy to approximate the (scaled) random variables operation by operation. As a deprioritised algorithm is based on a given prioritised algorithm, the deprioritised algorithm also has phases. We again classify phases with types. But now the definition of the phases becomes part of the definition of the deprioritised algorithm. So analysing deprioritised algorithms is significantly easier than analysing prioritised algorithms.

However, deprioritised algorithms do have one extra complication. For a deprioritised algorithm to proceed, it must be able to perform an operation of the type selected. So if an operation of type r has a non-zero probability in the type distribution at time t, then the digraph G_t must have at least one vertex of degree pair r (recall that, for the moment, types are degree pairs). To ensure that there are always sufficient vertices with the required degree pairs, we start each phase of the deprioritised algorithm with a *preprocessing subphase.* In the preprocessing subphase only operations of type $(0, 0)$ are performed; this allows vertices of other degree pairs to build up. Thus we may perform operations of any type. As long as the preprocessing subphase is sufficiently short, its impact on the behaviour of the algorithm is negligible. Preprocessing subphases are a necessary part of the definition of any deprioritised algorithm.

The design of a deprioritised algorithm is based on a given prioritised algorithm. In particular, both algorithms use the same operations. But the algorithms choose the types of operations performed differently, so the algorithms define distinct random digraph processes. However, the algorithms can be analysed using the same technique. Moreover, the deprioritised algorithm is analysed using the same random variables that are defined for the prioritised algorithm. We will only analyse deprioritised algorithms, since analysing prioritised algorithms is much more difficult.

So when we use randomised algorithms to investigate random d-in d-out digraphs, we often take the following approach. We start by designing a prioritised algorithm. Next a deprioritised algorithm is defined so as to approximate the prioritised algorithm. Finally we analyse the deprioritised algorithm. The last two steps we call the *deprioritised approach* to analysing prioritised algorithms. Wormald [61] has previously shown how to apply the deprioritised approach to a general class of prioritised algorithms. However, Wormald's theory does not apply to every algorithm of interest, including algorithms for random d-in d-out digraphs. The next two chapters describe how the deprioritised approach may be applied to a class of prioritised algorithms defined more generally than the class considered by Wormald. Chapter 6 describes the analysis of two algorithms which requires the new theory presented in this thesis.

Chapter 4

Prioritised and Deprioritised Algorithms

Defining a deprioritised algorithm based on a given prioritised algorithm, and then analysing the deprioritised algorithm, is called the *deprioritised approach* and is now a standard technique. The deprioritised approach has been used with many prioritised algorithms [11, 31] including DominatingSet2 [42], the algorithm presented in Chapter 3. This chapter, and the next, describe how the deprioritised approach may be applied to any of a large class of prioritised algorithms. In particular, in this chapter we define the deprioritised algorithms based on certain prioritised algorithms.

The chapter begins with a description, in Section 4.1, of a class of prioritised algorithms called *deprioritisable algorithms*. The description is given in a general context, so the theory presented is not restricted to algorithms for random d-in d-out digraphs. In Section 4.2 we determine functions which we believe approximate the expected proportions of operation types for a deprioritisable algorithm. Then in Section 4.3 we define functions which, under conditions given in the next chapter, describe the behaviour of the deprioritised algorithm. We end the chapter by giving an explicit definition of the deprioritised algorithm based on a given deprioritisable algorithm.

4.1 Deprioritisable Algorithms

The theory presented in this chapter, and the next, applies to a class of prioritised algorithms we call *deprioritisable algorithms.* We define deprioritisable algorithms in an abstract setting and, using this definition, we define a deprioritised algorithm based on a given deprioritisable algorithm.

A deprioritisable algorithm is a randomised algorithm $\mathcal{P}$ defined on sets Ω_n for $n \geq 1$. For each n, we assume that for any element $G_0(n) \in \Omega_n$ the algorithm $\mathcal{P}$, given $G_0(n)$ as input, generates a random process $\{G_t\}_{t=0}^{F(n)}$ on Ω_n. For example, Ω_n may be the set of all multi-digraphs on the vertex set $\{1, \ldots, n\}$ for which each vertex has in-degree at most d and out-degree at most d, and $G_0(n)$ the empty digraph. The properties $\mathcal{P}$ must satisfy to be a deprioritisable algorithm are given in the following definition.

Definition 4.1.1 (Deprioritisable Algorithms). A randomised algorithm $\mathcal{P}$ defined on sets Ω_n (for $n \geq 1$) is *deprioritisable* if the following properties hold.

(i) The random process $\{G_t\}_{t=0}^{F}$ on Ω_n is Markovian.

(ii) The algorithm proceeds via a sequence of operations $\mathrm{op}_1, \ldots, \mathrm{op}_F$ such that G_t is obtained from G_{t-1} via op_t for $t = 1, \ldots, F$. Moreover, the operations are classified into types. That is, for some fixed integer k, operation op_t has type $\tau(\mathrm{op}_t) \in \{0, 1, \ldots, k\}$, for $t = 1, \ldots, F$.

(iii) There exists random variables $Y_0, \ldots, Y_m$ defined on Ω_n (for all n) such that, for $i = 0, \ldots, m$ and $r = 0, \ldots, k$, we have

$$\begin{aligned}\mathbb{E}(Y_i(G_{t+1}) - Y_i(G_t) \,|\, G_t,\ \tau(\mathrm{op}_t) = r)& \\ = f_{i,r}(t/n, Y_0(G_t)/n, \ldots, Y_m(G_t)/n) + o(1)&\end{aligned} \tag{4.1.1}$$

for some functions $f_{i,r}$. Moreover, for some polynomial H, each function $f_{i,r}$ has the form

$$f_{i,r}(x, y_0, \ldots, y_m) = \frac{g_{i,r}(x, y_0, \ldots, y_m)}{(H(x, y_0, \ldots, y_m))^{\ell_i}} \tag{4.1.2}$$

for some integer $\ell_i \geq 1$ and some polynomial $g_{i,r}$.

(iv) For $r = 0, \dots, k$, an operation of type r can be performed on every $G \in \Omega_n$ for which $Y_r(G) > 0$. When $Y_r(G) > 0$ we say that an operation of type r is *permissible* (on G). Then, given G_t, the type $\tau(\mathrm{op}_{t+1})$ of the next operation is the maximum of all types which are permissible on G_t.

(v) For some fixed β we have

$$\max_{0 \le i \le m} |Y_i(G_{t+1}) - Y_i(G_t)| \le \beta$$

for all $t \ge 0$.

(vi) For some fixed M we have $|Y_i(G_t)| \le Mn$ for $i = 0, \dots, m$ and all $t \ge 0$.

(vii) The limits $\lim_{n \to \infty} Y_i(0)/n$ exist for $i = 0, \dots, m$.

Properties (i)–(iv), excluding property (4.1.2) of the form of the functions $f_{i,r}$, give an adequate definition of a prioritised algorithm; though we do not usually give such a precise definition. Much of what follows in this chapter applies to prioritised algorithms, not just deprioritisable algorithms; so we will only require a prioritised algorithm be deprioritisable when needed. Properties (4.1.2) and (v)–(vii) are necessary for analysing the deprioritised algorithm based on a deprioritisable algorithm. Note that the restriction on the functions $f_{i,r}$ given by (4.1.2) is reasonable since we are interested in combinatorial algorithms. Many prioritised algorithms, including the algorithms we are interested in, are also deprioritisable algorithms.

Since we are interested in the behaviour of deprioritisable algorithms as n tends to infinity, we often drop n from the notation. Recall that we consider two random processes on Ω_n: one defined by a deprioritisable algorithm and one defined by a deprioritised algorithm. The random variables Y_i are defined on both random processes. So we let $Y_i(t) = Y_i(G_t)$ for the random process $\{G_t\}$ when the random process being considered is clear from the context. The random process will depend on which algorithm (deprioritisable or deprioritised) we are considering.

Notice that the functions $f_{i,r}$ depend only on the random variables. So we let

$$f_{i,r}(t) = f_{i,r}(t/n, Y_0(t)/n, \dots, Y_m(t)/n)$$

when the underlying random process is clear from the context. Since the random variables are defined on both random processes, the functions $f_{i,r}$ are defined for the deprioritised algorithm as well. Later we consider continuous functions corresponding to the scaled random variables. So we also let

$$f_{i,r}(x) = f_{i,r}(x, y_0(x), \dots, y_m(x))$$

when the functions y_i are clear from the context.

At this point, it is useful to introduce the domain

$$\begin{aligned} D(\delta) = \{(x, y_0, \dots, y_m) \,:\, -\delta < x < C,\; y_0 > \delta,\; H > \delta,& \\ |y_i| < 2M \quad (i = 0, \dots, m)\}& \end{aligned}$$

for $\delta > 0$. Later, in Section 4.3, we use $D(\delta)$ to define functions which approximate the scaled random variables of the deprioritised algorithm. Notice that the condition $H > \delta$ ensures that the functions $f_{i,r}$ are defined on $D(\delta)$. We will discuss the domain $D(\delta)$ further in Section 4.3.

In an earlier analysis of deprioritised algorithms by Wormald [61, Theorem 2], the form of the functions $f_{i,r}$ is also used. In particular, Wormald requires that these functions satisfy

$$C_1 y_{i+1} - C_2 y_i \leq f_{i,r} \leq C_3 y_{i+1} \tag{4.1.3}$$

for positive constants C_1, C_2, and C_3 on an appropriate domain. However, the functions $f_{i,r}$ for the algorithms that we consider have the form

$$f_{i,r} = \delta_{i,r} - p y_i + q_1 y_{j_1} + q_2 y_{j_2}$$

where p, q_1, and q_2 are polynomials. Moreover, at least one term in q_1 does not contain the variable y_{j_2}. Thus we cannot bound $f_{i,r}$ above using just y_{j_2}. Indeed, there is no variable y_j and constant C for which $f_{i,r} \leq C y_j$. (In Section 4.2.4, we discuss this topic further.) Hence we cannot apply the earlier work of Wormald. So the theory presented in this thesis is developed in order to apply the deprioritised approach to a larger class of prioritised algorithms, including algorithms on random d-in d-out digraphs (where d is fixed).

We now review how a prioritised algorithm is analysed, as this affects the design of the corresponding deprioritised algorithms. Prioritised algorithms are not analysed operation by operation. Instead they are analysed using contiguous subsequences of operations called *clutches*. For $r = 0, \dots, k$, an (r, k)-*clutch* is a contiguous subsequence of the operations of a prioritised algorithm consisting of an operation of type r, followed by all operations of types greater than r, up to (but not including) the next operation of type r or less. Note that an (r, k)-clutch may contain (r', k)-clutches for $r' > r$. In fact, an (r, k)-clutch is an operation of type r, followed by zero or more (k, k)-clutches, followed by zero or more $(k - 1, k)$-clutches, and so on, until ending with zero or more $(r + 1, k)$-clutches. We use clutches to define the deprioritised algorithm based on a given prioritised algorithm.

Prioritised algorithms go through long periods in which the operations can be grouped into a sequence of short (r, k)-clutches (for some fixed r). We call such periods of the algorithm a *phase* of type r. The phases of prioritised algorithms are not defined precisely as it is difficult to do so and is not required to define the corresponding deprioritised algorithm. As we would like deprioritised algorithms to approximate prioritised algorithms, deprioritised algorithms also have phases. Again the phases of deprioritised algorithms are classified by types.

For the deprioritised approach to be useful, the correctness of the prioritised algorithm must not rely on the prioritisation of the operation types. Most prioritised algorithms are designed to maintain an invariant between operations. For example, DominatingSet2 maintains a set of vertices D for which, between each operation, every saturated vertex is either in D or is dominated by some vertex in D. Thus at the end of DominatingSet2, the set D is a dominating set (as every vertex is saturated). So the prioritisation of DominatingSet2 is not required for D to be a dominating set. All the deprioritisable algorithms we consider maintain such invariants and do not require the prioritisation to achieve their goal. For such algorithms the deprioritised approach is useful.

4.2 Type Distributions

Deprioritised algorithms proceed via a sequence of phases. The phases of deprioritised algorithms begin with a *preprocessing subphase*, in which only operations of type 0 are performed. The preprocessing subphase is followed by the *main subphase* of a phase. Preprocessing subphases are much shorter than main subphases: for each phase, the proportion of time spent in the preprocessing subphase is only $o(1)$. The main subphases of a deprioritised algorithm are based on phases of a prioritised algorithm while the preprocessing subphases allow the main phases to occur.

During a main subphase, the type of the next operation to be performed is chosen according to a probability distribution called the *type distribution.* The type distribution is calculated before each operation of the deprioritised algorithm. During a phase of type τ, we would like the type distribution to approximate the expected proportion of operation types during a (τ, k)-clutch at a similar stage of the prioritised algorithm. By 'a similar stage' we mean when the random variables of both algorithms have similar values. Unfortunately we are not able to determine the expected proportions of operation types in a clutch rigorously. However, we can determine functions that we believe approximate these proportions. These functions, given in the next section, are sufficient to define the deprioritised algorithm.

4.2.1 Proportions of operation types

A prioritised algorithm, when in a phase of type τ, proceeds via a sequence of (τ, k)-clutches. The phase continues while the clutches are sufficiently short; if the clutches become too long then the algorithm enters a new phase. When clutches are short, the expected proportions of operation types near step t should be close to the expected proportions of operation types in a clutch occurring near step t. So we determine functions which we believe approximate the expected proportions of operation types using clutches.

So we would like to know when a (τ, k)-clutch will be 'short'. Consider a (τ, k)-clutch. If the random variable Y_r, for some $r \in \{\tau+1, \dots, k\}$, increases quickly, then many operations of type r are performed and the clutch is long. So consider the expected change in Y_r due to a (r, k)-clutch for $r = \tau+1, \dots, k$. Note that if an (r, k)-clutch occurs, then Y_r is positive at the start of the clutch. If for all r, the expected change in Y_r due to a (r, k)-clutch is negative, that is, Y_r has an expected decrease between type r operations, then we expect a (τ, k)-clutch will be short.

Previously we defined clutches that consisted of operations of all types in $\{\tau, \dots, k\}$ for some τ. However, during a phase of type τ of a prioritised algorithm, it may be that for some types greater than τ, a.a.s. no operations of those types are performed. So we now generalise the definition of a clutch. Let $\mathcal{O} = \{w_1, \dots, w_a\}$ be a subset of $\{0, \dots, k\}$ such that $w_1 < w_2 < \dots < w_a$. Then an $\mathcal{O}$-clutch is a sequence of operations consisting of an initial operation of type w_1, followed by each subsequent operation of a type from $\mathcal{O}\backslash\{w_1\}$, up to (but not including) the next operation of a type not in $\mathcal{O}\backslash\{w_1\}$. Note that we can also define $\mathcal{O}$-clutches recursively. Define $\mathcal{O}_i = \mathcal{O}\backslash\{w_1, \dots, w_{i-1}\} = \{w_i, \dots, w_a\}$. Then an $\mathcal{O}$-clutch is an operation of type w_1, followed by zero or more $\mathcal{O}_a$-clutches, followed by zero or more $\mathcal{O}_{a-1}$-clutches, and so on, until ending with zero or more $\mathcal{O}_2$-clutches. This recursive definition is used to determine functions which we believe approximate the expected proportion of operation types in a prioritised algorithm.

Assume that the operations occurring near step t of the prioritised algorithm algorithm can be grouped into a sequence of $\mathcal{O}$-clutches. Let $\mathcal{O} = \{w_1, \dots, w_a\}$ with $w_1 < \dots < w_a$. In Definition 4.2.1, we define functions $p_r(t)$ (for $r = 0, \dots, k$) which we believe approximate the proportion of type r operations occurring near step t. We also define functions $\Phi_b(t)$ (for $b = 2, \dots, a$) which we believe approximate the expected change in Y_{w_b} due to an $\mathcal{O}_b$-clutch occurring near step t. As we do not prove these interpretations of p_r and Φ_b, we do not make the interpretations rigorous.

Now fix a particular $\mathcal{O}$-clutch Γ occurring near step t. Let n_r be the expected number of type r operations in Γ. For $j \geq 1$, the random variable Y_{w_j} is zero at the beginning and end of Γ. So the expected change in Y_{w_j} due to Γ is

$$\sum_{i=1}^{a} n_{w_i} f_{w_j,w_i} = 0.$$

We also have $n_{w_1} = 1$. Note that these equations have previously been considered by Shi and Wormald [55]. Setting $p_r = n_r / \sum_{r=0}^{k} n_r$ we obtain the equations

$$\sum_{i=1}^{a} p_{w_i} f_{w_j,w_i} = 0 \quad (\text{for } j = 2, \ldots, a) \qquad \text{and} \qquad \sum_{i=1}^{a} p_{w_i} = 1.$$

We can write the above equations as the matrix equation

$$(p_{w_1}, \ldots, p_{w_a})\mathcal{C} = (1, 0, \ldots, 0),$$

where $\mathcal{C}$ is called the *clutch matrix* and is defined in Definition 4.2.1 below. Applying Cramer's rule we find an explicit formula for p_{w_i}. The functions Φ_b are also defined using the clutch matrix, and both p_r and Φ_b are given in Definition 4.2.1. Note that solving the matrix equation numerically is a more efficient way of determining the values of p_{w_i} than using the explicit formula in Definition 4.2.1. However the explicit formula is useful for determining properties of the functions p_r and functions defined in terms of p_r. We determine such properties later in this chapter and in Chapter 5.

The functions p_r and Φ_b are defined using determinants of submatrices of the clutch matrix. For submatrices of an arbitrary matrix we use the following notation: for an $m \times m$ matrix A and sets $R, C \subseteq \{1, \ldots, m\}$ we denote by $A(R, C)$ the matrix obtained from A by deleting the rows and columns with indices in R and C respectively. If $R = \{r\}$ and $C = \{c\}$, then we write $A(R, C)$ as $A(r, c)$. It is also useful to define the determinant of the matrix with zero rows and zero columns to be 1. The following definition gives functions we believe approximate the expected proportion of operation types in a clutch.

Definition 4.2.1 (Proportions of Operation Types). Let $\mathcal{O} = \{w_1, \ldots, w_a\}$ with $w_1 < \cdots < w_a$. The *clutch matrix* for an $\mathcal{O}$-clutch at step t is

$$\mathcal{C}(t) = \begin{pmatrix} 1 & f_{w_2,w_1}(t) & \ldots & f_{w_a,w_1}(t) \\ \vdots & \vdots & & \vdots \\ 1 & f_{w_2,w_a}(t) & \ldots & f_{w_a,w_a}(t) \end{pmatrix}.$$

Then, for $b = 2, \ldots, |\mathcal{O}|$, we define

$$\Phi_b(t) = \frac{\det \mathcal{C}(t)(\{1,\ldots,b-1\},\{1,\ldots,b-1\})}{\det \mathcal{C}(t)(\{1,\ldots,b\},\{1,\ldots,b\})}, \tag{4.2.1}$$

and, for $r = 0, \ldots, k$, we define

$$p_r(t) = \begin{cases} 0 & \text{if } r \notin \mathcal{O}, \\ (-1)^{b+1}\frac{\det \mathcal{C}(t)(b,1)}{\det \mathcal{C}(t)} & \text{if } r = w_b \in \mathcal{O}. \end{cases} \tag{4.2.2}$$

Notice that the clutch matrix and the functions p_r and Φ_b depend only on $\mathcal{O}$ and the random variables Y_i (via the functions $f_{i,r}$). So the above definition applies for both a prioritised algorithm and the corresponding deprioritised algorithm. Later in this section we define the type distribution of a deprioritised algorithms using Definition 4.2.1. But first we give another heuristic justification for the definition of p_r which also addresses the definition of the functions Φ_b.

Again we consider a $\mathcal{O}$-clutch Γ which occurs near step t of the prioritised algorithm. Note that for this justification we are considering the random process defined by the prioritised algorithm. Recall that $\Phi_b(t)$ is analogous to the expected change in Y_{w_b} due to a $\mathcal{O}_b$-clutch occurring in Γ. When $\Phi_b(t) < 0$ for $b = 2, \ldots, a$, we expect Γ to be short. Indeed Γ should be sufficiently short that the random variables only change by $o(n)$ during Γ. So the functions $f_{i,r}$, p_r, and Φ_b should only change by $o(1)$. Such a result has been proved in some special cases [60, Section 4] [55, Section 5] (and is trivially true for $|\mathcal{O}| = 1$), however we are unable to prove this claim for arbitrary clutches. If we could prove the above claim, then we should be able to provide a rigorous interpretation of the functions p_r and Φ_b.

The justification proceeds by induction on $|\mathcal{O}|$ and the recursive definition of clutches given above. Throughout the justification we ignore any $o(1)$ terms and drop t from the notation. We break Γ into a *head*, consisting of all the operations occurring before the first operation of type w_2, and a *tail*, consisting of all the operations not in the head. During the head no operations of type w_2 are performed. So we view the head as a $\mathcal{O}\backslash\{w_2\}$-clutch. The tail is a sequence of $\mathcal{O}_2$-clutches. The clutch matrix corresponding to the head is $\mathcal{D} = \mathcal{C}(2,2)$, while the clutch matrix for an $\mathcal{O}_2$-clutch is $\mathcal{E} = \mathcal{C}(1,2)$.

Notice that

$$\frac{\det \mathcal{D}(\{1,\ldots,s-1\},\{1,\ldots,s-1\})}{\det \mathcal{D}(\{1,\ldots,s\},\{1,\ldots,s\})} = \frac{\det \mathcal{E}(\{1,\ldots,s-1\},\{1,\ldots,s-1\})}{\det \mathcal{E}(\{1,\ldots,s\},\{1,\ldots,s\})}$$
$$= \Phi_{s+1} < 0$$

for $s = 2,\ldots,a-1$. Thus the expected proportion of type w_b operations in the head is h_b where

$$h_1 = \frac{\det \mathcal{C}(\{1,2\},\{1,2\})}{\det \mathcal{C}(2,2)},\ h_2 = 0, \quad \text{and} \quad h_b = (-1)^b \frac{\det \mathcal{C}(\{2,b\},\{1,2\})}{\det \mathcal{C}(2,2)}$$

for $b = 3,\ldots,a$. While the expected proportion of type w_b operations in an $\mathcal{O}_2$-clutch of the tail is t_b where

$$t_1 = 0 \ \text{ and } \ t_b = (-1)^b \frac{\det \mathcal{C}(\{1,b\},\{1,2\})}{\det \mathcal{C}(1,2)}$$

for $b = 2,\ldots,a$.

Next we determine the expected number of $\mathcal{O}_2$-clutches in the tail of Γ. Let β be the expected change in Y_{w_2} due to an $\mathcal{O}_2$-clutch. Since there is exactly one operation of type w_2 in an $\mathcal{O}_2$-clutch, the expected length of an $\mathcal{O}_2$-clutch is $1/t_2$. Thus

$$\beta = f_{w_2,w_2} + \frac{1}{t_2}\left(t_3 f_{w_2,w_3} + \cdots + t_a f_{w_2,w_a}\right)$$
$$= \det \mathcal{C}(1,1)/\det \mathcal{C}(\{1,2\},\{1,2\}) = \Phi_2.$$

Similarly to the above, the expected change in Y_{w_2} due to the head is α where

$$\alpha = f_{w_2,w_1} + \frac{1}{h_1}\left(h_3 f_{w_2,w_3} + \cdots + h_a f_{w_2,w_a}\right)$$
$$= \det \mathcal{C}(2,1)/\det \mathcal{C}(\{1,2\},\{1,2\}).$$

Note that Γ is a head followed by a sequence of $\mathcal{O}_2$-clutches. So Γ is similar to a clutch consisting of two operations, but instead of operations there are clutches. Consider an $\{a,b\}$-clutch where $a < b$. The expected number of type b operations is $-f_{b,a}/f_{b,b}$ provided $f_{b,b} < 0$. Similarly, the expected number of $\mathcal{O}_2$-clutches in Γ is $-\alpha/\beta$ provided $\beta < 0$. As $\beta = \Phi_2 < 0$, the expected proportion of type w_b operations in Γ is

$$p_{w_b} = \left(\frac{h_b}{h_1} - \frac{\alpha t_b}{\beta t_2}\right)\left(\frac{1}{h_1} - \frac{\alpha}{\beta t_2}\right)^{-1} \qquad (4.2.3)$$

for $b = 1,\ldots,a$. Thus we have a equalities to prove.

Consider p_{w_1}: we want to show that

$$\left(\frac{1}{h_1} - \frac{\alpha}{\beta t_2}\right)^{-1} = \frac{\det \mathcal{C}(1,1)}{\det \mathcal{C}}. \tag{4.2.4}$$

Expanding and simplifying, we see that (4.2.4) is equivalent to

$$\det \mathcal{C}(2,2) \det \mathcal{C}(1,1) - \det \mathcal{C}(1,2) \det \mathcal{C}(2,1) = \det \mathcal{C} \det \mathcal{C}(\{1,2\},\{1,2\}). \tag{4.2.5}$$

Equation (4.2.5) holds by a theorem of Jacobi [3, Section 2.4], and indeed equation (4.2.5) still holds when $\mathcal{C}$ is replaced by an arbitrary matrix. From (4.2.4), equation (4.2.3) holds for $b = 2$ as well.

Now for $b = 3, \ldots, a$, we want to show that

$$\left(\frac{h_b}{h_1} - \frac{\alpha t_b}{\beta t_2}\right)\left(\frac{1}{h_1} - \frac{\alpha}{\beta t_2}\right)^{-1} = (-1)^{b+1} \frac{\det \mathcal{C}(b,1)}{\det \mathcal{C}}.$$

Expanding and simplifying, this equation is equivalent to

$$\begin{aligned} &\det \mathcal{C}(\{1,b\},\{1,2\}) \det \mathcal{C}(2,1) \\ &\qquad - \det \mathcal{C}(\{2,b\},\{1,2\}) \det \mathcal{C}(1,1) = \det \mathcal{C}(b,1) \det \mathcal{C}(\{1,2\},\{1,2\}). \end{aligned} \tag{4.2.6}$$

Each matrix in (4.2.6) is obtained from $\mathcal{C}$ by deleting at least the first column. So let Q be the matrix obtained by replacing the first column of $\mathcal{C}$ by the b-th standard basis vector. Then (4.2.6) is equivalent to the corresponding equation for Q. Applying (4.2.5) with Q in place of $\mathcal{C}$ shows that (4.2.6) holds. So (4.2.3) is satisfied for $b = 3, \ldots, a$. This concludes our heuristic justification of Definition 4.2.1.

We now define the type distributions of the deprioritised algorithm based on a given deprioritisable algorithm. The type distribution for a phase of type τ is defined using Definition 4.2.1 and a set of types $\mathcal{M}^{(\tau)} \subseteq \{\tau, \ldots, k\}$ called an *irreducible type set*. We note that the irreducible type sets are determined by the deprioritisable algorithm. Irreducible type sets are defined later in the chapter (see Definition 4.2.5).

In terms of the deprioritisable algorithm, we believe that operations of types in $\{\tau, \ldots, k\} \backslash \mathcal{M}^{(\tau)}$ a.a.s. do not occur in a phase of type τ (but, we do not prove this).

We could also define the type distribution using Definition 4.2.1 with $\mathcal{O} = \{\tau, \ldots, k\}$. However, we prove later in Lemma 4.2.8 that such a definition would result in the same type distribution as that obtained using $\mathcal{O} = \mathcal{M}^{(\tau)}$. We prefer to define the type distribution using the irreducible type sets.

Definition 4.2.2 (Type Distributions). Let $\mathcal{P}$ be a deprioritisable algorithm and let $\mathcal{M}^{(\tau)}$ be the irreducible type set for a phase of type τ of $\mathcal{P}$. Then the type distribution for a phase of type τ of the deprioritised algorithm based on $\mathcal{P}$ is defined by the functions $p_r^{(\tau)}(t) = p_r(t)$ (for $r = 0, \ldots, k$) obtained via Definition 4.2.1 with $\mathcal{O} = \mathcal{M}^{(\tau)}$.

We also define $\mathcal{C}^{(\tau)}(t)$ to be the clutch matrix of an $\mathcal{M}^{(\tau)}$-clutch at step t and $\Phi_b^{(\tau)}(t) = \Phi_b(t)$ via Definition 4.2.1 with $\mathcal{O} = \mathcal{M}^{(\tau)}$.

Next we determine conditions under which the above definition of a type distribution should be used. These same conditions also determine when a deprioritised algorithm is in the main subphase of a phase of type τ. Recall that the definitions of the type distribution and phases are part of the definition of the deprioritised algorithm.

The functions $p_r^{(\tau)}(t)$ are used to define the type distribution of a deprioritised algorithm at step t during a phase of type τ. (The underlying random process is now the process defined by the deprioritised algorithm.) Thus the functions $p_r^{(\tau)}(t)$ must satisfy $p_r^{(\tau)}(t) \geq 0$ (for $r = 0, \ldots, k$) and $\sum_{r=0}^{k} p_r^{(\tau)}(t) = 1$. From the form of the functions $p_r^{(\tau)}(t)$ (see (4.2.2)), we see that they do indeed sum to one. We ensure that each function is non-negative by restricting the definition of a phase of type τ. For technical reasons we also require that $p_r^{(\tau)}(t)$ be positive for $r \in \mathcal{M}^{(\tau)}$. Note that $p_r^{(\tau)}(t) = 0$ for $r \in \{\tau, \ldots, k\} \backslash \mathcal{M}^{(\tau)}$, which we prove later in Lemma 4.2.8. The functions $p_r^{(\tau)}(t)$ are used to define differential equations and so must also be Lipschitz. So we also require that $\left|\det \mathcal{C}^{(\tau)}(t)\right| > \delta$ for some constant $\delta > 0$. Thus for a deprioritised algorithm to be in a phase of type τ at step t, we require that:

(a) $\left|\det \mathcal{C}^{(\tau)}(t)\right| > \delta$ for some $\delta > 0$,

(b) $p_r^{(\tau)}(t) > 0$ for $r \in \mathcal{M}^{(\tau)}$, and

(c)
$$\Phi_b^{(\tau)}(t) = \frac{\det \mathcal{C}^{(\tau)}(t)(\{1,\ldots,b-1\},\{1,\ldots,b-1\})}{\det \mathcal{C}^{(\tau)}(t)(\{1,\ldots,b\},\{1,\ldots,b\})} < 0$$
for $b = 2,\ldots,\left|\mathcal{M}^{(\tau)}\right|$.

The next lemma provides equivalent conditions which are easier to use. We use this result when analysing each main subphase of a deprioritised algorithm.

Lemma 4.2.3. *Let $a = \left|\mathcal{M}^{(\tau)}\right|$. For all t, the conditions (a), (b), and (c) above are equivalent to the following:*

(i) $(-1)^{a+1} \det \mathcal{C}^{(\tau)}(t) > \delta$ *for some* $\delta > 0$,

(ii) $(-1)^{a+b} \det \mathcal{C}^{(\tau)}(t)(b,1) > 0$ *for* $b = 1,\ldots,a$, *and*

(iii) $(-1)^{a-b+1} \det \mathcal{C}^{(\tau)}(t)(\{1,\ldots,b-1\},\{1,\ldots,b-1\}) > 0$ *for* $b = 2,\ldots,a$.

Also, conditions (i) and (ii) imply (a) and (b).

Proof. Let $\mathcal{M}^{(\tau)} = \{w_1,\ldots,w_a\}$ where $w_1 < \ldots < w_a$. First we show that (a), (b), and (c) imply (i), (ii), and (iii). Note that

$$\mathcal{C}^{(\tau)}(t)(\{1,\ldots,a\},\{1,\ldots,a\})$$

is the empty matrix and by convention has determinant 1. Thus for $b = 2,\ldots,a$ we have

$$\det \mathcal{C}^{(\tau)}(t)(\{1,\ldots,b-1\},\{1,\ldots,b-1\}) = \Phi_b^{(\tau)}(t) \cdots \Phi_a^{(\tau)}(t).$$

By (c) the sign of $\Phi_b^{(\tau)}(t) \cdots \Phi_a^{(\tau)}(t)$ is $(-1)^{a-b+1}$. Therefore (c) implies (iii).

Now by (iii) with $b = 2$ we have

$$(-1)^{a+1} \det \mathcal{C}^{(\tau)}(t)(1,1) > 0.$$

So (ii) with $b = 1$ is satisfied. From (b) and (4.2.2) we have

$$p_{w_1}^{(\tau)}(t) = \frac{\det \mathcal{C}^{(\tau)}(t)(1,1)}{\det \mathcal{C}^{(\tau)}(t)} > 0,$$

and so

$$(-1)^{a+1} \det \mathcal{C}^{(\tau)}(t) > 0. \tag{4.2.7}$$

Together with (a), this implies that (i) is satisfied. Now from (4.2.7) and $p_r^{(\tau)}(t) > 0$ for $r \in \mathcal{M}^{(\tau)}$ condition (ii) holds for $b = 3, \ldots, a$. Thus (a),(b), and (c) imply (i),(ii), and (iii).

On the other hand, substituting (i) and (ii) into the equations $p_r^{(\tau)}(t)$, we obtain (a) and (b). Similarly (iii) implies (c). □

For a given type τ, we use (i), (ii), and (iii), plus some other conditions, to define a domain $V^{(\tau)}(\delta)$ which specifies the main subphase of a phase of type τ for a deprioritised algorithm. In particular, the main subphase of a phase of type τ lasts while the scaled random variables of the deprioritised algorithm remain in $V^{(\tau)}(\delta)$. When the scaled random variables exit $V^{(\tau)}(\delta)$, the algorithm either finishes or enters a new phase of a different type. Later in the chapter, we complete the definition of the deprioritised algorithm by defining $V^{(\tau)}(\delta)$ and the sequence of phase types of the algorithm.

4.2.2 Irreducible type sets

We now define the irreducible type sets $\mathcal{M}^{(\tau)}$. Consider a phase of type τ during a prioritised algorithm. This phase is a sequence of $\{\tau, \ldots, k\}$-clutches. Note that at the start of each clutch of the phase, the random variables $Y_{\tau+1}, \ldots, Y_k$ are zero. Now an operation of type r, for $r > \tau$, is performed only if $Y_r > 0$. So for an operation of type r to be performed during the phase, the random variable Y_r must increase. Thus we must have $f_{r,w} > 0$ for some type w such that an operation of type w is performed. This motivates the definitions and results that follow. Note that these definitions and results concern only the functions $f_{i,r}$, p_r, and Φ_b and not the random processes defined by the algorithms. Throughout this section, we refer to the domain $D(\delta)$ defined after Definition 4.1.1.

Definition 4.2.4 (Expected Change Digraph (ECD)). The *expected change digraph* (or *ECD*) for a phase of type τ is the digraph $\Gamma(\tau)$ on the vertices $\{\tau, \ldots, k\}$ for which there is an edge from w_1 to w_2 if and only if $w_1 \neq w_2$ and there exists $(x, y_0, \ldots, y_m) \in D(\delta)$ such that

$$y_{\tau+1} = \cdots = y_k = 0 \text{ and } f_{w_2,w_1}(x, y_0, \ldots, y_m) \neq 0.$$

To check the condition above, we can substitute $y_i = 0$ for $i = \tau + 1, \ldots, k$ in f_{w_2,w_1} and simplify the resulting expression. We will obtain either 0 or a polynomial in the variables $x, y_0, \ldots, y_\tau, y_{k+1}, \ldots, y_m$. So determining an ECD should not be too difficult, especially with the aid of a computer. Next we define the irreducible type sets in terms of expected change digraphs.

Definition 4.2.5 (Irreducible Type Set). The *irreducible type set* for a phase of type τ is the set $\mathcal{M}^{(\tau)}$ of vertices of $\Gamma(\tau)$ that can be reached from τ via directed paths.

During a phase of type τ of a prioritised algorithm, (we believe that) operations of types from $\{\tau, \ldots, k\} \backslash \mathcal{M}^{(\tau)}$ a.a.s. do not occur. Thus we have defined the deprioritised algorithm so that, during a phase of type τ, operations of types from $\{\tau, \ldots, k\} \backslash \mathcal{M}^{(\tau)}$ are not performed. The most important property of $\mathcal{M}^{(\tau)}$ is given by the next lemma. It is also useful to note that $\tau \in \mathcal{M}^{(\tau)}$ always.

Lemma 4.2.6. *Fix a type $r \in \{\tau, \ldots, k\} \backslash \mathcal{M}^{(\tau)}$. Then for all $(x, y_0, \ldots, y_m) \in D(\delta)$ with $y_{\tau+1} = \cdots = y_k = 0$, we have $f_{r,w}(x, y_0, \ldots, y_m) = 0$ for all $w \in \mathcal{M}^{(\tau)}$.*

Proof. Assume not. That is, assume for some $r \in \{\tau, \ldots, k\} \backslash \mathcal{M}^{(\tau)}$ there exists $(x, y_0, \ldots, y_m) \in D(\delta)$ with $y_{\tau+1} = \cdots = y_k = 0$ and $f_{r,w}(x, y_0, \ldots, y_m) \neq 0$ for some $w \in \mathcal{M}^{(\tau)}$. Then by definition, there is an edge from w to r in the expected change digraph. Since w lies on a directed path from τ, so too must r. Therefore $r \in \mathcal{M}^{(\tau)}$; a contradiction. □

Lemma 4.2.6 is useful because the functions $\hat{y}_i$ that describe the scaled random variables of the deprioritised algorithm (or so we wish to prove) are such that $\hat{y}_i = 0$ for $i = \tau + 1, \ldots, k$ during a phase of type τ. These functions are defined in the next section. We use Lemma 4.2.6 to show, for example, that the type distributions obtained via Definition 4.2.1 with $\mathcal{O} = \mathcal{M}^{(\tau)}$ and $\mathcal{O} = \{\tau, \ldots, k\}$ are the same. However, the functions $f_{i,r}$ often satisfy a stronger, and very useful, property than that implied by Lemma 4.2.6.

Definition 4.2.7 (Independent Types Property). The functions $f_{i,r}$ satisfy the *Independent Types Property* for a phase of type τ if, for $r \in \{\tau, \ldots, k\} \backslash \mathcal{M}^{(\tau)}$ and for all $(x, y_0, \ldots, y_m) \in D(\delta)$ with $y_{\tau+1} = \cdots = y_k = 0$, we have

$$f_{r,w}(x, y_0, \ldots, y_k) = -\delta_{r,w}$$

for $w = 0, \ldots, k$.

When the functions $f_{i,r}$ satisfy the Independent Types Property for all the phase types of a deprioritised algorithm, analysing the algorithm is significantly easier.

4.2.3 Using irreducible type sets

In this section, we consider the functions of Definition 4.2.1 obtained with $\mathcal{O} = \mathcal{M}^{(\tau)}$ and $\mathcal{O} = \{\tau, \ldots, k\}$.

Let

$$\mathrm{ix}(r, S) = \begin{cases} |\{w \in S \,:\, w \leq r\}| & \text{if } r \in S, \\ 0 & \text{if } r \notin S. \end{cases}$$

That is, if $r \in S$, then $\mathrm{ix}(r, S)$ is the index of r in S; otherwise $\mathrm{ix}(r, S) = 0$. Recall that $p_r^{(\tau)}$ are the functions obtained from (4.2.2) of Definition 4.2.1 using $\mathcal{O} = \mathcal{M}^{(\tau)}$.

Lemma 4.2.8. *Let $\mathcal{N} = \{\tau, \dots, k\}$ and let n_r be the functions* (4.2.2) *of Definition 4.2.1 with $\mathcal{O} = \mathcal{N}$. If the clutch matrix for a $\mathcal{N}$-clutch has non-zero determinant, then the clutch matrix for a $\mathcal{M}^{(\tau)}$-clutch has non-zero determinant, and for all $(x, y_0, \dots, y_m) \in D(\delta)$ with $y_{\tau+1} = \dots = y_k = 0$ we have*

$$n_r(x, y_0, \dots, y_m) = \begin{cases} 0 & \text{if } r \in \mathcal{N} \backslash \mathcal{M}^{(\tau)}, \\ p_r^{(\tau)}(x, y_0, \dots, y_m) & \text{if } r \in \mathcal{M}^{(\tau)}. \end{cases}$$

Moreover, if the Independent Types Property for a phase of type τ is satisfied and the clutch matrix for a $\mathcal{M}^{(\tau)}$-clutch has non-zero determinant, then the clutch matrix for a $\mathcal{N}$-clutch has non-zero determinant.

Proof. Fix $(x, y_0, \dots, y_m) \in D(\delta)$ with $y_{\tau+1} = \dots = y_k = 0$; all functions are evaluated at $(x, y_0, \dots, y_m)$, although we do not make this explicit. Let $a = |\mathcal{N}|$ and let $\mathcal{N} \backslash \mathcal{M}^{(\tau)} = \{w_1, \dots, w_b\}$ where $w_1 < \dots < w_b$. So operations of types $w_1, \dots, w_b$ occur in $\mathcal{N}$-clutches but not in $\mathcal{M}^{(\tau)}$-clutches; note that $\tau \notin \{w_1, \dots, w_b\}$. Let $\mathcal{D}$ be the clutch matrix for a $\mathcal{N}$-clutch and $\mathcal{C}^{(\tau)}$ be the clutch matrix for a $\mathcal{M}^{(\tau)}$-clutch.

We start by considering $\mathcal{D}$. Let σ be the permutation of $\{1, \dots, a\}$ that maps $\mathrm{ix}(w_i, \mathcal{N})$ to i (for $i = 1, \dots, b$) and maps $\mathrm{ix}(w, \mathcal{N})$ to $\mathrm{ix}(w, \mathcal{M}^{(\tau)}) + b$ for $w \in \mathcal{M}^{(\tau)}$. Then permuting both the rows and columns of $\mathcal{D}$ by σ we obtain the matrix

$$\widehat{\mathcal{D}} = \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{O} & \mathbf{C} \end{pmatrix}$$

where

$$\mathbf{A} = \begin{pmatrix} f_{w_1,w_1} & \cdots & f_{w_b,w_1} \\ \vdots & & \vdots \\ f_{w_1,w_b} & \cdots & f_{w_b,w_b} \end{pmatrix}, \tag{4.2.8}$$

$\mathbf{B}$ is a $b \times (a-b)$ matrix, $\mathbf{O}$ is a $(a-b) \times b$ matrix, and

$$\mathbf{C} = \mathcal{D}(\{\mathrm{ix}(w_1, \mathcal{N}), \dots, \mathrm{ix}(w_b, \mathcal{N})\}, \{\mathrm{ix}(w_1, \mathcal{N}), \dots, \mathrm{ix}(w_b, \mathcal{N})\}) = \mathcal{C}^{(\tau)}.$$

By Lemma 4.2.6, each entry of $\mathbf{O}$ is zero.

As the permutation applied to the rows is the same as that applied to the columns, we have

$$\det \mathcal{D} = \det \widehat{\mathcal{D}} = \det \mathbf{A} \det \mathcal{C}^{(\tau)}.$$

Since $\det \mathcal{D} \neq 0$ we have $\det \mathbf{A} \neq 0$ and $\det \mathcal{C}^{(\tau)} \neq 0$. Moreover, if the Independent Types Property holds, then $A = \mathrm{diag}(-1, \ldots, -1)$ and so $\det \mathcal{C}^{(\tau)} \neq 0$ implies that $\det \mathcal{D} \neq 0$.

Now fix $r \in \mathcal{N} \backslash \mathcal{M}^{(\tau)}$. We show that $n_r = 0$. Let R be the rows of $\mathcal{D}$ with indices from $\{\mathrm{ix}(w, \mathcal{N}) : w \in \mathcal{M}^{(\tau)}\}$. By Lemma 4.2.6, each vector in R has a zero in columns

$$\mathrm{ix}(w_1, \mathcal{N}), \ldots, \mathrm{ix}(w_b, \mathcal{N}).$$

Note that these indices are all at least 2 as $\tau \notin \{w_1, \ldots, w_b\}$. Let $\widehat{R}$ be the vectors obtained by removing the first component of the vectors of R. There are $a - b$ vectors in $\widehat{R}$ and each vector appears as a row of $\mathcal{D}(\mathrm{ix}(r, \mathcal{N}), 1)$. So each vector of $\widehat{R}$ has at least b zeros, namely in columns

$$\mathrm{ix}(w_1, \mathcal{N}) - 1, \ldots, \mathrm{ix}(w_b, \mathcal{N}) - 1.$$

Hence the vectors of $\widehat{R}$ are contained in a subspace of dimension $a - b - 1$. So $\widehat{R}$ is linearly dependent and thus $\det \mathcal{D}(\mathrm{ix}(r, \mathcal{N}), 1) = 0$. Therefore

$$n_r = (-1)^{\mathrm{ix}(r,\mathcal{N})+1} \frac{\det \mathcal{D}(\mathrm{ix}(r, \mathcal{N}), 1)}{\det \mathcal{D}} = 0.$$

Now fix $r \in \mathcal{M}^{(\tau)}$ and let $i = \mathrm{ix}(r, \mathcal{N})$. Consider the matrix $\mathcal{D}(i, 1)$. Let $\mathcal{E}$ be the matrix obtained from $\mathcal{D}$ by replacing the entries in row i and column 1 with zeros, then placing a 1 in the entry with row i and column 1. Then $\det \mathcal{E} = (-1)^{i+1} \det \mathcal{D}(i, 1)$. The permutation taking $\mathcal{D}$ to $\widehat{\mathcal{D}}$ also takes $\mathcal{E}$ to

$$\widehat{\mathcal{E}} = \begin{pmatrix} \mathbf{A} & \widehat{\mathbf{B}} \\ \mathbf{O} & \widehat{\mathbf{C}} \end{pmatrix}$$

where $\mathbf{A}$ and $\mathbf{O}$ are as before, $\widehat{\mathbf{B}}$ is another $b \times (a - b)$ matrix, and $\widehat{\mathbf{C}}$ is the matrix obtained from $\mathcal{C}^{(\tau)}$ by replacing the entries of row $\mathrm{ix}(r, \mathcal{M}^{(\tau)})$ and column 1 with zeros, then placing a 1 in the entry with row $\mathrm{ix}(r, \mathcal{M}^{(\tau)})$ and column 1.

Therefore

$$(-1)^{i+1}\det\mathcal{D}(i,1) = \det\mathcal{E} = \det\widehat{\mathcal{E}}$$
$$= (-1)^{\mathrm{ix}(r,\mathcal{M}^{(\tau)})+1}\det\mathbf{A}\det\mathcal{C}^{(\tau)}(\mathrm{ix}(r,\mathcal{M}^{(\tau)}),1).$$

Then we have

$$n_r = (-1)^{i+1}\det\mathcal{D}(i,1)/\det\mathcal{D}$$
$$= (-1)^{\mathrm{ix}(r,\mathcal{M}^{(\tau)})+1}\frac{\det\mathbf{A}\det\mathcal{C}^{(\tau)}(\mathrm{ix}(r,\mathcal{M}^{(\tau)}),1)}{\det\mathbf{A}\det\mathcal{C}^{(\tau)}} = p_r^{(\tau)}.$$

This concludes the proof of the lemma. □

We also have a similar lemma concerning to the functions Φ_b of Definition 4.2.1.

Lemma 4.2.9. *Let $\mathcal{N} = \{\tau,\dots,k\}$. Denote the clutch matrices of a $\mathcal{N}$-clutch and a $\mathcal{M}^{(\tau)}$-clutch by $\mathcal{D}$ and $\mathcal{C}^{(\tau)}$ respectively. For $r \in \mathcal{N}$, define*

$$F(r) = \det\mathcal{D}(\{1,\dots,r-\tau\},\{1,\dots,r-\tau\})$$

and for $r \in \mathcal{M}^{(\tau)}$, define

$$E(r) = \det\mathcal{C}^{(\tau)}(\{1,\dots,\mathrm{ix}(r,\mathcal{M}^{(\tau)})-1\},\{1,\dots,\mathrm{ix}(r,\mathcal{M}^{(\tau)})-1\}).$$

Note that $F(r)$ and $E(r)$ are functions from $\mathbb{R}^{m+2}$ to $\mathbb{R}$. Also, for $w \in \mathcal{N}$ with $w \le \max\mathcal{M}^{(\tau)}$, let

$$\mu(w) = \min\{r \in \mathcal{M}^{(\tau)} \,:\, r \ge w\}.$$

Then, for $w \in \mathcal{N}$ and for all $\mathbf{x} = (x, y_0, \dots, y_m) \in D(\delta)$ with $y_{\tau+1} = \cdots = y_k = 0$, there exists a function $C_w\colon \mathbb{R}^{m+2} \to \mathbb{R}$ such that

$$F(w)(\mathbf{x}) = \begin{cases} C_w(\mathbf{x})E(\mu(w))(\mathbf{x}) & \text{if } w \le \max\mathcal{M}^{(\tau)}, \\ C_w(\mathbf{x}) & \text{if } w > \max\mathcal{M}^{(\tau)}. \end{cases}$$

Moreover, if the Independent Types Property for a phase of type τ is satisfied, then

$$C_w = \begin{cases} (-1)^{k-w-\left|\mathcal{M}^{(\tau)}\right|+\mathrm{ix}\left(\mu(w),\mathcal{M}^{(\tau)}\right)} & \text{if } w \le \max\mathcal{M}^{(\tau)}, \\ (-1)^{k-w+1} & \text{if } w > \max\mathcal{M}^{(\tau)}. \end{cases}$$

Proof. The proof is similar to the proof of Lemma 4.2.8. So fix $(x, y_0, \ldots, y_m) \in D(\delta)$ with $y_{\tau+1} = \cdots = y_k = 0$; again, all functions are evaluated at $(x, y_0, \ldots, y_m)$ although we do not make this explicit. Define $\widehat{\mathcal{D}}$ to be the matrix obtained by permuting the rows and columns of $\mathcal{D}$ according to σ (where σ is defined in the proof of Lemma 4.2.8). Let $a = |\mathcal{N}|$ and let $b = \left|\mathcal{N} \backslash \mathcal{M}^{(\tau)}\right|$. Recall from the proof of Lemma 4.2.8 that

$$\widehat{\mathcal{D}} = \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{O} & \mathbf{C} \end{pmatrix}$$

where $\mathbf{A}$ is a $b \times b$ matrix (as given by (4.2.8)), $\mathbf{B}$ is a $b \times (a-b)$ matrix, $\mathbf{O}$ is an $(a-b) \times b$ zero matrix, and $\mathbf{C} = \mathcal{C}^{(\tau)}$. Now fix $w \in \mathcal{N}$. Then

$$F(w) = \det \widehat{\mathcal{D}}(\{\sigma(1), \ldots, \sigma(w-\tau)\}, \{\sigma(1), \ldots, \sigma(w-\tau)\}).$$

Let $\xi_w = \left|\{\tau, \ldots, w-1\} \backslash \mathcal{M}^{(\tau)}\right|$ and let $\chi_w = \left|\{\tau, \ldots, w-1\} \cap \mathcal{M}^{(\tau)}\right|$. Then

$$\sigma(\{1, \ldots, w-\tau\}) = \{1, \ldots, \xi_w\} \cup \{b+1, \ldots, b+\chi_w\}.$$

Therefore

$$F(w) = \det \begin{pmatrix} \widehat{\mathbf{A}} & \widehat{\mathbf{B}} \\ \widehat{\mathbf{O}} & \widehat{\mathbf{C}} \end{pmatrix}$$

where

- $\widehat{\mathbf{A}} = \mathbf{A}(\{1, \ldots, \xi_w\}, \{1, \ldots, \xi_w\})$,
- $\widehat{\mathbf{B}} = \mathbf{B}(\{1, \ldots, \xi_w\}, \{1, \ldots, \chi_w\})$,
- $\widehat{\mathbf{O}} = \mathbf{O}(\{1, \ldots, \chi_w\}, \{1, \ldots, \xi_w\})$, and
- $\widehat{\mathbf{C}} = \mathcal{C}^{(\tau)}(\{1, \ldots, \chi_w\}, \{1, \ldots, \chi_w\})$.

Now, if $w > \max \mathcal{M}^{(\tau)}$, then $\chi_w = \left|\mathcal{M}^{(\tau)}\right|$ and $\widehat{\mathbf{C}}$ is the empty matrix (which by convention has determinant 1). Thus $F(w) = \det \widehat{\mathbf{A}}$. Otherwise $w \leq \max \mathcal{M}^{(\tau)}$ and

$$\chi_w + 1 = \left|\{r \in \mathcal{M}^{(\tau)} : r \leq w-1\}\right| + 1 = \mathrm{ix}(\mu(w), \mathcal{M}^{(\tau)}),$$

thus

$$\begin{aligned} F(w) &= \det \widehat{\mathbf{A}} \det \mathcal{C}^{(\tau)}(\{1,\dots,\chi_w\},\{1,\dots,\chi_w\}) \\ &= \det \widehat{\mathbf{A}}\, E(\mu(w)). \end{aligned}$$

Now assume that the Independent Types Property is satisfied. Then $\widehat{A}$ is diagonal of order $\left|\{w,\dots,k\}\backslash\mathcal{M}^{(\tau)}\right|$ and each diagonal entry is -1. If $w > \max \mathcal{M}^{(\tau)}$, then $\left|\{w,\dots,k\}\backslash\mathcal{M}^{(\tau)}\right| = k - w + 1$ and so

$$C_w = \det \widehat{\mathbf{A}} = (-1)^{k-w+1}.$$

Otherwise,

$$\begin{aligned} \left|\{w,\dots,k\}\backslash\mathcal{M}^{(\tau)}\right| &= k - w + 1 - \left|\{r \in \mathcal{M}^{(\tau)} \,:\, r \geq w\}\right| \\ &= k - w - \left|\mathcal{M}^{(\tau)}\right| + \mathrm{ix}(\mu(w),\mathcal{M}^{(\tau)}) \end{aligned}$$

and so

$$C_w = (-1)^{k-w-\left|\mathcal{M}^{(\tau)}\right|+\mathrm{ix}(\mu(w),\mathcal{M}^{(\tau)})}.$$

□

Lemma 4.2.8 and Lemma 4.2.9 suggest that operations of types from $\mathcal{N}\backslash\mathcal{M}^{(\tau)}$ do not affect the behaviour of a $\mathcal{N}$-clutch. We may use either $\mathcal{N}$ or $\mathcal{M}^{(\tau)}$ to define the type distribution of the deprioritised algorithm. We choose to define the deprioritised algorithm using irreducible type sets.

To complete the definition of the deprioritised algorithm, we must specify the phases of the algorithm. We do so as we define functions that approximate the scaled random variables of the algorithm. But first we consider how to determine the irreducible type sets for algorithms defined on random d-in d-out digraphs.

4.2.4 Determining irreducible type sets

For random d-in d-out digraphs, we do not require the expected change digraphs to determine the irreducible type sets. Instead we can use a method called the

Degree Pair Progress Digraph (or DPPD) method, which we describe in this section. The DPPD method should be easier to apply than using expected change digraphs directly. Although we consider only algorithms for random d-in d-out digraphs in this section, the main ideas could be adapted to other algorithms.

The DPPD method, as its name suggests, is based on how the degree pairs of vertices change during an operation of a given deprioritisable algorithm. Note that we now require prioritised algorithms to be deprioritisable . We start by defining a digraph called the *degree pair progress digraph*, which can be defined for every deprioritisable algorithm on a random d-in d-out digraph.

Definition 4.2.10 (Degree Pair Progress Digraph). Consider a deprioritisable algorithm on a random d-in d-out digraph defining the random process $\{G_t\}$. For each degree pair (i, j), let $B_{(i,j)}$ be the minimal set of degree pairs such that, for all $t \geq 1$, if vertex v has degree pair (i, j) in G_t, then a.a.s. v has degree pair $(p, q) \in B_{(i,j)} \cup \{(i, j)\}$ in G_{t-1}. Then the *degree pair progress digraph* is the digraph with the vertex set

$$\{(i, j) \,:\, 0 \leq i, j \leq d\}$$

and the edge set

$$\bigcup_{(i,j)} \{ \big((p, q), (i, j)\big) \,:\, (p, q) \in B_{(i,j)} \}.$$

For example, consider the algorithm DominatingSet2 described in Chapter 3. During an operation of DominatingSet2, the degree pair of a vertex v changes only if v is saturated or a randomly selected free point associated with v is exposed. As the number of free points exposed by an operation is bounded (independently of n), a.a.s. no vertex has two free points randomly selected and exposed during an operation.

Hence, if v has degree pair (i, j) (with $(i, j) \neq (2, 2)$) after an operation, then before that operation a.a.s. v had degree pair $(p, q) \in B_{(i,j)}$ where

$$B_{(i,j)} = \{(i - 1, j), (i, j - 1)\} \cap \{(p, q) \,:\, 0 \leq p, q \leq 2\}.$$

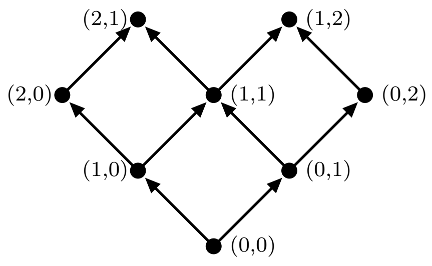


Figure 4.2.1: The Degree Pair Progress Digraph for DominatingSet2 (with vertex $(2,2)$ removed).

For $(i,j) = (2,2)$ we have $B_{(2,2)} = \{(i,j) : 0 \le i,j \le 2\} \backslash \{(2,2)\}$. Note that by definition $(i,j) \notin B_{(i,j)}$. The DPPD for DominatingSet2 is shown in Figure 4.2.1; since vertex $(2,2)$ has no out-neighbours but many in-neighbours, we do not show vertex $(2,2)$.

To apply the DPPD method, the random variables $Y_0, \ldots, Y_m$ (from Definition 4.1.1 (iii)) must include the random variables $Z_{(i,j)}$ which count the number of vertices of degree pair (i,j); moreover, each operation type must correspond to a degree pair.

So we define a function

$$\mathrm{dp}\colon \{0,\ldots,k\} \to \{(i,j) : 0 \le i,j \le d\}$$

such that $Y_r = Z_{\mathrm{dp}(r)}$, and a function

$$\nu\colon \{(i,j) : 0 \le i,j \le d\} \to \{0,\ldots,m\}$$

such that $Z_{(i,j)} = Y_{\nu(i,j)}$. Note that $\nu(\mathrm{dp}(r)) = r$ for $r \in \{0,\ldots,k\}$. We use these functions to go between the general setting of Definition 4.1.1 and the specific setting of algorithms for random d-in d-out digraphs.

For example, the algorithm DominatingSet2 has four types of operations. In Chapter 3, we took these types to be the degree pairs $(0,0)$, $(0,1)$, $(0,2)$, and $(1,2)$. Note that we now consider the type $(0,0)$ to have a priority less than the priority of the types $(0,1)$, $(0,2)$, and $(1,2)$. To analyse DominatingSet2 in the general setting of this chapter, we must use the types 0, 1, 2, and 3. The two sets of types are related via ν and dp in the following way: $\nu(0,0) = 0$, $\nu(0,1) = 1$, $\nu(0,2) = 2$, and $\nu(1,2) = 3$. The definition of ν may be completed in any way.

We can determine the irreducible type sets using the degree pair progress digraph. Consider a phase of type τ. We colour black those vertices corresponding (via dp and ν) to types greater than τ. The remaining vertices, including the vertex $\mathrm{dp}(\tau)$, are coloured white. For example, Figure 4.2.2 shows the DPPD for DominatingSet2 coloured for a phase of type 1. Let $\mathcal{O}^{(\tau)}$ be the subset of $\{\tau, \dots, k\}$ such that $r \in \mathcal{O}^{(\tau)}$ if and only if the vertex $\mathrm{dp}(r)$ or any of the in-neighbours of $\mathrm{dp}(r)$ are coloured white. In Lemma 4.2.11, given below, we prove that under certain conditions the irreducible type set for a phase of type τ is $\mathcal{O}^{(\tau)}$.

To apply the DPPD method, properties of the functions $f_{i,r}$ must be captured by the DPPD. Fix a phase type τ. Recall that $f_{i,r} = g_{i,r}/H^{\ell_i}$ for some polynomials $g_{i,r}$ and H, and some $\ell_i \geq 1$. Let $\Sigma = \{(b, a_0, \dots, a_m) \in \mathbb{Z}^{m+2} : b, a_0, \dots, a_m \geq 0\}$. Then consider the polynomial $g_{i,r} + \delta_{i,r} H^{\ell_i}$: we have

$$g_{i,r} + \delta_{i,r} H^{\ell_i} = \sum_{(b,a_0,\dots,a_m)\in\Sigma} C(b, a_0, \dots, a_m) x^b y_0^{a_0} \cdots y_m^{a_m}$$

for real numbers $C(b, a_0, \dots, a_m)$. We say that $f_{i,r}$ is $B_{\mathrm{dp}(i)}$-*determined* if

(a) for all $(b, a_0, \dots, a_m) \in \Sigma$ with $C(b, a_0, \dots, a_m) \neq 0$, there exists $j \in \{i\} \cup \nu(B_{\mathrm{dp}(i)})$ such that $a_j > 0$, and

(b) for $(p, q) \in B_{\mathrm{dp}(i)}$, there exists $b, a_0, \dots, a_m$ with $a_{\nu(p,q)} > 0$ and $a_j = 0$ for $j = \tau + 1, \dots, k$, such that $C(b, a_0, \dots, a_m) \neq 0$.

If $f_{i,r}$ is $B_{\mathrm{dp}(i)}$-determined for all $i \in \{\tau, \dots, k\}$ and for all $r \in \{0, \dots, k\}$, then the DPPD can be used to determine the irreducible types sets.

Lemma 4.2.11. *Define $\mathcal{O}^{(\tau)}$ as above and recall that $\mathcal{M}^{(\tau)}$ is the irreducible type set for a phase of type τ (see Definition 4.2.5). If $f_{i,r}$ is $B_{\mathrm{dp}(i)}$-determined for all $i \in \{\tau, \dots, k\}$ and for all $r \in \{0, \dots, k\}$, then $\mathcal{O}^{(\tau)} = \mathcal{M}^{(\tau)}$ and the Independent Types Property is satisfied for a phase of type τ.*

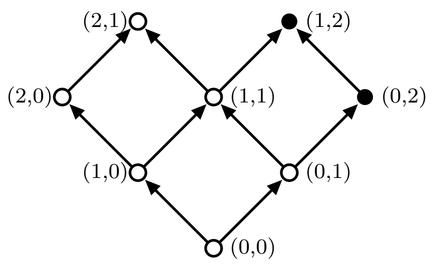


Figure 4.2.2: The Degree Pair Progress Digraph for DominatingSet2 (with the vertex $(2, 2)$ removed) coloured for a phase of type 1.

Proof. Notice that, in the coloured DPPD (for a phase of type τ), a vertex (i, j) is coloured black if and only if $\nu(i, j) \in \{\tau + 1, \ldots, k\}$. Also $\tau \in \mathcal{O}^{(\tau)}$ and $\tau \in \mathcal{M}^{(\tau)}$ by the definitions of $\mathcal{O}^{(\tau)}$ and $\mathcal{M}^{(\tau)}$.

Consider $w \in \mathcal{O}^{(\tau)}$ with $w \neq \tau$. Then in the coloured DPPD, the vertex $\mathrm{dp}(w)$ has at least one white in-neighbour. That is, there exists $(p, q) \in B_{\mathrm{dp}(w)}$ such that $\nu(p, q) \notin \{\tau + 1, \ldots, k\}$. Now $f_{w,\tau}$ is $B_{\mathrm{dp}(w)}$-determined: so by part (b), setting $y_i = 0$ in $f_{w,\tau}$ for $i = \tau + 1, \ldots, k$ we obtain a non-zero rational function. Note that $\tau + 1 > 0$. Hence there exists $(x, y_0, \ldots, y_m) \in D(\delta)$ with $y_{\tau+1} = \cdots = y_k = 0$ and $f_{w,\tau}(x, y_0, \ldots, y_m) \neq 0$. So in the ECD there is an edge from τ to w and thus $w \in \mathcal{M}^{(\tau)}$.

Now consider $w \in \{\tau+1, \ldots, k\} \backslash \mathcal{O}^{(\tau)}$. In the coloured DPPD, all the in-neighbours of the vertex $\mathrm{dp}(w)$, and the vertex $\mathrm{dp}(w)$, are coloured black. Hence, for all $(p, q) \in B_{\mathrm{dp}(w)} \cup \{\mathrm{dp}(w)\}$, we have $\nu(p, q) \in \{\tau + 1, \ldots, k\}$. Fix $u \in \{0, \ldots, k\}$. Now $f_{w,u}$ is $B_{\mathrm{dp}(w)}$-determined: so by part (a), for all $(x, y_0, \ldots, y_m) \in D(\delta)$ with

$$y_{\tau+1} = \cdots = y_k = 0$$

we have $f_{w,u}(x, y_0, \ldots, y_m) = -\delta_{w,u}$. Hence, in the ECD, vertex w has zero in-degree and so $w \notin \mathcal{M}^{(\tau)}$. Therefore $\mathcal{O}^{(\tau)} = \mathcal{M}^{(\tau)}$ and, moreover, the Independent Types Property is satisfied for a phase of type τ. $\square$

The DPPD for DominatingSet2 coloured for a phase of type 1 is given in Figure 4.2.2. Applying Lemma 4.2.11 we can see that the irreducible type set for a phase of type 1 is $\{1, 2, 3\}$, which corresponds to the set of degree pairs $\{(0, 1), (0, 2), (1, 2)\}$. Later, in Chapter 6, we shall see examples where $\mathcal{M}^{(\tau)} \neq \{\tau, \ldots, k\}$.

With a slight adjustment, the DPPD method could be used for algorithms for random regular undirected graphs. Many such algorithms use random variables counting the number of vertices of a given degree. In this case the corresponding DPPD would be a directed path, with edges from vertex i to vertex $i + 1$. Assuming the DPPD method is valid, the irreducible type sets would have size 1 or 2. Indeed, this is assumed in Wormald's [61] earlier analysis of deprioritised algorithms. For the algorithms we consider, most irreducible type sets have size greater than two.

4.3 The Differential Equations

We now define functions that, under certain conditions, asymptotically approximate the scaled random variables of the deprioritised algorithm. The behaviour of these functions depends on the phase of the algorithm and so are defined piecewise, with each piece corresponding to a phase. The functions for each piece are the solutions to a system of differential equations defined using the functions $f_{i,r}$ and the type distribution given by $p_r^{(\tau)}$. For each phase, we choose the domain on which we solve the differential equations so that the solutions do indeed approximate the scaled random variables; the domain also determines the phases of the deprioritised algorithm. From the functions defined in this section, asymptotic properties of the deprioritised algorithm are determined.

The differential equations are suggested by the expected changes in the random variables $Y_0, \ldots, Y_m$ due to an operation. For the rest of this chapter, we consider the random process defined by the deprioritised algorithm. Consider a phase of type τ. The expected change in the random variable Y_i due to the operation at step t (occurring in a phase of type τ) is

$$\begin{aligned}\mathbb{E}(Y_i(t+1) - Y_i(t) \mid \mathrm{op}_{t+1}) &= \sum_{r=0}^{k} \mathbb{E}(Y_i(t+1) - Y_i(t) \mid \tau(\mathrm{op}_{t+1}) = r)\mathbb{P}(\tau(\mathrm{op}_{t+1}) = r) \\ &= \sum_{r=0}^{k} f_{i,r}(t) p_r^{(\tau)}(t) + o(1).\end{aligned}$$

Thus, for a phase of type τ, we use the differential equations

$$\frac{dy_i}{dx}(x;\tau) = \sum_{r=0}^{k} f_{i,r}(x) p_r^{(\tau)}(x) \tag{4.3.1}$$

for $0 \leq i \leq m$. Solving these differential equations with appropriate initial conditions we obtain functions that, under certain conditions (including Lipschitz conditions), a.a.s. approximate the scaled random variables in a phase of type τ.

The domain on which we solve the differential equations is defined as the intersection of three subsets of $\mathbb{R}^{m+2}$. Of course, the differential equations must be Lipschitz on this domain. Let C be a constant such that Cn is an upper bound on the number of operations performed by the deprioritised algorithm. Recall the constant M and polynomial H from the definition of deprioritisable algorithms (Definition 4.1.1). In particular, M is an upper bound on each $|Y_i|/n$ and the denominator of each $f_{i,r}$ is some power of the polynomial H. Recall that

$$\begin{aligned}D(\delta) = \{(x, y_0, \ldots, y_m) \,:\, -\delta < x < C,\ y_0 > \delta,\ H > \delta,& \\ |y_i| < 2M \quad (i = 0, \ldots, m)\}.&\end{aligned}$$

On $D(\delta)$ the functions $f_{i,r}$ are Lipschitz, which we prove later in this section. The parameter δ expresses how close to the end of the deprioritised algorithm we can approximate the random variables; while the condition $y_0 > \delta$ allows the preprocessing subphases to be performed.

Recall that $\mathcal{C}^{(\tau)}$ is the clutch matrix for an $\mathcal{M}^{(\tau)}$-clutch. Now for $b = 1, \dots, \left|\mathcal{M}^{(\tau)}\right|$, let

$$q_b^{(\tau)} = (-1)^{\left|\mathcal{M}^{(\tau)}\right|+b} \det \mathcal{C}^{(\tau)}(b, 1). \tag{4.3.2}$$

We then define

$$\begin{aligned} L^{(\tau)}(\delta) = \{(x, y_0, \dots, y_m) \,:\, (-1)^{\left|\mathcal{M}^{(\tau)}\right|+1} \det \mathcal{C}^{(\tau)} > \delta, \\ q_b^{(\tau)} > 0 \quad (b = 1, \dots, \left|\mathcal{M}^{(\tau)}\right|)\}. \end{aligned}$$

By Lemma 4.2.3, the definition of $L^{(\tau)}(\delta)$ ensures that, on $D(\delta) \cap L^{(\tau)}(\delta)$, the differential equations (4.3.1) are Lipschitz and the functions $p_r^{(\tau)}$ define a probability distribution.

Now for $b = 2, \dots, \left|\mathcal{M}^{(\tau)}\right|$, we let

$$E_b^{(\tau)} = (-1)^{\left|\mathcal{M}^{(\tau)}\right|-b+1} \det \mathcal{C}^{(\tau)}(\{1, \dots, b-1\}, \{1, \dots, b-1\}). \tag{4.3.3}$$

Then we define

$$A^{(\tau)} = \{(x, y_0, \dots, y_m) \,:\, y_\tau > 0,\ E_b^{(\tau)} > 0 \quad (b = 2, \dots, \left|\mathcal{M}^{(\tau)}\right|)\}.$$

The domain $A^{(\tau)}$ defines the boundaries of a phase of type τ. We solve the differential equations (4.3.1) on the domain $V^{(\tau)}(\delta) = D(\delta) \cap L^{(\tau)}(\delta) \cap A^{(\tau)}$.

Solutions to the differential equations (4.3.1) for each phase are combined to define functions $\hat{y}_i$ $(i = 0, \dots, m)$ that, under certain conditions, a.a.s. approximate the scaled random variables for nearly the entire deprioritised algorithm. The functions $\hat{y}_i$ are defined piecewise, with piece j given by functions $y_i^{(j)}$ (for $i = 0, \dots, m$) which approximate the scaled random variables during phase j of type τ_j. So to define $\hat{y}_i$, we need to know the types of the phases of the deprioritised algorithm. We assume that the type of the first phase is given; it can be determined from the prioritised algorithm. For subsequent phases, the phase type is determined by the functions $y_i^{(j)}$ which correspond to the previous phase. We expect, but do not prove, that the number and types of phases are the same for both the prioritised and deprioritised algorithms.

Definition 4.3.1 (Functions $\hat{y}_i$). Let $x_0 = 0$ and define $\hat{y}_i(x_0) = \lim_{n\to\infty} Y_i(0)/n$ for $i = 0, \dots, m$ (recall that these limits exist for deprioritisable algorithms). Fix $\delta > 0$ and assume that τ_1 is given and satisfies

$$\hat{y}_{\tau_1+1}(x_0) = \cdots = \hat{y}_k(x_0) = 0.$$

Denote the closure of $V^{(\tau)}(\delta)$ by $\overline{V}^{(\tau)}(\delta)$. Assume that $j \geq 1$ and that τ_j is defined. If

$$(x_{j-1}, \hat{y}_0(x_{j-1}), \dots, \hat{y}_m(x_{j-1})) \notin \overline{V}^{(\tau_j)}(\delta),$$

then there is no phase j, so set $K = j - 1$ and finish the definition of $\hat{y}_i$. Otherwise let $y_0^{(j)}, \dots, y_m^{(j)}$ be the solutions to the differential equations (4.3.1) with $\tau = \tau_j$ and initial conditions $y_i^{(j)}(x_{j-1}) = \hat{y}_i(x_{j-1})$ (for $i = 0, \dots, m$) on some open set containing $V^{(\tau)}(\delta)$ (the Lipschitz property guarantees that a unique solution exists). Then define x_j to be the infimum of those $x > x_{j-1}$ for which

$$(x, y_0^{(j)}(x), \dots, y_m^{(j)}(x)) \notin V^{(\tau_j)}(\delta).$$

We extend $\hat{y}_i$ by defining $\hat{y}_i(x) = y_i^{(j)}(x)$ for $x \in [x_{j-1}, x_j]$. Whether there is a next phase, and if so its type, is determined by the following steps:

(i) if $(x_j, y_0^{(j)}(x_j), \dots, y_m^{(j)}(x_j))$ lies on the boundary of $D(\delta)$, then there is no phase $j+1$, so set $K = j$ and finish the definition of $\hat{y}_i$; otherwise

(ii) if $(x_j, y_0^{(j)}(x_j), \dots, y_m^{(j)}(x_j))$ lies on the boundary of

$$\{(x, y_0, \dots, y_m) \, : \, E_b^{(\tau_j)} > 0 \text{ for } b = 2, \dots, |\mathcal{M}^{(\tau_j)}|\},$$

then we set τ_{j+1} to be the maximum $r \in \mathcal{M}^{(\tau_j)}$ such that $E_b^{(\tau_j)}(x_j) = 0$ where $b = \mathrm{ix}(r, \mathcal{M}^{(\tau_j)})$; otherwise

(iii) if $y_{\tau_j}^{(j)}(x_j) = 0$, then we set τ_{j+1} to be the maximum $r \in \{0, \dots, k\}$ such that $y_r^{(j)}(x_j) > 0$; otherwise

(iv) there is no phase $j+1$, so set $K = j$ and finish the definition of $\hat{y}_i$.

Note that order is important in the above steps, and that τ_j may be defined even if $K = j - 1$.

The above definition allows for a more general sequence of phases that allowed in Wormald's [61] previous analysis. In particular, we allow multiple phases of the same type and we allow phases of type 0. Note that a phase of type 0 is different from a preprocessing subphase, since operations of types greater than 0 may (and usually do) occur. The next corollary gives some useful properties of the phase types defined by Definition 4.3.1.

Corollary 4.3.2. *For some $j \geq 2$, assume that τ_j is defined according to Definition 4.3.1. Then the following are true:*

- *the point $(x_{j-1}, \hat{y}_0(x_{j-1}), \ldots, \hat{y}_m(x_{j-1}))$ lies in $D(\delta)$,*
- *if $\tau_j > \tau_{j-1}$, then $\tau_j \in \mathcal{M}^{(\tau_{j-1})}$ and for $\beta = \mathrm{ix}(\tau_j, \mathcal{M}^{(\tau_{j-1})})$ we have*
$$E_{\beta}^{(\tau_{j-1})}(x_{j-1}) = 0 \text{ and } E_b^{(\tau_{j-1})}(x_{j-1}) > 0$$
for $b = \beta + 1, \ldots, \left|\mathcal{M}^{(\tau_{j-1})}\right|$, and
- *if $\tau_j < \tau_{j-1}$, then $E_b^{(\tau_{j-1})}(x_{j-1}) > 0$ for $b = 2, \ldots, \left|\mathcal{M}^{(\tau_{j-1})}\right|$.*

Later in the chapter we describe some properties of the functions $\hat{y}_i$. First though, we prove that the differential equations (4.3.1) are Lipschitz on the domain $V^{(\tau)}(\delta)$. We also prove that the functions defining the domain $V^{(\tau)}(\delta)$, and their derivatives, are Lipschitz.

4.3.1 Proving the Lipschitz property

To prove the Lipschitz property of a function we use the following standard lemma [43, Chapter 2].

Lemma 4.3.3. *Let $R \subset \mathbb{R}^a$ be embedded in a larger region $D \subseteq \mathbb{R}^a$ such that every point of R is at least a (Euclidean) distance of ρ from the boundary of D. If $f = f(x_1, \ldots, x_a) \colon \mathbb{R}^a \to \mathbb{R}$ and $\partial f / \partial x_i$ $(i = 0, \ldots, a)$ are bounded by M and N respectively on D, then f satisfies a Lipschitz condition on R with Lipschitz constant*
$$\max\left(\frac{2M}{\rho}, aN\right).$$

We first show that each $f_{i,r}$ is Lipschitz in the variables $x, y_0, \ldots, y_m$ on the domain $D(\delta)$. From Definition 4.1.1, the functions $f_{i,r}$ of a deprioritisable algorithm can be written as $g_{i,r}/H^a$ where $g_{i,r}$ and H are polynomials in $x, y_0, \ldots, y_m$ and a is a positive integer. The partial derivatives of each $f_{i,r}$ have a similar form: a polynomial in $x, y_0, \ldots, y_m$ divided by some positive power of H. So the functions $f_{i,r}$ and their partial derivatives are bounded on the domain

$$\begin{aligned}\widehat{D}(\delta) = \{(x, y_0, \ldots, y_m) \,:\, -3\delta/2 \le x \le C + \delta/2,\ y_0 \ge \delta/2,\ H \ge \delta/2,\\ |y_i| \le 2M + \delta/2 \text{ for } i = 0, \ldots, m.\}\end{aligned}$$

Every point of $D(\delta)$ is at least a distance of $\delta/2$ from the boundary of $\widehat{D}(\delta)$. So by Lemma 4.3.3 the functions $f_{i,r}$ are Lipschitz on $D(\delta)$.

Now consider the functions dy_i/dx given by (4.3.1). Similarly to the above, each function dy_i/dx is a polynomial in $x, y_0, \ldots, y_m$ divided by some positive power of H and some positive power of $\det \mathcal{C}^{(\tau)}$. The partial derivatives of each dy_i/dx also have the same form. So let

$$\widehat{L}^{(\tau)}(\delta) = \{(x, y_0, \ldots, y_m) \,:\, \det \mathcal{C}^{(\tau)} \ge \delta/2\}.$$

Then each dy_i/dx and their partial derivatives are bounded on $\widehat{D}(\delta) \cap \widehat{L}^{(\tau)}(\delta)$. Every point of $D(\delta) \cap L^{(\tau)}(\delta)$ is at least a distance of $\delta/2$ from the boundary of $\widehat{D}(\delta) \cap \widehat{L}^{(\tau)}(\delta)$; so by Lemma 4.3.3 the functions dy_i/dx are Lipschitz on $D(\delta) \cap L^{(\tau)}(\delta)$.

The same argument shows that higher order derivatives of each dy_i/dx are also Lipschitz on $D(\delta) \cap L^{(\tau)}(\delta)$. Moreover, the same argument shows that the functions defining the domain $V^{(\tau)}(\delta)$, and their derivatives, are Lipschitz on $D(\delta) \cap L^{(\tau)}(\delta)$.

Note that for any $\delta > 0$ and $\tau \in \{1, \ldots, k\}$ we can always find some open and bounded set W containing $D(\delta) \cap L^{(\tau)}(\delta)$ on which the functions dy_i/dx are Lipschitz. So the solutions to the differential equations given by (4.3.1) exist on a domain larger than require. This is useful when solving the differential equations numerically. We now consider properties of the functions $\hat{y}_i$.

4.3.2 Properties of the differential equations

In this section we note some properties of the functions $\hat{y}_i$. These properties are useful when applying the theory presented in this chapter and Chapter 5. We start by considering the differential equations, given by (4.3.1), for a phase of type τ.

Lemma 4.3.4. *Let dy_i/dx be defined by* (4.3.1)*. For all $(x, y_0, \dots, y_m) \in D(\delta)$, we have*

$$\frac{dy_i}{dx}(x, y_0, \dots, y_m) = 0$$

for $i \in \mathcal{M}^{(\tau)} \backslash \{\tau\}$. Furthermore, if $y_{\tau+1} = \cdots = y_k = 0$ also, then for $i = \tau+1, \dots, k$ we have

$$\frac{dy_i}{dx}(x, y_0, \dots, y_m) = 0.$$

Proof. Let $\mathcal{M}^{(\tau)} = \{w_1, \dots, w_a\}$ where $w_1 < \cdots < w_a$. Recall that $\mathcal{C}^{(\tau)}$ is the clutch matrix for an $\mathcal{M}^{(\tau)}$-clutch. Then

$$\det \mathcal{C}^{(\tau)} \frac{dy_i}{dx} = \det \mathbf{A} \quad \text{where} \quad \mathbf{A} = \begin{pmatrix} f_{i,w_1} & f_{w_2,w_1} & \cdots & f_{w_a,w_1} \\ \vdots & \vdots & & \vdots \\ f_{i,w_a} & f_{w_2,w_a} & \cdots & f_{w_a,w_a} \end{pmatrix}.$$

For $i \in \mathcal{M}^{(\tau)}$ with $i > \tau$, the matrix A has two equal columns, and so $dy_i/dx = 0$. While for $i \in \{\tau+1, \dots, k\} \backslash \mathcal{M}^{(\tau)}$, by Lemma 4.2.6, when $y_{\tau+1} = \cdots = y_k = 0$ the first column of $\mathbf{A}$ is all zero. So again $dy_i/dx = 0$. $\square$

Using Lemma 4.3.4 and the definition of the phase types (from Definition 4.3.1), we prove below that for each phase j, the functions $\hat{y}_{\tau_j+1}, \dots, \hat{y}_k$ are zero on the interval $[x_{j-1}, x_j]$. We also note that $\hat{y}_{\tau_j}(x_{j-1}) = 0$ when $\tau_j > \tau_{j-1}$, which we will find useful in the next chapter.

Lemma 4.3.5. *For $j = 1, \dots, K$, where K is defined in Definition 4.3.1, we have*

$$\hat{y}_{\tau_j+1}(x) = \cdots = \hat{y}_k(x) = 0$$

for all $x \in [x_{j-1}, x_j]$. Therefore, for $j > 1$, if $\tau_j > \tau_{j-1}$, then $\hat{y}_{\tau_j}(x_{j-1}) = 0$.

Proof. We prove the lemma by induction on j. So assume that for phase j we have

$$\hat{y}_{\tau_j+1}(x_{j-1}) = \cdots = \hat{y}_k(x_{j-1}) = 0.$$

This is true for phase 1 from Definition 4.3.1. Consider the differential equations obtained by setting $y_{\tau+1}, \ldots, y_k$ to zero in dy_i/dx given by (4.3.1). These modified differential equations are Lipschitz on $V^{(\tau_j)}(\delta)$ for the same reasons given in Section 4.3.1. Thus the modified differential equations have a unique solution which, by Lemma 4.3.4, extend to the unique solution of the original differential equations dy_i/dx. Thus for $i = \tau_j + 1, \ldots, k$ we have $\hat{y}_i(x) = y_i^{(j)}(x) = 0$ for $x \in [x_{j-1}, x_j]$. Now consider phase $j+1$. If $\tau_{j+1} > \tau_j$, then we have already shown that $\hat{y}_i(x_j) = 0$ for $i = \tau_{j+1} + 1, \ldots, k$; on the other hand, if $\tau_{j+1} < \tau_j$, then by definition we have $\hat{y}_i(x_j) = 0$ for $i = \tau_{j+1} + 1, \ldots, k$. Hence the result follows. $\square$

4.4 The Deprioritised Algorithm

We finish Chapter 4 by explicitly defining the deprioritised algorithm based on a given deprioritisable algorithm. Recall that the deprioritised algorithm uses the same operations and random variables as the deprioritisable algorithm. Pseudo-code for the deprioritised algorithm is given in Algorithm 5. We do not expect the deprioritised algorithm to be implemented in practice, instead the deprioritisable algorithm would be used.

As input, the deprioritised algorithm takes a sequence of positive real numbers $\epsilon_1, \ldots, \epsilon_K$. These numbers are the (scaled) lengths of the preprocessing subphases. The number of phases, and their types, are determined by Definition 4.3.1, which relies only on the definition of the deprioritisable algorithm. The domains $D(\delta)$ and $V^{(\tau)}(\delta)$ and the type distribution also depend solely on the deprioritisable algorithm.

During the main subphase of the deprioritised algorithm, the types of the operations performed are chosen randomly. So there is a chance that an operation of the selected type is not able to be performed. In such a case, we say that the algorithm has failed and we stop the execution of the algorithm. The analysis of the deprioritised algorithm, presented in the next chapter, shows that a.a.s. the deprioritised algorithm does not fail.

Algorithm 5 The deprioritised algorithm

```
Set $G := G_0$;
for $j = 1, \dots, K$ do
  # The preprocessing subphase of phase $j$
  for $v = 1, \dots, \lfloor \epsilon_j n \rfloor$ do
    if $(Y_0(G)/n, \dots, Y_m(G)/n) \in D(0)$ then
      Perform an operation of type 0;
    else
      return fail;
    end if
  end for
  # The main subphase of phase $j$
  while $(Y_0(G)/n, \dots, Y_m(G)/n) \in V^{(\tau_j)}(0)$ do
    Calculate the type distribution $p_r^{(\tau_j)}(Y_0(G), \dots, Y_m(G))$ for $r = 0, \dots, k$;
    Select a type $r$ randomly using the type distribution;
    if $Y_r(G) = 0$ then
      return fail;
    else
      Perform an operation of type $r$;
    end if
  end while
end for
```

In this chapter we have introduced deprioritisable algorithms and described the deprioritised algorithm based on a given deprioritisable algorithm. We have also defined functions $\hat{y}_i$ which we believe approximate the scaled random variables of the deprioritised algorithm. In the next chapter we prove that, when certain conditions are satisfied, the functions $\hat{y}_i$ (given in Definition 4.3.1) a.a.s. approximate the scaled random variables of the deprioritised algorithm. We may then determine properties of the deprioritised algorithm from the functions $\hat{y}_i$.

Chapter 5

Analysing Deprioritised Algorithms

In Chapter 4 we defined the deprioritised algorithm based on a given deprioritisable algorithm. In this chapter we analyse the asymptotic behaviour of the deprioritised algorithm. So we only consider the random process defined by the deprioritised algorithm. Recall the functions $\hat{y}_i$, defined in Chapter 4, which correspond to the (scaled) random variables of the deprioritised algorithm. (Throughout this chapter, we use the definitions and notation introduced in Chapter 4). The major result of this thesis, presented as Theorem 5.4.1, shows that under certain conditions, a.a.s. the functions $\hat{y}_i$ approximate closely the scaled random variables of the deprioritised algorithm. In the next chapter, we apply Theorem 5.4.1 to two algorithms for random d-in d-out digraphs.

A deprioritised algorithm is analysed phase by phase. Each phase is analysed using Theorem 5.1.1, which is similar to theorems of Wormald [61, Theorem 3] [60, Theorem 5.1] and is introduced in Section 5.1. In Section 5.2 we describe how Theorem 5.1.1 is applied for a particular phase. Next we motivate and state the conditions under which the functions $\hat{y}_i$ approximate the scaled random variables of the deprioritised algorithm for a given phase; in particular, Section 5.3 describes the hypotheses to Theorem 5.4.1, the main theorem of this thesis. The statement and proof of Theorem 5.4.1 is given in Section 5.4. Finally we consider how to satisfy the hypotheses of Theorem 5.4.1 and determine a more convenient set of hypotheses, which are useful for many algorithms.

5.1 A Differential Equations Theorem

Each phase of a deprioritised algorithm is analysed with Theorem 5.1.1, given below. This theorem is very similar to two theorems by Wormald [61, Theorem 3], [60, Theorem 5.1]. Using Theorem 5.1.1 we will show that, when certain conditions are satisfied, a.a.s. the scaled random variables $Y_i(t)/n$ are approximated by the solutions to a certain system of differential equations.

Theorem 5.1.1 concerns a discrete time Markov process (indexed by a variable n, although not always explicitly) and random variables $Z_1, \ldots, Z_a$ defined on this process. There are three main hypotheses to Theorem 5.1.1. Each hypothesis refers to a domain W which contains some subset of the possible values of the scaled random variables. The first hypothesis ensures that a.a.s. the process starts in W. The Boundedness Hypothesis, which is the second hypothesis, ensures that the change in the random variables during one step of the process is not too large. The third hypothesis, the Trend Hypothesis, ensures that (asymptotically) the behaviour of the random variables is described by the random variables alone.

Before stating the theorem we need a few definitions. For a given domain $W \subseteq \mathbb{R}^{a+1}$, we define the *stopping time* T_W for the random variables $Z_1, \ldots, Z_a$ to be the minimum t such that

$$(t/n, Z_1(t)/n, \ldots, Z_a(t)/n) \notin W.$$

A function $f : \mathbb{R}^b \to \mathbb{R}$ is *Lipschitz on* W (for $W \subseteq \mathbb{R}^b$) with Lipschitz constant L if, for a positive constant L, for all $\mathbf{x}$ and $\mathbf{y}$ in W we have

$$|f(\mathbf{x}) - f(\mathbf{y})| \leq L \max_{1 \leq i \leq b} |x_i - y_i|.$$

Note that the function $\|\cdot\|$ defined by $\|\mathbf{x}\| = \max_{1\le i\le b} |x_i|$ is the ℓ^∞ norm. Finally, a sequence of functions f_n *uniformly converges* to a function f on an interval I if, for every $\epsilon > 0$, there exists an N such that

$$|f(x) - f_n(x)| < \epsilon$$

for all $x \in I$ and all $n > N$. We are now ready to state Theorem 5.1.1.

Theorem 5.1.1. *Let a be a fixed positive integer. For $1 \le \ell \le a$, let Z_ℓ be a random variable defined on a discrete time Markov process $\{G_t\}_{t\ge 0}$. Assume that $W \subset \mathbb{R}^{a+1}$ is open and bounded such that*

(i) for some closed subset U of W, asymptotically almost surely

$$(0, Z_1(0)/n, \ldots, Z_a(0)/n) \in U,$$

(ii) (Boundedness Hypothesis) for some constant β we have

$$\max_{1\le\ell\le a} |Z_\ell(t+1) - Z_\ell(t)| \le \beta$$

for $t \ge 0$, and

(ii) (Trend Hypothesis) for some functions $F_\ell : \mathbb{R}^{a+1} \to \mathbb{R}$, which are Lipschitz on W for all $1 \le \ell \le a$, and for some $\lambda = \lambda(n) = o(1)$, we have

$$|\mathbb{E}(Z_\ell(t+1) - Z_\ell(t) \,|\, G_0, \ldots, G_t) - F_\ell(t/n, Z_1(t)/n, \ldots, Z_a(t)/n)| \le \lambda$$

for $t < T_W$ and $1 \le \ell \le a$.

Then the following are true:

(a) For $(0, \hat{z}_1, \ldots, \hat{z}_a) \in W$ the system of differential equations

$$\frac{dz_\ell}{dx} = F_\ell(x, z_1, \ldots, z_a) \text{ for } \ell = 1, \ldots, a \qquad (5.1.1)$$

has a unique solution in W for $z_\ell : \mathbb{R} \to \mathbb{R}$ such that $z_\ell(0) = \hat{z}_\ell$ for $1 \le \ell \le a$ and which extends to points arbitrarily close to the boundary of W.

(b) Asymptotically almost surely, for $\ell = 1, \ldots, a$, we have

$$Z_\ell(t) = nz_\ell(t/n) + o(n)$$

uniformly for $0 \leq t \leq \sigma n$, where $(x, z_1(x), \ldots, z_a(x))$ is the solution to (5.1.1) *with initial conditions $(0, Z_1(0)/n, \ldots, Z_a(0)/n)$ and $\sigma = \sigma(n)$ is the supremum of those x to which the solution can be extended before being within ℓ^∞-distance $C\mu$ of the boundary of W, for some constant $C > 0$ and for some $\mu > \lambda$ with $\mu = o(1)$.*

5.2 Applying the Differential Equations Theorem

Theorem 5.1.1 is used to analyse each phase of the deprioritised algorithm. Recall that each phase is split into a preprocessing subphase and a main subphase; Theorem 5.1.1 is applied to both subphases. In this section, we introduce the functions obtained by applying Theorem 5.1.1 and determine some properties of these functions.

Consider phase j of the deprioritised algorithm. Set the length of the preprocessing subphase to be ϵ_j for some $\epsilon_j > 0$. Then $t_{j-1} = \lfloor nx_{j-1} \rfloor$ is the end of phase $j-1$ and the start of phase j, and $t'_{j-1} = t_{j-1} + \lfloor n\epsilon_j \rfloor$ is the end of the preprocessing subphase of phase j and the start of the main subphase. Later we take $\epsilon_j = \epsilon_j(n) = o(1)$ and show that the effect of the preprocessing subphases on the behaviour of the random variables is negligible.

For the preprocessing subphase, we apply Theorem 5.1.1 to the random variables $Z_i(t) = Y_i(t_{j-1} + t)$ with the functions $F_i = f_{i,0}$ on the domain

$$W_{\delta,\epsilon_j} = D(\delta) \cap \{(x, y_0, \ldots, y_m) \, : \, -\delta < x < 2\epsilon_j\}.$$

Recall that during a preprocessing subphase, only type 0 operations are performed. The Boundedness and Trend Hypotheses are satisfied by the definition of deprioritisable algorithms (see Definition 4.1.1). Hypothesis (i) is satisfied whenever we have

$$(x_{j-1}, \hat{y}_0(x_{j-1}), \ldots, \hat{y}_m(x_{j-1})) \in D(\delta) \tag{5.2.1}$$

and

$$\begin{aligned} \big|(x_{j-1}, \hat{y}_0(x_{j-1}), \ldots, \hat{y}_m(x_{j-1})) \\ - (x_{j-1}, Y_0(t_{j-1})/n, \ldots, Y_m(t_{j-1})/n)\big| = o(1). \end{aligned} \tag{5.2.2}$$

We will only analyse phase j when the above two conditions hold. Note that, for $j > 1$, equation (5.2.1) holds whenever τ_j is defined according to Definition 4.3.1 (see Corollary 4.3.2).

Now define $(x, z_0^{(\mathrm{p})}(x), \ldots, z_m^{(\mathrm{p})}(x))$ to be the solution to the system of differential equations

$$\frac{dy_i}{dx} = f_{i,0}$$

with initial conditions $(0, Y_0(t_{j-1})/n, \ldots, Y_m(t_{j-1})/n)$ on W_{δ,ϵ_j}. Applying Theorem 5.1.1 we obtain the following results concerning the functions $z_i^{(\mathrm{p})}$ $(i = 0, \ldots, m)$.

Lemma 5.2.1. *If* (5.2.1) *and* (5.2.2) *hold, then for all ϵ_j sufficiently small, a.a.s. we have*

(i) $Y_i(t'_{j-1}) = nz_i^{(\mathrm{p})}(\epsilon_j) + o(n)$ *for* $i = 0, \ldots, m$, *and*

(ii) *for* $x \in [0, \epsilon_j]$, *the point* $(x, z_0^{(\mathrm{p})}(x), \ldots, z_m^{(\mathrm{p})}(x))$ *is bounded away from the boundary of* W_{δ,ϵ_j}.

Proof. By applying Theorem 5.1.1 as described above, we conclude that for $i = 0, \ldots, m$, a.a.s. we have

$$Y_i(t_{j-1} + t) = nz_i^{(\mathrm{p})}(t/n) + o(n) \tag{5.2.3}$$

for $0 \le t \le \sigma n$, where σ is the supremum of those x to which $(x, z_0^{(\mathrm{p})}(x), \ldots, z_m^{(\mathrm{p})}(x))$ can be extended before being within some distance $d(n) = o(1)$ of the boundary of W_{δ,ϵ_j}.

We now show that $\sigma \to 2\epsilon_j$ for sufficiently small ϵ_j. Consider the condition $H > \delta$ from the definition of $D(\delta)$ (and hence the definition of W_{δ,ϵ_j}). Recall that H is a polynomial such that the denominator of each $f_{i,r}$ is some power of H. So the condition $H > \delta$ ensures the functions $f_{i,r}$ are Lipschitz. By (5.2.1), (5.2.2), and the Lipschitz property of H we have

$$H(t_{j-1}/n, Y_0(t_{j-1})/n, \ldots, Y_m(t_{j-1})/n) > \delta$$

for sufficiently large n. Since each operation changes the random variables by only a constant, and as H is Lipschitz, we have

$$\begin{aligned} &H((t_{j-1}+t)/n, Y_0(t_{j-1}+t)/n, \ldots, Y_m(t_{j-1}+t)/n) = \\ &\qquad\qquad H(t_{j-1}/n, Y_0(t_{j-1})/n, \ldots, Y_m(t_{j-1})/n) + O(t/n) \end{aligned}$$

for $t = 0, \ldots, \lfloor 2n\epsilon_j \rfloor$. Thus, for all ϵ_j sufficiently small and for n sufficiently large, we have

$$H((t_{j-1}+t)/n, Y_0(t_{j-1}+t)/n, \ldots, Y_m(t_{j-1}+t)/n)$$

bounded above δ for $t = 0, \ldots, \lfloor 2n\epsilon_j \rfloor$. Therefore, from (5.2.3), a.a.s.

$$H(x, z_0^{(\mathrm{p})}(x), \ldots, z_m^{(\mathrm{p})}(x))$$

is bounded above δ for $x \in [0, \epsilon_j]$. Similar reasoning applies for the other conditions defining W_{δ,ϵ_j} except $x < 2\epsilon_j$. Therefore $\sigma \to 2\epsilon_j$ and the result follows. □

To analyse the preprocessing subphase of phase j, we also consider the solution $(x, y_0^{(\mathrm{p})}(x), \ldots, y_m^{(\mathrm{p})}(x))$ to the system of differential equations $dy_i/dx = f_{i,0}$, with initial conditions $(x_{j-1}, \hat{y}_0(x_{j-1}), \ldots, \hat{y}_m(x_{j-1}))$. The functions $y_i^{(\mathrm{p})}$ are useful because their initial conditions are deterministic. Using the following standard lemma [61, Lemma 1], we can show that when (5.2.2) holds, the solutions $z_i^{(\mathrm{p})}$ and $y_i^{(\mathrm{p})}$ are very similar.

Lemma 5.2.2 ([61, Lemma 1]). *Let W be some bounded and open set. Suppose that $(x, \mathbf{y}_n(x))$ and $(x, \mathbf{z}_n(x))$ satisfy the same differential equations on W with initial conditions $(0, \mathbf{y}_n(0))$ and $(0, \mathbf{z}_n(0))$ respectively. If the differential equations are Lipschitz on W and $|\mathbf{y}_n(0) - \mathbf{z}_n(0)| \to 0$ as $n \to \infty$, then $|\mathbf{y}_n(x) - \mathbf{z}_n(x)| \to 0$ uniformly for $x \in [0, x_n^\star)$ where $x_n^\star$ is the infimum of those $x > 0$ for which $(x, \mathbf{y}_n(x)) \notin W$ or $(x, \mathbf{z}_n(x)) \notin W$.*

In particular, for $z_i^{(\mathrm{p})}$ and $y_i^{(\mathrm{p})}$ we have the following result.

Lemma 5.2.3. *If* (5.2.1) *and* (5.2.2)*, then for all ϵ_j sufficiently small, a.a.s. we have*

$$\left|(z_0^{(\mathrm{p})}(\epsilon_j), \dots, z_m^{(\mathrm{p})}(\epsilon_j)) - (y_0^{(\mathrm{p})}(x'_{j-1}), \dots, y_m^{(\mathrm{p})}(x'_{j-1}))\right| = o(1).$$

Proof. Define $w_i(x) = y_i^{(\mathrm{p})}(x_{j-1} + x)$. By (5.2.2), we may apply Lemma 5.2.2 to $(x, z_0^{(\mathrm{p})}(x), \dots, z_m^{(\mathrm{p})}(x))$ and $(x, w_0(x), \dots, w_m(x))$ on the domain W_{δ,ϵ_j}. Assume that

$$(x^\star, w_0(x^\star), \dots, w_m(x^\star)) \notin W_{\delta,\epsilon_j}$$

for some positive $x^\star \le \epsilon_j$. Then by Lemma 5.2.2, the solution $(x, z_0^{(\mathrm{p})}(x), \dots, z_m^{(\mathrm{p})}(x))$ approaches arbitrarily close to the boundary of W_{δ,ϵ_j} at $x = x^\star$. This contradicts part (ii) of Lemma 5.2.1; hence the result follows. □

We also apply Theorem 5.1.1 to the main subphase of phase j. So we define $(x, z_0^{(j)}(x), \dots, z_m^{(j)}(x))$ to be the solution to the system of differential equations (4.3.1) (with $\tau = \tau_j$) for the initial conditions

$$(0, Y_i(t'_{j-1})/n, \dots, Y_m(t'_{j-1})/n).$$

Theorem 5.1.1 is used in the proof of Theorem 5.4.1 to show that, under certain conditions, the functions $z_i^{(j)}$ approximate the scaled random variables during the main subphase of phase j. Recall the functions $y_i^{(j)}$ (from Definition 4.3.1) which are used to define $\hat{y}_i$ on the interval corresponding to phase j. We will relate the functions $z_i^{(j)}$ and $y_i^{(j)}$ using Lemma 5.2.2 as in the proof of Lemma 5.2.3. Thus we show that, under conditions given in the next section, the functions $\hat{y}_i$ approximate the scaled random variables for the deprioritised algorithm during phase j.

5.3 The Hypotheses

Theorem 5.4.1 allows us to analyse a deprioritised algorithm based on a given deprioritisable algorithm. In this section, we determine hypotheses that allow phase j to be analysed. So Theorem 5.4.1 requires that these hypotheses be satisfied for each phase.

Each hypothesis for phase j concerns one or more of the functions defining the domain $V^{(\tau_j)}(\delta)$. These are the functions in the set

$$\begin{aligned}\mathcal{B}^{(\tau_j)}(\delta) = \{&y_0 - \delta,\ x + \delta,\ C - x,\ H - \delta,\ y_{\tau_j},\\ &(-1)^{|\mathcal{M}^{(\tau_j)}|+1} \det \mathcal{C}^{(\tau_j)} - \delta,\\ &2M - y_i \quad (i = 0, \ldots, m),\\ &y_i + 2M \quad (i = 0, \ldots, m),\\ &q_b^{(\tau_j)} \quad (b = 1, \ldots, |\mathcal{M}^{(\tau_j)}|),\\ &E_b^{(\tau_j)} \quad (b = 2, \ldots, |\mathcal{M}^{(\tau_j)}|)\}\end{aligned} \tag{5.3.1}$$

where $q_b^{(\tau_j)}$ and $E_b^{(\tau_j)}$ are defined by (4.3.2) and (4.3.3) respectively. Notice that

$$h(x, y_0, \ldots, y_m) > 0 \text{ for all } h \in \mathcal{B}^{(\tau_j)}(\delta) \Longrightarrow (x, y_0, \ldots, y_m) \in V^{(\tau_j)}(\delta). \tag{5.3.2}$$

Moreover a point $(x, y_0, \ldots, y_m)$ satisfying $h(x, y_0, \ldots, y_m) \geq 0$ for all $h \in \mathcal{B}^{(\tau_j)}(\delta)$ lies in the closure of $V^{(\tau_j)}(\delta)$. From the definition of $\hat{y}_i$ (Definition 4.3.1), we only analyse the deprioritised algorithm while the functions

$$\begin{aligned}&y_0 - \delta,\ x + \delta,\ C - x,\ H - \delta,\\ &\qquad 2M - y_i \quad (i = 0, \ldots, m),\ y_i + 2M \quad (i = 0, \ldots, m)\end{aligned}$$

are strictly positive. The remaining functions of $\mathcal{B}^{(\tau_j)}(\delta)$ may be zero at some point of the analysis. Thus it is also useful to define the set of functions

$$\begin{aligned}\widehat{\mathcal{B}}^{(\tau_j)}(\delta) = \{&(-1)^{|\mathcal{M}^{(\tau_j)}|+1} \det \mathcal{C}^{(\tau_j)} - \delta,\ y_{\tau_j},\\ &q_b^{(\tau_j)} \quad (b = 1, \ldots, |\mathcal{M}^{(\tau_j)}|),\\ &E_b^{(\tau_j)} \quad (b = 2, \ldots, |\mathcal{M}^{(\tau_j)}|)\}.\end{aligned}$$

To prove Theorem 5.4.1, we require that phases have non-zero length. That is, we require that for some $c_1 > x_{j-1}$ we have

$$(x, y_0^{(j)}(x), \ldots, y_m^{(j)}(x)) \in V^{(\tau_j)}(\delta)$$

for $x \in [x_{j-1}, c_1]$. By (5.3.2), it is sufficient to show that, for all $h \in \mathcal{B}^{(\tau_j)}(\delta)$, we have

$$h(x, y_0^{(j)}(x), \ldots, y_m^{(j)}(x)) > 0$$

for $x \in [x_{j-1}, c_1]$. Similarly, we need to show that for some $c_2 > 0$, we have

$$(x, z_0^{(j)}(x), \ldots, z_m^{(j)}(x)) \in V^{(\tau_j)}(\delta)$$

for $x \in [0, c_2]$. So we also consider the functions $h(x, z_0^{(j)}(x), \ldots, z_m^{(j)}(x))$ for $x \geq 0$. Such considerations lead us to the hypotheses of Theorem 5.4.1.

5.3.1 Remaining in $V^{(\tau)}(\delta)$

We consider a general setting. Assume that $y_0(x), \ldots, y_m(x)$ satisfy the differential equations $dy_i/dx = g_i$ $(i = 0, \ldots, m)$ for some functions g_i of $x, y_0, \ldots, y_m$. For a function h of $x, y_0, \ldots, y_m$, the derivative of h with respect to x is given by

$$\sum_{i=0}^{m} \frac{\partial h}{\partial y_i} \cdot g_i.$$

There are two important cases: when $dy_i/dx = f_{i,0}$, which corresponds to a preprocessing subphase, and when dy_i/dx is given by (4.3.1), which corresponds to a main subphase and the functions $\hat{y}_i$. For the first case we denote the derivative using the differential operator $\Delta^{(\mathrm{P})}$ defined by

$$\Delta^{(\mathrm{P})} h = \sum_{i=0}^{m} \frac{\partial h}{\partial y_i} \cdot f_{i,0}.$$

In the second case we denote the derivative using the differential operator $\Delta^{(\tau)}$ defined by

$$\Delta^{(\tau)} h = \sum_{i=0}^{m} \left[\frac{\partial h}{\partial y_i} \cdot \sum_{r=0}^{k} p_r^{(\tau)} f_{i,r} \right],$$

where $p_r^{(\tau)}$ are defined using Definition 4.2.1 with irreducible type sets. For arbitrary differential equations we denote the corresponding differential operator by Δ_g.

We are interested in showing that some of the functions in $\mathcal{B}^{(\tau)}(\delta)$ are increasing, so we make the following definition.

Definition 5.3.1 (Positive Growth). A function h of $(x, y_0, \ldots, y_m)$ has *positive growth (of order s) at $(x^\star, y_0^\star, \ldots, y_m^\star)$ (with respect to the differential equations $dy_i/dx = g_i$)* if, for some non-negative integer s, we have

(i) for $r = 0, \ldots, s$, the function $(\Delta_g)^r h$ exists and is continuous in $(x, y_0, \ldots, y_m)$ on an open neighbourhood containing $(x^\star, y_0^\star, \ldots, y_m^\star)$,

(ii) $(\Delta_g)^\alpha h(x^\star, y_0^\star, \ldots, y_m^\star) = 0$ for $\alpha = 0, \ldots, s-1$, and

(iii) $(\Delta_g)^s h(x^\star, y_0^\star, \ldots, y_m^\star) > 0$.

We also say that a function has *negative growth* if the inequality in (iii) is reversed. For the particular cases we are interested in, that is, when $\Delta_g = \Delta^{(\mathrm{P})}$ or $\Delta_g = \Delta^{(\tau)}$, we prefer to say 'during a preprocessing subphase' and 'during a phase of type τ' respectively, instead of 'with respect to the differential equations $dy_i/dx = g_i$'. We will only use Definition 5.3.1 with

$$(x^\star, y_0^\star, \ldots, y_m^\star) = (x_j, \hat{y}_0(x_j), \ldots, \hat{y}_m(x_j))$$

for some j. So, instead of 'at the point $(x_j, \hat{y}_0(x_j), \ldots, \hat{y}_m(x_j))$', we say 'at the point x_j' . Finally we note that the functions of $\mathcal{B}^{(\tau)}(\delta)$ are all rational functions in $x, y_0, \ldots, y_m$. As we consider these functions on domains that exclude their poles, condition (i) is always satisfied.

For a function of positive or negative growth we may obtain a lower bound on the value of the function over some interval. The next lemma determines this lower bound, which is negative for a function with negative growth. In particular, the next lemma shows that a function with positive growth is positive on some interval.

Lemma 5.3.2. *Let $h(x, y_0, \ldots, y_m)$ have positive or negative growth of order s at $(x^\star, y_0^\star, \ldots, y_m^\star)$ with respect to the differential equations $dy_i/dx = g_i$. If each g_i is Lipschitz on some domain open domain U containing $(x^\star, y_0^\star, \ldots, y_m^\star)$ then define*

$$h(x) = h(x, y_0(x), \ldots, y_m(x))$$

where $y_0(x), \ldots, y_m(x)$ are the solutions to the differential equations $dy_i/dx = g_i$ with initial conditions $(x^\star, y_0^\star, \ldots, y_m^\star)$. For any strict lower bound L on $(\Delta_g)^s h(x^\star)$ there exists a constant $C_L > 0$ such that

$$h(x) > \frac{L(x - x^\star)^s}{s!}$$

for $x \in (x^\star, x^\star + C_L]$.

Proof. The functions $y_i(x)$ (for $i = 0, \ldots, m$) are solutions to differential equations satisfying a Lipschitz property, so they are continuous on some open neighbourhood of $(x^\star, y_0^\star, \ldots, y_m^\star)$. Thus, for $\alpha = 0, \ldots, s$, the function $(\Delta_g)^\alpha h(x)$ is also continuous on some open neighbourhood of $x^\star$. Thus there exists a $C_L > 0$ such that

$$(\Delta_g)^s h(x) > L$$

for $x \in [x^\star, x^\star + C_L]$. Then, using the Fundamental Theorem of Calculus, we have

$$(\Delta_g)^{s-1} h(x) = (\Delta_g)^{s-1} h(x) - (\Delta_g)^{s-1} h(x^\star) = \int_{x^\star}^{x} (\Delta_g)^s h(u) du > \int_{x^\star}^{x} L du$$

for $x \in (x^\star, x^\star + C_L]$. The required result follows by induction. □

If each function of $\mathcal{B}^{(\tau_j)}(\delta)$ has positive growth at x_{j-1}, then by Lemma 5.3.2, condition (5.3.2) is satisfied on a non-empty interval. Thus the functions $y_i^{(j)}$ remain in $V^{(\tau_j)}(\delta)$ for a non-empty interval; this interval does not necessarily contain the initial point x_{j-1}. Note that from the definition of $\hat{y}_i$ (see Definition 4.3.1), only functions in the set $\widehat{\mathcal{B}}^{(\tau_j)}(\delta)$ may have positive growth of order s with $s \geq 1$. Next we consider how we might apply similar reasoning to the functions $z_i^{(j)}$.

5.3.2 After the preprocessing subphase

To prove Theorem 5.4.1 we need to show that, for some $C > 0$, we have

$$(x, z_0^{(j)}(x), \dots, z_m^{(j)}(x)) \in V^{(\tau_j)}(\delta) \text{ for } x \in [0, C]\,. \tag{5.3.3}$$

By analysing the preprocessing subphase, we are able to show that

$$(0, z_0^{(j)}(0), \dots, z_m^{(j)}(0)) = (0, Y_0(t'_{j-1})/n, \dots, Y_m(t'_{j-1})/n) \in V^{(\tau_j)}(\delta). \tag{5.3.4}$$

In this section we determine conditions that, together with (5.3.4), imply (5.3.3). We use an approach similar to that of the previous section.

Assume that $h \in \mathcal{B}^{(\tau_j)}(\delta)$ has positive growth of order s at x_{j-1} during a phase of type τ_j. Since each operation changes the random variables by $O(1)$ (see Definition 4.1.1) and $(\Delta^{(\tau_j)})^\alpha h$ is Lipschitz for $\alpha = 0, \dots, s$ (see Section 4.3.1), we have

$$\begin{aligned}(\Delta^{(\tau_j)})^\alpha h(x'_{j-1}, z_0^{(j)}(x'_{j-1}), \dots, z_m^{(j)}(x'_{j-1}))&\\ = (\Delta^{(\tau_j)})^\alpha h(x_{j-1}, Y_0(t_{j-1})/n, \dots, Y_m(t_{j-1})/n) &+ O(\epsilon_j)\end{aligned}$$

for $\alpha = 0, \dots, s$. So, when (5.2.2) holds and taking ϵ_j tending to zero sufficiently slowly, we have

$$\begin{aligned}(\Delta^{(\tau_j)})^\alpha h(x'_{j-1}, z_0^{(j)}(x'_{j-1}), \dots, z_m^{(j)}(x'_{j-1}))&\\ = (\Delta^{(\tau_j)})^\alpha h(x_{j-1}, \hat{y}_0(x_{j-1}), \dots, \hat{y}_m(x_{j-1})) &+ O(\epsilon_j)\end{aligned}$$

for $\alpha = 0, \dots, s$. Therefore

$$(\Delta^{(\tau_j)})^\alpha h(x'_{j-1}, z_0^{(j)}(x'_{j-1}), \dots, z_m^{(j)}(x'_{j-1})) = O(\epsilon_j)$$

for $\alpha = 0, \dots, s-1$, and

$$(\Delta^{(\tau_j)})^s h(x'_{j-1}, z_0^{(j)}(x'_{j-1}), \dots, z_m^{(j)}(x'_{j-1})) > L + O(\epsilon_j)$$

where L is a lower bound on $(\Delta^{(\tau_j)})^s(x_{j-1}, \hat{y}_0(x_{j-1}), \dots, \hat{y}_m(x_{j-1}))$.

To obtain a result similar to Lemma 5.3.2 we require explicit lower bounds on

$$(\Delta^{(\tau_j)})^\alpha h(x'_{j-1}, z_0^{(j)}(x'_{j-1}), \ldots, z_m^{(j)}(x'_{j-1}))$$

for $\alpha = 0, \ldots, s$. So we make the following definition. For any $h \in \mathcal{B}^{(\tau_j)}(\delta)$ satisfying

(i) for some non-negative integer s, the function h has positive growth of order s at x_{j-1} during a phase of type τ_j, and

(ii) for $\alpha = 0, \ldots, s-1$, for some non-negative integer s_α, the function $(\Delta^{(\tau_j)})^\alpha h$ has positive or negative growth of order s_α at x_{j-1} during a preprocessing subphase,

we let B_α (for $\alpha = 0, \ldots, s-1$) be a strict lower bound for

$$(\Delta^{(\mathrm{P})})^{s_\alpha}(\Delta^{(\tau_j)})^\alpha h(x_{j-1}, \hat{y}_0(x_{j-1}), \ldots, \hat{y}_0(x_{j-1}))$$

and let $L > 0$ be a strict lower bound for

$$(\Delta^{(\tau_j)})^s h(x_{j-1}, \hat{y}_0(x_{j-1}), \ldots, \hat{y}_0(x_{j-1})),$$

and define

$$\mathcal{L}_h(x, \epsilon_j) = \frac{Lx^s}{s!} + \sum_{\alpha=0}^{s-1} \frac{B_\alpha \epsilon_j^{s_\alpha} x^\alpha}{s_\alpha! \alpha!}.$$

Note that the function $\mathcal{L}_h(x, \epsilon_j)$ is only defined when (i) and (ii) (above) are satisfied for h. The next lemma shows that, under certain conditions, the function $\mathcal{L}_h(x, \epsilon_j)$ is a lower bound for $h(x, z_0^{(j)}(x), \ldots, z_m^{(j)}(x))$.

Lemma 5.3.3. *If* (5.2.1) *and* (5.2.2) *hold and the function $\mathcal{L}_h(x, \epsilon_j)$ is defined for $h \in \mathcal{B}^{(\tau_j)}(\delta)$, then for some constant $C_h > 0$ and for all ϵ_j sufficiently small, a.a.s. we have*

$$h(x, z_0^{(j)}(x), \ldots, z_m^{(j)}(x)) > \mathcal{L}_h(x - x'_{j-1}, \epsilon_j)$$

for $x \in \left(x'_{j-1}, x'_{j-1} + C_h\right]$.

Proof. For $\alpha = 0, \dots, s-1$, let B'_α be a strict lower bound on

$$(\Delta^{(\mathrm{P})})^{s_\alpha}(\Delta^{(\tau_j)})^\alpha h(x_{j-1}, \hat{y}_0(x_{j-1}), \dots, \hat{y}_0(x_{j-1}))$$

with $B'_\alpha > B_\alpha$. Then, since $(\Delta^{(\tau_j)})^\alpha h$ has positive or negative growth of order s_α at x_{j-1} during a preprocessing subphase, by Lemma 5.3.2, we have

$$(\Delta^{(\tau_j)})^\alpha h(x, y_0^{(\mathrm{p})}(x), \dots, y_m^{(\mathrm{p})}(x)) > B'_\alpha \frac{(x - x_{j-1})^{s_\alpha}}{s_\alpha!}$$

for $x \in (x_{j-1}, x_{j-1} + C_\alpha]$ for some constant $C_h > 0$.

Then using Lemma 5.2.3, Lemma 5.2.1, and the Lipschitz property of $(\Delta^{(\tau_j)})^\alpha h$, a.a.s. we have

$$(\Delta^{(\tau_j)})^\alpha h(0, z_0^{(j)}(0), \dots, z_m^{(j)}(0)) > B'_\alpha \frac{(\epsilon_j)^{s_\alpha}}{s_\alpha!} + o(1).$$

By replacing B'_α with B_α, we may drop the $o(1)$ term. The result follows as in the proof of Lemma 5.3.2. □

So, by showing that $\mathcal{L}_h(x, \epsilon_j)$ is positive for all sufficiently small and positive x and ϵ_j, we may conclude from the previous lemma that for some $C > 0$ we have

$$h(x, z_0^{(j)}(x), \dots, z_m^{(j)}(x)) > 0$$

for $x \in [0, C]$. Thus we can use the functions $\mathcal{L}_h(x, \epsilon_j)$ (for $h \in \mathcal{B}^{(\tau_j)}(\delta)$) to show that (5.3.3) holds. The next lemma considers four of the most useful cases of h and $\mathcal{L}_h(x, \epsilon_j)$; the proof of the lemma follows easily from the definition of the functions $\mathcal{L}_h(x, \epsilon_j)$.

Lemma 5.3.4. *Let $h \in \mathcal{B}^{(\tau_j)}(\delta)$ be such that the function $\mathcal{L}_h(x, \epsilon_j)$ is defined. Then $\mathcal{L}_h(x, \epsilon_j)$ is positive for all sufficiently small and positive x and ϵ_j when any of the following hold:*

(i) $s = 0$, *or*

(ii) $s = 1$ *and* $B_0 > 0$, *or*

(iii) $s = 2$, $s_0 = 1$, $s_1 = 1$, *and* $B_0 > 0$, *or*

(iv) $B_\alpha > 0$ *for* $\alpha = 0, \dots, s-1$.

We are now ready to give the hypotheses that allow phase j of the deprioritised algorithm to be analysed. Theorem 5.4.1 requires that these hypotheses are satisfied for each phase.

Hypotheses: to be satisfied for each phase

Let j be the phase number.

(A) There is a phase j according to Definition 4.3.1. That is, τ_j is defined and

$$(x_{j-1}, \hat{y}_0(x_{j-1}), \dots, \hat{y}_m(x_{j-1})) \in \overline{V}^{(\tau_j)}(\delta),$$

where $\overline{V}^{(\tau_j)}(\delta)$ is the closure of $V^{(\tau_j)}(\delta)$.

(B) Each function in the set $\widehat{\mathcal{B}}^{(\tau_j)}(\delta)$ has positive growth at x_{j-1} during a phase of type τ_j.

(C) For $i \in \mathcal{M}^{(\tau_j)}$, the function y_i has positive growth at x_{j-1} during a preprocessing subphase.

(D) Each

$$q \in \{(-1)^{\left|\mathcal{M}^{(\tau_j)}\right|+1} \det \mathcal{C}^{(\tau_j)} - \delta,\ q_b^{(\tau_j)} \quad (b = 1, \dots, \left|\mathcal{M}^{(\tau_j)}\right|)\}$$

has positive growth at x_{j-1} during a preprocessing subphase.

(E) For $b = 2, \dots, \left|\mathcal{M}^{(\tau_j)}\right|$, the function $E_b^{(\tau_j)}$ has positive growth at x_{j-1} during a preprocessing subphase.

(F) For each $h \in \widehat{\mathcal{B}}^{(\tau_j)}(\delta)$, the function $\mathcal{L}_h(x, \epsilon_j)$ is defined and positive for all sufficiently small and positive x and ϵ_j.

When hypotheses (A)–(F) hold for phase j, a.a.s. the scaled random variables of the deprioritised algorithm are well approximated by the functions $\hat{y}_i$ ($i = 0, \ldots, m$) on $[x_{j-1}, x_j]$. Hypothesis (A) allows the functions $\hat{y}_i$ to be defined on $[x_{j-1}, x_j]$. We show that phase j has non-zero length using (B). By (C) and (D), we can apply Theorem 5.1.1 to the main subphase of phase j: using (C) we show that an operation of the selected type can be performed and using (D) we show that the Trend Hypothesis of Theorem 5.1.1 is satisfied. Hypothesis (E), together with (C), ensures that the phase does end immediately. The last hypothesis, (F), allows us to show that the functions $z_i^{(j)}$, obtained by applying Theorem 5.1.1 to the main subphase of phase j, do no stop approximating the scaled random variables immediately; this is required to show that the functions $z_i^{(j)}$ approximate the functions $\hat{y}_i$.

Conditions on certain derivatives are also required for Wormald's analysis of deprioritised algorithms [61]. However, the extra restrictions Wormald places on the functions $f_{i,r}$ result in fewer conditions than contained in hypotheses (A)–(F). For example, Wormald considers only algorithms for which (in our notation) $\left|\mathcal{M}^{(\tau_j)}\right| = 2$. Now, although it may seem that hypotheses (A)–(F) require much effort to check, in practice checking the hypotheses is quite straightforward. First, many functions are positive at x_{j-1}, so their derivatives do not need to be considered. Also, any derivatives required can be easily calculated using automatic differentiation [38]. Indeed, the verification of hypotheses (A)–(F) can be programmed along with the numerical approximation of the functions $\hat{y}_i$.

5.4 The Theorem

We now state the major theorem of this thesis. Note that the conclusion of this theorem concerns the random process defined by the deprioritised algorithm, not the random process defined by the deprioritisable algorithm.

Theorem 5.4.1. *Let $\mathcal{P}$ be a deprioritisable algorithm and let $\hat{y}_i$ (for $i = 0, \dots, m$), K, and x_K be defined as in Definition 4.3.1. Recall that $\hat{y}_i(x_0) = \lim_{n\to\infty} Y_i(0)/n$. If*

$$(x_0, \hat{y}_0(x_0), \dots, \hat{y}_m(x_0)) \in D(\delta) \tag{5.4.1}$$

and hypotheses (A)–(F) are satisfied for $j = 1, \dots, K$, then, for the random process $\{G_t\}$ generated by the deprioritised algorithm based on $\mathcal{P}$, a.a.s. we have

$$Y_i(G_t) = n\hat{y}_i(t/n) + o(n) \quad \textit{for } i = 0, \dots, m,$$

uniformly for $t = 0, \dots, \lfloor nx_K \rfloor$.

5.4.1 The proof

We prove Theorem 5.4.1 by induction on the number of phases. The base case is $K = 0$ and holds by the definition of $(x_0, \hat{y}_0(x_0), \dots, \hat{y}_m(x_0))$ and (vii) of Definition 4.1.1. Now assume that, for some $j \le K$, the conclusion of Theorem 5.4.1 holds for $t = 0, \dots, \lfloor nx_{j-1} \rfloor$. The proof of the inductive step is done in three parts. Throughout the proof we make use of the functions $z_i^{(\mathrm{p})}$, $y_i^{(\mathrm{p})}$, and $z_i^{(j)}$ defined in Section 5.2. We start by showing that phase j has non-zero length.

Part I: Showing $x_j > x_{j-1}$.

Let $\overline{V}^{(\tau_j)}(\delta)$ be the closure of $V^{(\tau_j)}(\delta)$. By hypothesis (A) and Definition 4.3.1, we have

$$(x_{j-1}, \hat{y}_0(x_{j-1}), \dots, \hat{y}_m(x_{j-1})) \in \overline{V}^{(\tau_j)}(\delta).$$

Recall that $\hat{y}_i$ is defined in terms of $y_i^{(j)}$ where the functions $y_i^{(j)}$ ($i = 0, \dots, m$) are the solutions to the differential equations dy_i/dx given by (4.3.1) (with $\tau = \tau_j$) for the initial conditions $y_i^{(j)}(x_{j-1}) = \hat{y}_i(x_{j-1})$.

The functions in the set $\mathcal{B}^{(\tau_j)}(\delta) \backslash \widehat{\mathcal{B}}^{(\tau_j)}(\delta)$ have positive growth of order 0 at x_{j-1} by hypothesis (A) and Corollary 4.3.2 if $j > 1$, or by (5.4.1) if $j = 1$. So, together with

(B), we may apply Lemma 5.3.2 to each $h \in \mathcal{B}^{(\tau_j)}(\delta)$. Hence for each $h \in \mathcal{B}^{(\tau_j)}(\delta)$, there exists a $c_h > 0$ such that

$$h(x, y_0^{(j)}(x), \ldots, y_m^{(j)}(x)) > 0$$

for $x \in (x_{j-1}, x_{j-1} + c_h]$. Thus we have

$$x_j > x_{j-1} + \min_{h \in \mathcal{B}^{(\tau_j)}(\delta)} c_h > x_{j-1},$$

from (5.3.2) and as $\mathcal{B}^{(\tau_j)}(\delta)$ is finite. Next we consider the random variables during the preprocessing subphase of phase j.

Part II: The Preprocessing Subphase.

We apply Theorem 5.1.1 to the main subphase of phase j with the random variables $Z_i(t) = Y_i(t'_{j-1} + t)$ on the domain

$$U^{(\tau_j)}(\delta) = V^{(\tau_j)}(\delta) \cap \{(x, y_0, \ldots, y_m) \,:\, y_i > 0 \text{ for } i \in \mathcal{M}^{(\tau_j)} \backslash \{\tau_j\}\}.$$

Recall that $x'_{j-1} = x_{j-1} + \epsilon_j$ and that $t'_{j-1} = \lfloor n x'_{j-1} \rfloor$ is the end of the preprocessing subphase. To satisfy hypothesis (i) of Theorem 5.1.1, we need to show that a.a.s. the point

$$(t'_{j-1}/n, Y_0(t'_{j-1})/n, \ldots, Y_m(t'_{j-1})/n)$$

lies in $U^{(\tau_j)}(\delta)$ and is a distance of at least a constant from the boundary of $U^{(\tau_j)}(\delta)$.

Now the functions $y_i^{(\mathrm{p})}$ $(i = 0, \ldots, m)$ are the solutions to the differential equations $dy_i/dx = f_{i,0}$ with initial conditions $y_i^{(\mathrm{p})}(x_{j-1}) = \hat{y}_i(x_{j-1})$. By Corollary 4.3.2 and (A) (for $j > 1$), or by (5.4.1) (for $j = 1$), equation (5.2.1) holds. Equation (5.2.2) holds by the inductive hypothesis for $j > 1$, and by (vii) of Definition 4.1.1 for $j = 1$. So applying Lemma 5.2.3, for all sufficiently small ϵ_j, a.a.s. we have

$$y_i^{(\mathrm{p})}(x'_{j-1}) = z_i^{(\mathrm{p})}(\epsilon_j) + o(1) \tag{5.4.2}$$

for $i = 0, \ldots, m$. Thus, by Lemma 5.2.1 (ii), for all sufficiently small ϵ_j, a.a.s. the point

$$(x'_{j-1}, y_0^{(\mathrm{p})}(x'_{j-1}), \ldots, y_m^{(\mathrm{p})}(x'_{j-1}))$$

lies in $D(\delta)$ and is a distance of at least a constant from the boundary of $D(\delta)$.

As hypotheses (C), (D), and (E) hold, we may apply Lemma 5.3.2. Thus, by (5.3.2), for all ϵ_j sufficiently small, a.a.s. the point

$$(x'_{j-1}, y_0^{(\mathrm{p})}(x'_{j-1}), \dots, y_m^{(\mathrm{p})}(x'_{j-1}))$$

lies in $U^{(\tau_j)}(\delta)$ and is a distance of at least a constant from the boundary of $U^{(\tau_j)}(\delta)$. Now from (5.4.2) and Lemma 5.2.1 (i), for all ϵ_j sufficiently small, a.a.s. we have $y_i^{(\mathrm{p})}(x'_{j-1}) = Y_i(t'_{j-1})/n + o(1)$ for $i = 0, \dots, m$.

Therefore, for all ϵ_j sufficiently small, a.a.s. the point

$$(t'_{j-1}/n, Y_0(t'_{j-1})/n, \dots, Y_m(t'_{j-1})/n)$$

lies in $U^{(\tau_j)}(\delta)$ and is a distance of at least a constant from the boundary of $U^{(\tau_j)}(\delta)$. Thus hypothesis (i) of Theorem 5.1.1 is satisfied. We complete the proof of Theorem 5.4.1 by analysing the main subphase of phase j.

Part III: The Main Subphase

We analyse the main subphase of phase j by applying Theorem 5.1.1 to the random variables $Z_i(t) = Y_i(t'_{j-1}+t)$ with functions F_i given by the right hand side of (4.3.1) (with $\tau = \tau_j$) on the domain $U^{(\tau_j)}(\delta)$. Hypothesis (i) is satisfied by Part II and the Boundedness hypothesis is satisfied by part (v) of Definition 4.1.1. The definition of $U^{(\tau_j)}(\delta)$ ensures that operations of types from $\mathcal{M}^{(\tau_j)}$ can be performed and that, by Lemma 4.2.3, the functions $p_0^{(\tau_j)}, \dots, p_k^{(\tau_j)}$ define a probability distribution. Thus the Trend hypothesis follows from part (iii) of Definition 4.1.1, the definition of $U^{(\tau_j)}(\delta)$, the definition of the deprioritised algorithm, and Section 4.3.1 (for the Lipschitz conditions).

Recall that $(x, z_0^{(j)}(x), \dots, z_m^{(j)}(x))$ is the solution to the system of differential equations (4.3.1) (with $\tau = \tau_j$) for the initial conditions $z_i^{(j)}(0) = Y_i(t'_{j-1})/n$, on the domain $U^{(\tau_j)}(\delta)$. From Theorem 5.1.1, a.a.s. we have

$$Y_i(t'_{j-1} + t) = nz_i^{(j)}(t/n) + o(n) \tag{5.4.3}$$

uniformly for $0 \leq t \leq \sigma n$, where σ is the supremum of those x for which the solution $(x, z_0^{(j)}(x), \ldots, z_m^{(j)}(x))$ can be extended before being within some distance $d(n) = o(1)$ of the boundary of $U^{(\tau_j)}(\delta)$. Note that σ depends on ϵ_j.

First, by Lemma 4.3.4, for $i \in \mathcal{M}^{(\tau_j)} \backslash \{\tau_j\}$ we have

$$z_i^{(j)}(x) = z_i^{(j)}(0) = Y_i(t'_{j-1})/n > 0$$

for all x for which the solutions are defined. Therefore

$$(x, z_0^{(j)}(x), \ldots, z_m^{(j)}(x)) \in V^{(\tau_j)}(\delta) \Longrightarrow (x, z_0^{(j)}(x), \ldots, z_m^{(j)}(x)) \in U^{(\tau_j)}(\delta). \quad (5.4.4)$$

Now

$$\left|(z_0^{(j)}(0), \ldots, z_m^{(j)}(0)) - (\hat{y}_0(x_{j-1}), \ldots, \hat{y}_m(x_{j-1}))\right| = O(\epsilon_j)$$

by part (v) of Definition 4.1.1. Taking $\epsilon_j = \epsilon_j(n) = o(1)$, we apply Lemma 5.2.2 on the domain $V^{(\tau_j)}(\delta)$ and conclude that

$$\left|(z_0^{(j)}(x), \ldots, z_m^{(j)}(x)) - (\hat{y}_0(x_{j-1} + x), \ldots, \hat{y}_m(x_{j-1} + x))\right| = o(1) \quad (5.4.5)$$

uniformly for $x \in [0, \min\{x_1^\star, x_2^\star\})$, where $x_1^\star$ is the infimum of those $x > 0$ for which

$$(x, z_0^{(j)}(x), \ldots, z_m^{(j)}(x)) \notin V^{(\tau_j)}(\delta)$$

and $x_2^\star$ is the infimum of those $x > 0$ for which

$$(x, \hat{y}_0(x_{j-1} + x), \ldots, \hat{y}_m(x_{j-1} + x)) \notin V^{(\tau_j)}(\delta).$$

Note that by definition we have $\sigma < x_1^\star$ and $x_2^\star = x_j - x_{j-1}$.

Next we show that $x_j - x_{j-1}$ can be at most $o(1)$ larger than σ. By (F) and Lemma 5.3.3, for some constant $C > 0$ and for all $h \in \mathcal{B}^{(\tau_j)}(\delta)$ we have

$$h(x, z_0^{(j)}(x), \ldots, z_m^{(j)}(x)) > 0$$

for $x \in [0, C]$. Therefore a.a.s. $\sigma > C$; note that C does not depend on ϵ_j. Let

$$D = \min\{C, \min_{h \in \mathcal{B}^{(\tau_j)}(\delta)} c_h\}$$

where c_h is defined in Part I.

Now condition on the event $\sigma > D$. For all $\kappa > 0$, sufficiently small so that $x_j - x_{j-1} - \kappa > D$ (such κ exist by Part I), we want to show that $x_j - x_{j-1} - \kappa < \sigma$ for sufficiently large n. Assume that $x_j - x_{j-1} - \kappa > \sigma$ for infinitely many n. Then, as $\sigma < x_1^\star$ and $\sigma < x_j - x_{j-1} = x_2^\star$, from (5.4.5) we have

$$\left|(z_0^{(j)}(\sigma), \ldots, z_m^{(j)}(\sigma)) - (\hat{y}_0(\sigma), \ldots, \hat{y}_m(\sigma))\right| = o(1).$$

By definition, at $x = \sigma$, the solution $(x, z_0^{(j)}(x), \ldots, z_m^{(j)}(x))$ approaches the boundary of $V^{(\tau_j)}(\delta)$. Hence the distance from $(x, \hat{y}_0(x_{j-1} + x), \ldots, \hat{y}_m(x_{j-1} + x))$ to the boundary of $V^{(\tau_j)}(\delta)$ is bounded above by a function that tends to zero as n tends to infinity. This is a contradiction since $(x, \hat{y}_0(x_{j-1} + x), \ldots, \hat{y}_m(x_{j-1} + x))$ is bounded away from the boundary of $V^{(\tau_j)}(\delta)$ on $[D, x_j - x_{j-1} - \kappa]$ and $\sigma \in [D, x_j - x_{j-1} - \kappa]$. Hence $x_j - x_{j-1} - \kappa < \sigma$ for sufficiently large n. Therefore, a.a.s. there exists a sequence $\kappa(n) = o(1)$ such that $x_j - x_{j-1} - \kappa(n) < \sigma(n)$.

Now for $t = \lfloor nx_{j-1} \rfloor + 1, \ldots, \lfloor nx_j \rfloor$, let $x = t/n$ and $x' = x - x_{j-1} - \kappa(n)$. By part (v) of Definition 4.1.1 we have

$$Y_i(t) = Y_i(nx) = Y_i(t_{j-1} + nx') + o(n) = Y_i(t'_{j-1} + \lfloor n(x' - \epsilon_j) \rfloor) + o(n).$$

By (5.4.3), a.a.s. we have

$$Y_i(t'_{j-1} + \lfloor n(x' - \epsilon_j) \rfloor) + o(n) = nz_i^{(j)}(x' - \epsilon_j) + o(n).$$

Since $t \leq \lfloor nx_j \rfloor$, we have $x' < x_2^\star$ and (from above) a.a.s. $x' = x_j - x_{j-1} - \kappa(n) < \sigma$. Therefore, by (5.4.5), a.a.s. we have

$$\begin{aligned} nz_i^{(j)}(x' - \epsilon_j) + o(n) &= n\hat{z}_i(x' - \epsilon_j) + o(n) \\ &= n\hat{y}_i(x' + x_{j-1} - \epsilon_j) + o(n) \\ &= n\hat{y}_i(x - \kappa(n) - \epsilon_j) + o(n). \end{aligned}$$

Finally since each $\hat{y}_i$ is Lipschitz, a.a.s. we have

$$n\hat{y}_i(x - \kappa(n) - \epsilon_j) + o(n) = n\hat{y}_i(x) + o(n).$$

Therefore, for $i = 0, \ldots, m$, a.a.s. we have

$$Y_i(t) = n\hat{y}_i(t/n) + o(n)$$

uniformly for $t = \lfloor nx_{j-1} \rfloor + 1, \ldots, \lfloor nx_j \rfloor$, as required. This completes the proof of Theorem 5.4.1.

5.5 Changing Phase

In the applications considered in Chapter 6, we satisfy the hypotheses of Theorem 5.4.1 using values calculated numerically and the theoretical results given in this section. The theoretical results show that certain functions are zero at the change of phase. These results also allow us to specify alternative hypotheses that are easier to satisfy and are sufficient in many cases.

First we describe a common situation which motivates the work of this section. Assume that after phase $j-1$ of type τ_{j-1} there is a phase of type $\tau_j = \tau_{j-1}+1$. Difficulties arise when we try to check hypothesis (B) for phase j numerically; for example, we cannot show that $\Delta^{(\tau_j)} y_{\tau_j}$ has positive growth at x_{j-1} during a phase of type τ_j. When we solve the differential equations numerically, as we usually do, we are unable to determine x_{j-1} exactly. Instead we determine points x_1 and x_2 such that $x_1 < x_{j-1} < x_2$. At x_1 the function $\Delta^{(\tau_j)} y_{\tau_j}$ is small and negative, while at x_2 the function $\Delta^{(\tau_j)} y_{\tau_j}$ is small and positive. However, to apply Theorem 5.4.1 we must show that $\Delta^{(\tau_j)} y_{\tau_j}(x_{j-1})$ is non-negative. The results of this section show that, in fact, $\Delta^{(\tau_j)} y_{\tau_j}(x_{j-1}) = 0$. Other functions, such as $q_b^{(\tau_j)}$ for some b, behave similarly.

First we consider $\Delta^{(\tau_j)} y_{\tau_j}$ at x_{j-1} when $\tau_j > \tau_{j-1}$. Recall the definition of the Independent Types Property from Definition 4.2.7.

Lemma 5.5.1. *Assume that $j > 1$ and $\tau_j > \tau_{j-1}$. Let $\mathcal{E}$ be the clutch matrix for a $\{\tau_j, \dots, k\}$-clutch. If $\det \mathcal{E}(x_{j-1}) \neq 0$ or the Independent Types Property holds for a phase of type τ_j, then $\Delta^{(\tau_j)} y_{\tau_j}(x_{j-1}) = 0$.*

Proof. Recall that $\mathrm{ix}(r, S) = |\{w \in S : w \leq r\}|$ if $r \in S$, and $\mathrm{ix}(r, S) = 0$ if $r \notin S$. Now, by Corollary 4.3.2, we have $E_\beta^{(\tau_{j-1})}(x_{j-1}) = 0$ for $\beta = \mathrm{ix}(\tau_j, \mathcal{M}^{(\tau_{j-1})})$. Therefore

$$\det \mathcal{C}^{(\tau_{j-1})}(\{1, \dots, \mathrm{ix}(\tau_j, \mathcal{M}^{(\tau_{j-1})}) - 1\}, \{1, \dots, \mathrm{ix}(\tau_j, \mathcal{M}^{(\tau_{j-1})}) - 1\})$$

evaluated at x_{j-1} is zero. Let $\mathcal{D}$ be the clutch matrix for a $\{\tau_{j-1}, \dots, k\}$-clutch.

Then by Lemma 4.2.9, the function

$$\det \mathcal{D}(\{1,\ldots,\tau_j-\tau_{j-1}\},\{1,\ldots,\tau_j-\tau_{j-1}\})$$

evaluated at x_{j-1} is also zero.

Now by definition

$$\Delta^{(\tau_j)} y_{\tau_j} = \sum_{r\in\mathcal{M}^{(\tau_j)}} p_r^{(\tau_j)} f_{\tau_j,r}.$$

By the hypothesis to the lemma, we may apply Lemma 4.2.8 to show that

$$\Delta^{(\tau_j)} y_{\tau_j} = \sum_{r=\tau_j}^{k} p_r f_{\tau_j,r}$$

where

$$p_r = (-1)^{r-\tau_j} \frac{\det \mathcal{E}(r-\tau_j+1,1)}{\det \mathcal{E}}.$$

Therefore

$$\begin{aligned}\Delta^{(\tau_j)} y_{\tau_j} &= \frac{1}{\det\mathcal{E}} \det \begin{pmatrix} f_{\tau_j,\tau_j} & f_{\tau_j+1,\tau_j} & \cdots & f_{k,\tau_j} \\ \vdots & \vdots & \vdots & \vdots \\ f_{\tau_j,k} & f_{\tau_j+1,k} & \cdots & f_{k,k}\end{pmatrix} \\ &= \frac{\det \mathcal{D}(\{1,\ldots,\tau_j-\tau_{j-1}\},\{1,\ldots,\tau_j-\tau_{j-1}\})}{\det\mathcal{E}}.\end{aligned}$$

Hence $\Delta^{(\tau_j)} y_{\tau_j}(x_{j-1}) = 0$. □

The next lemma gives sufficient conditions for $q_b^{(\tau_j)}$ to be zero.

Lemma 5.5.2. *Let $\mathcal{M}^{(\tau_j)} = \{w_1,\ldots,w_a\}$ with $w_1 < \cdots < w_a$, and let $x \in [x_{j-1}, x_j]$. If τ is such that $0 \le \tau < \tau_j$, $\tau_j \in \mathcal{M}^{(\tau)}$, and $\hat{y}_i(x) = 0$ for $i = \tau+1,\ldots,\tau_j$, then $q_b^{(\tau_j)}(x) = 0$ whenever $w_b \in \mathcal{M}^{(\tau_j)}\backslash\mathcal{M}^{(\tau)}$.*

Proof. Recall that $\mathcal{C}^{(\tau_j)}$ is the clutch matrix for a $\mathcal{M}^{(\tau_j)}$-clutch and

$$q_b^{(\tau_j)} = (-1)^{a+b} \det \mathcal{C}^{(\tau_j)}(b,1).$$

By Lemma 4.3.5, we have $\hat{y}_i(x) = 0$ for $i = \tau_j+1,\ldots,k$; so by Lemma 4.2.6, for $w \in \mathcal{M}^{(\tau_j)}\backslash\mathcal{M}^{(\tau)}$, we have $f_{w,u}(x) = 0$ for all $u \in \mathcal{M}^{(\tau)}$.

Now fix $w_b \in \mathcal{M}^{(\tau_j)} \backslash \mathcal{M}^{(\tau)}$ and, for $s = 2, \ldots, a$, denote column $s-1$ of $\mathcal{C}^{(\tau_j)}(b,1)(x)$ by $\mathbf{c}_s$; so

$$\mathbf{c}_s = (f_{w_s,w_1}(x), \ldots, f_{w_s,w_{b-1}}(x), f_{w_s,w_{b+1}}(x), \ldots, f_{w_s,w_a}(x))^T.$$

For $w_s \in \mathcal{M}^{(\tau_j)} \backslash \mathcal{M}^{(\tau)}$ we have $s \geq 2$ as $\tau_j \in \mathcal{M}^{(\tau)}$. So let $\mathbf{C}$ be the set of columns $\mathbf{c}_s$ for all s such that $w_s \in \mathcal{M}^{(\tau_j)} \backslash \mathcal{M}^{(\tau)}$. For each $w_i \in \mathcal{M}^{(\tau)}$ we have $i \neq b$ and so every column in $\mathbf{C}$ has a zero in row i if $i < b$, and $i-1$ if $i > b$. Thus $\mathbf{C}$ is a set of $\left|\mathcal{M}^{(\tau_j)} \backslash \mathcal{M}^{(\tau)}\right|$ vectors contained in a vector space of dimension $a-1-\left|\mathcal{M}^{(\tau_j)} \cap \mathcal{M}^{(\tau)}\right|$. Now

$$\left|\mathcal{M}^{(\tau_j)} \backslash \mathcal{M}^{(\tau)}\right| + \left|\mathcal{M}^{(\tau_j)} \cap \mathcal{M}^{(\tau)}\right| = a,$$

so $\mathbf{C}$ is linearly dependent. Therefore $q_b^{(\tau_j)}(x) = 0$. □

The above lemma is most useful at the beginning of phase j when $\tau_j > \tau_{j-1}$, and at the end of phase $j-1$ when $\tau_j < \tau_{j-1}$. These cases are given in the next corollary. We may also apply Lemma 5.5.2 more generally, for example, to the first phase of an algorithm.

Corollary 5.5.3. *Let $\mathcal{M}^{(\tau_j)} = \{w_1, \ldots, w_a\}$ with $w_1 < \cdots < w_a$.*

(i) *Let $j > 1$ (so that $\mathcal{M}^{(\tau_{j-1})}$ is defined) and assume that $\tau_j > \tau_{j-1}$. Then for each $w_b \in \mathcal{M}^{(\tau_j)} \backslash \mathcal{M}^{(\tau_{j-1})}$ we have $q_b^{(\tau_j)}(x_{j-1}) = 0$.*

(ii) *Let $j < K$ (so that $\mathcal{M}^{(\tau_{j+1})}$ is defined) and assume that $\tau_{j+1} < \tau_j$. If we have $\tau_j \in \mathcal{M}^{(\tau_{j+1})}$, then $q_b^{(\tau_j)}(x_j) = 0$ for each $w_b \in \mathcal{M}^{(\tau_j)} \backslash \mathcal{M}^{(\tau_{j+1})}$.*

Proof. (i) By Lemma 4.3.5 we have

$$\hat{y}_{\tau_{j-1}}(x_{j-1}) = \cdots = \hat{y}_{\tau_j}(x_{j-1}) = 0,$$

and by Definition 4.3.1 (see also Corollary 4.3.2) we have $\tau_j \in \mathcal{M}^{(\tau_{j-1})}$. So the result follows by Lemma 5.5.2 with $\tau = \tau_{j-1}$ and $x = x_{j-1}$.

(ii) By Definition 4.3.1 (see also Corollary 4.3.2) we have

$$\hat{y}_{\tau_{j+1}+1}(x_j) = \cdots = \hat{y}_{\tau_j}(x_j) = 0.$$

So the result follows by Lemma 5.5.2 with $\tau = \tau_{j+1}$ and $x = x_j$.

□

In order to satisfy hypothesis (C), we must show that each $q_b^{(\tau_j)}$ has positive growth at x_{j-1} during a phase of type τ_j. So when $q_b^{(\tau_j)}(x_{j-1}) = 0$, we must consider the derivatives of $q_b^{(\tau_j)}$ at x_{j-1}. The next lemma shows that $(\Delta^{(\tau_j)})q_b^{(\tau_j)}(x_{j-1})$ may also be zero.

Lemma 5.5.4. *Assume that $j > 1$ and $\tau_j > \tau_{j-1}$. Let $\mathcal{M}^{(\tau_j)} = \{w_1, \ldots, w_a\}$ with $w_1 < \cdots < w_a$. If the clutch matrix for a $\{\tau_j, \ldots, k\}$-clutch has a non-zero determinant or the Independent Types Property holds for a phase of type τ_j, then $\Delta^{(\tau_j)} q_b^{(\tau_j)}(x_{j-1}) = 0$ for $w_b \in \mathcal{M}^{(\tau_j)} \backslash \mathcal{M}^{(\tau_{j-1})}$.*

Proof. Recall that

$$\Delta^{(\tau_j)} q_b^{(\tau_j)}(x_{j-1}) = \sum_{i=0}^{m} \frac{\partial q_b^{(\tau_j)}}{\partial y_i}(x_{j-1}) \frac{dy_i}{dx}(x_{j-1}; \tau_j).$$

By Corollary 4.3.2, Lemma 4.3.4, and Lemma 5.5.1 we have $dy_i/dx(x_{j-1}) = 0$ for $i = \tau_j, \ldots, k$. Thus it is enough to show that for $i \in \{0 \ldots, m\} \backslash \{\tau_j, \ldots, k\}$ we have

$$\frac{\partial q_b^{(\tau_j)}}{\partial y_i}(x_{j-1}) = (-1)^{a+b} \frac{\partial}{\partial y_i} \left(\det \mathcal{C}^{(\tau_j)}(b, 1)\right)(x_{j-1}) = 0.$$

So fix $i \notin \{\tau_j, \ldots, k\}$. Let $\mathcal{C}_\ell$ be the matrix obtained by replacing column ℓ of $\mathcal{C}^{(\tau_j)}(b, 1)$, which is

$$(f_{w_{\ell+1}, w_1}, \ldots, f_{w_{\ell+1}, w_{b-1}}, f_{w_{\ell+1}, w_{b+1}}, \ldots, f_{w_{\ell+1}, w_a})^T,$$

by

$$\left(\frac{\partial}{\partial y_i}(f_{w_{\ell+1}, w_1}), \ldots, \frac{\partial}{\partial y_i}(f_{w_{\ell+1}, w_{b-1}}), \frac{\partial}{\partial y_i}(f_{w_{\ell+1}, w_{b+1}}), \ldots, \frac{\partial}{\partial y_i}(f_{w_{\ell+1}, w_a})\right)^T.$$

Then
$$\frac{\partial}{\partial y_i}\left(\det \mathcal{C}^{(\tau_j)}(b,1)\right) = \sum_{\ell=1}^{a-1} \det \mathcal{C}_\ell.$$
We now show that $\det \mathcal{C}_\ell(x_{j-1}) = 0$ for $\ell = 1, \ldots, a-1$. Let $w \in \mathcal{M}^{(\tau_j)} \backslash \mathcal{M}^{(\tau_{j-1})}$. Then, by Lemma 4.2.6, for all $(x, y_0, \ldots, y_m) \in D(\delta)$ with $y_{\tau_j} = \ldots = y_k = 0$, we have
$$f_{w,u}(x, y_0, \ldots, y_m) = 0 \text{ for all } u \in \mathcal{M}^{(\tau_{j-1})}. \tag{5.5.1}$$
Note that by Corollary 4.3.2 we have $\hat{y}_i(x_{j-1}) = 0$ for $i = \tau_j, \ldots, k$.

Recall from Definition 4.1.1 that, for some polynomial H, each function $f_{s,r}$ can be written as $g_{s,r}/H^{\ell_s}$ for some polynomial function $g_{s,r}$ and some positive integer ℓ_s. Let $\Sigma = \{(b, a_0, \ldots, a_m) \in \mathbb{Z}^{m+2} : b, a_0, \ldots, a_m \geq 0\}$. Then
$$g_{s,r} = \sum_{(b,a_0,\ldots,a_m)\in\Sigma} C(b, a_0, \ldots, a_m) x^b y_0^{a_0} \cdots y_m^{a_m}$$
where all but a finite number of the values $C(b, a_0, \ldots, a_m)$ are zero. Consider $g_{w,u}$ with $u \in \mathcal{M}^{(\tau_{j-1})}$. By (5.5.1), for each $(b, a_0, \ldots, a_m) \in \Sigma$ with $C(b, a_0, \ldots, a_m) > 0$ we have at least one of $a_{\tau+1}, \ldots, a_k$ positive. Since $i \notin \{\tau_j, \ldots, k\}$, this property still holds after differentiating $g_{w,u}$ with respect to y_i. Therefore, for all $(x, y_0, \ldots, y_m) \in D(\delta)$ with $y_{\tau_j} = \ldots = y_k = 0$, we have
$$\frac{\partial f_{w,u}}{\partial y_i}(x, y_0, \ldots, y_m) = 0 \text{ for all } u \in \mathcal{M}^{(\tau_{j-1})}.$$

Now, as in the proof of Lemma 5.5.2 (and using Corollary 4.3.2), each matrix $\mathcal{C}_\ell$ has a set of $\left|\mathcal{M}^{(\tau_j)} \backslash \mathcal{M}^{(\tau_{j-1})}\right|$ column vectors contained in a subspace of dimension $a - 1 - \left|\mathcal{M}^{(\tau_j)} \cap \mathcal{M}^{(\tau_{j-1})}\right|$. Hence $\det \mathcal{C}_\ell = 0$ for $\ell = 1, \ldots, a-1$, and the result follows. □

The next lemma, similar to Lemma 5.5.2, considers the derivatives of the functions $\hat{y}_i$ during a preprocessing subphase.

Lemma 5.5.5. *Assume that the Independent Types Property holds for a phase of type τ_j. If $\tau \in \{0, \dots, \tau_j - 1\}$ is such that*

$$\hat{y}_{\tau+1}(x_{j-1}) = \cdots = \hat{y}_{\tau_j}(x_{j-1}) = 0,$$

then for all $i \in \mathcal{M}^{(\tau_j)} \backslash \mathcal{M}^{(\tau)}$ we have $\Delta^{(\mathrm{P})}\hat{y}_i(x_{j-1}) = 0$.

Proof. By Lemma 4.3.5 we have

$$\hat{y}_{\tau_j}(x_{j-1}) = \cdots = \hat{y}_k(x_{j-1}) = 0.$$

As $i \in \mathcal{M}^{(\tau_j)}$, we have $i \geq \tau_j > \tau \geq 0$ and so $i \neq 0$. Thus by the Independent Types Property (see Definition 4.2.7), we have $\Delta^{(\mathrm{P})}\hat{y}_i(x_{j-1}) = f_{i,0}(x_{j-1}) = 0$. □

As with Lemma 5.5.2 we usually apply Lemma 5.5.5 with $\tau = \tau_{j-1}$ when changing to a phase of higher type. Finally we consider the functions $E_b^{(\tau_j)}$ at a change of phase.

Lemma 5.5.6. *Assume that $j > 1$, so $\mathcal{M}^{(\tau_{j-1})}$ is defined. If the Independent Types Property holds for a phase of type τ_j, then*

$$E_b^{(\tau_j)}(x_{j-1}) > 0 \textit{ for } b = \max\{2, 2 + \tau_{j-1} - \tau_j\}, \dots, \left|\mathcal{M}^{(\tau_j)}\right|.$$

In particular, if $\tau_j > \tau_{j-1}$, then hypothesis (E) is satisfied for phase j.

Proof. For $\tau = \tau_{j-1}$ and $\tau = \tau_j$, let $\mathcal{C}^{(\tau)}$ be the clutch matrix for a $\mathcal{M}^{(\tau)}$-clutch. Define $\mathcal{N}^{(\tau)} = \{\tau, \dots, k\}$ and let $\mathcal{D}^{(\tau)}$ be the clutch matrix for a $\mathcal{N}^{(\tau)}$-clutch. Also define

$$S^{(\tau)}(r) = \det \mathcal{D}^{(\tau)}(\{1, \dots, r - \tau\}, \{1, \dots, r - \tau\})$$

for $r \in \mathcal{N}^{(\tau)}$, and

$$T^{(\tau)}(r) = \det \mathcal{C}^{(\tau)}(\{1, \dots, \mathrm{ix}(r, \mathcal{M}^{(\tau)}) - 1\}, \{1, \dots, \mathrm{ix}(r, \mathcal{M}^{(\tau)}) - 1\})$$

for $r \in \mathcal{M}^{(\tau)}$.

Consider $w \in \mathcal{M}^{(\tau_j)}$ with $\mathrm{ix}\big(w, \mathcal{M}^{(\tau_j)}\big) = b$ for $b \geq \max\{2, 2+\tau_{j-1}-\tau_j\}$. Recall that (from (4.3.3))

$$E_b^{(\tau_j)} = (-1)^{|\mathcal{M}^{(\tau_j)}| - \mathrm{ix}\left(w, \mathcal{M}^{(\tau_j)}\right)+1} T^{(\tau_j)}(w).$$

By Lemma 4.3.5, Lemma 4.2.9, and as $w \in \mathcal{M}^{(\tau_j)}$, we have

$$S^{(\tau_j)}(w)(x_{j-1}) = (-1)^{k-w-|\mathcal{M}^{(\tau_j)}|+\mathrm{ix}\left(w,\mathcal{M}^{(\tau_j)}\right)} T^{(\tau_j)}(w)(x_{j-1})$$

and so

$$E_b^{(\tau_j)}(x_{j-1}) = (-1)^{k-w+1} S^{(\tau_j)}(w)(x_{j-1}).$$

Now, for $\tau_j > \tau_{j-1}$ we have

$$\mathcal{D}^{(\tau_j)} = \mathcal{D}^{(\tau_{j-1})}(\{1, \ldots, \tau_j - \tau_{j-1}\}, \{2, \ldots, \tau_j - \tau_{j-1}+1\}),$$

while for $\tau_j < \tau_{j-1}$ we have

$$\mathcal{D}^{(\tau_{j-1})} = \mathcal{D}^{(\tau_j)}(\{1, \ldots, \tau_{j-1} - \tau_j\}, \{2, \ldots, \tau_{j-1} - \tau_j + 1\}).$$

If $\tau_j > \tau_{j-1}$, then $S^{(\tau_j)}(w) = S^{(\tau_{j-1})}(w)$ as $(w-\tau_j)+(\tau_j-\tau_{j-1}) = w-\tau_{j-1}$. On the other hand, if $\tau_j < \tau_{j-1}$, then $w - \tau_j = (w-\tau_{j-1}) + (\tau_{j-1}-\tau_j)$ and $w - \tau_{j-1} > 0$ as $\mathrm{ix}(w, \mathcal{M}^{(\tau_j)}) \geq 2 + \tau_{j-1} - \tau_j$; hence $S^{(\tau_j)}(w) = S^{(\tau_{j-1})}(w)$ also.

Therefore

$$E_b^{(\tau_j)}(x_{j-1}) = (-1)^{k-w+1} S^{(\tau_{j-1})}(w)(x_{j-1}).$$

Recall that $\mu(w) = \min\{r \in \mathcal{M}^{(\tau_{j-1})} : r \geq w\}$. By Lemma 4.3.5 and Lemma 4.2.9, we have

$$S^{(\tau_{j-1})}(w) = (-1)^{k-w-|\mathcal{M}^{(\tau_{j-1})}|+\mathrm{ix}(\mu(w),\mathcal{M}^{(\tau_{j-1})})} T^{(\tau_{j-1})}(\mu(w))$$

if $w \leq \max \mathcal{M}^{(\tau_{j-1})}$, and $S^{(\tau_{j-1})}(w) = (-1)^{k-w+1}$ if $w > \max \mathcal{M}^{(\tau_{j-1})}$.

If $w > \max \mathcal{M}^{(\tau_{j-1})}$, then $E_b^{(\tau_j)}(x_{j-1}) = 1$. On the other hand, if $w \leq \max \mathcal{M}^{(\tau_{j-1})}$, then $E_b^{(\tau_j)}(x_{j-1}) = E_{\hat{b}}^{(\tau_{j-1})}(x_{j-1})$ where $\hat{b} = \mathrm{ix}(\mu(w), \mathcal{M}^{(\tau_{j-1})})$. Now, if $\tau_j > \tau_{j-1}$, then $w > \tau_j$ and so $\mathrm{ix}(\mu(w), \mathcal{M}^{(\tau_{j-1})}) > \mathrm{ix}(\tau_j, \mathcal{M}^{(\tau_{j-1})})$; thus by Corollary 4.3.2 we have $E_b^{(\tau_j)}(x_{j-1}) > 0$. If $\tau_j < \tau_{j-1}$, then $\hat{b} \geq 2$ as $b \geq 2 + \tau_j - \tau_{j-1}$; thus, again by Corollary 4.3.2, we have $E_b^{(\tau_j)}(x_{j-1}) > 0$. $\square$

5.5.1 Alternative hypotheses

The hypotheses (A)–(F) presented in Section 5.3 are more general than has been required for the applications considered so far (see Chapter 6). So in this section we give two alternative sets of hypotheses: one for changing to a phase of higher type and one for changing to a phase of lower type. These hypotheses assume that the algorithm is changing phase and so cannot be used for the first phase. However, for the first phase, we often know the initial conditions exactly, so it is not hard to check the hypotheses (A)–(F). First we consider changing to a phase of a higher type.

Hypotheses for changing to a phase of higher type

Let $j > 1$ be the phase number.

(A2) The phase type τ_j is defined (according to Definition 4.3.1) and $\tau_j > \tau_{j-1}$.

(B2) At x_{j-1} we have

(i) $(-1)^{|\mathcal{M}^{(\tau_j)}|+1} \det \mathcal{C}^{(\tau_j)} > \delta$,

(ii) $(\Delta^{(\tau_j)})^2 \hat{y}_{\tau_j}(x_{j-1}) > 0$, and

(iii) $\Delta^{(\mathrm{P})}\Delta^{(\tau_j)} \hat{y}_{\tau_j}(x_{j-1}) \neq 0$.

(C2) For $w \in \mathcal{M}^{(\tau_j)} \backslash \mathcal{M}^{(\tau_{j-1})}$ we have

(i) $\Delta^{(\mathrm{P})} q_b^{(\tau_j)}(x_{j-1}) > 0$,

(ii) $\Delta^{(\mathrm{P})}\Delta^{(\tau_j)} q_b^{(\tau_j)}(x_{j-1}) \neq 0$,

(iii) $(\Delta^{(\tau_j)})^2 q_b^{(\tau_j)}(x_{j-1}) > 0$, and

(iv) $(\Delta^{(\mathrm{P})})^2 \hat{y}_w(x_{j-1}) > 0$

where $b = \mathrm{ix}(w, \mathcal{M}^{(\tau_j)})$.

(D2) For $w \in \mathcal{M}^{(\tau_j)} \cap \mathcal{M}^{(\tau_{j-1})}$ we have

$$q_b^{(\tau_j)}(x_{j-1}) > 0 \text{ and } \Delta^{(\mathrm{P})}\hat{y}_w(x_{j-1}) > 0$$

where $b = \mathrm{ix}(w, \mathcal{M}^{(\tau_j)})$.

We now prove that the hypotheses (A2)–(D2) imply the original hypotheses (A)–(F) when the Independent Types Property holds.

Lemma 5.5.7. *Assume that $j > 1$ and the Independent Types Property holds for a phase of type τ_j. If (A2)–(D2) hold, then there is a phase j of type τ_j and hypotheses (A)–(F) are satisfied for phase j.*

Proof. We treat each of the hypotheses (A)–(F) in turn.

Hypotheses (A) By (A2), it is enough to show that $(x_{j-1}, \hat{y}_0(x_{j-1}), \ldots, \hat{y}_m(x_{j-1}))$ lies in the closure of $V^{(\tau_j)}(\delta)$. By Corollary 4.3.2 we have

$$(x_{j-1}, \hat{y}_0(x_{j-1}), \ldots, \hat{y}_m(x_{j-1})) \in D(\delta).$$

So it remains to consider the functions in the set

$$\begin{aligned}\widehat{\mathcal{B}}^{(\tau_j)}(\delta) = \{(-1)^{|\mathcal{M}^{(\tau_j)}|+1} \det \mathcal{C}^{(\tau_j)} - \delta,\ & y_{\tau_j},\\ & q_b^{(\tau_j)} \quad (b = 1, \ldots, |\mathcal{M}^{(\tau_j)}|),\\ & E_b^{(\tau_j)} \quad (b = 2, \ldots, |\mathcal{M}^{(\tau_j)}|)\}.\end{aligned}$$

These functions are all non-negative by (B2) (ii), Corollary 4.3.2, Corollary 5.5.3, (D2), and Lemma 5.5.6. Therefore hypothesis (A) is satisfied.

Hypothesis (B) We need to show that each function in the set $\widehat{\mathcal{B}}^{(\tau_j)}(\delta)$ has positive growth at x_{j-1} during a phase of type τ_j. Any function that is positive at x_{j-1} has positive growth at x_{j-1}. Thus by Lemma 5.5.6, (B2) (i), (D2), it remains to consider the functions $\hat{y}_{\tau_j}$ and $q_b^{(\tau_j)}$ for b such that $b = \mathrm{ix}(w, \mathcal{M}^{(\tau_j)})$ for some $w \in \mathcal{M}^{(\tau_j)} \backslash \mathcal{M}^{(\tau_{j-1})}$. By Corollary 4.3.2, Lemma 5.5.1, and (B2) (ii), the function $\hat{y}_{\tau_j}$ has positive growth of order 2 at x_{j-1} during a phase of type τ_j. By Corollary 5.5.3, Lemma 5.5.4, and (C2) (iii), each function $q_b^{(\tau_j)}$ has positive growth of order 2 at x_{j-1} during a phase of type τ_j. Therefore hypothesis (B) is satisfied.

Hypothesis (C) For $w \in \mathcal{M}^{(\tau_j)} \backslash \mathcal{M}^{(\tau_{j-1})}$, the function $\hat{y}_w$ has positive growth of order 2 at x_{j-1} during a preprocessing subphase by Lemma 4.3.5, Lemma 5.5.5, and (C2) (iv). For $w \in \mathcal{M}^{(\tau_j)} \cap \mathcal{M}^{(\tau_{j-1})}$, the function $\hat{y}_w$ has positive growth of order 1 at x_{j-1} during a preprocessing subphase by Lemma 4.3.5, Corollary 4.3.2, and (D2). Therefore hypothesis (C) is satisfied.

Hypothesis (D) Hypothesis (D) follows from (B2) (i), (C2) (i), and (D2).

Hypothesis (E) As $\tau_j > \tau_{j-1}$, hypothesis (E) is satisfied by Lemma 5.5.6.

Hypothesis (F) Each function in the set $\widehat{\mathcal{B}}^{(\tau_j)}(\delta)$ has positive growth at x_{j-1} during a phase of type τ_j of order 0 or 2. By (B2) (iii), (D2), (C2) (i) and (ii), each function in the set $\mathcal{B}^{(\tau_j)}(\delta)$ is covered by case (i) or (iii) of Lemma 5.3.4. Hence hypothesis (F) is satisfied. □

Now we consider changing to a phase of a lower type.

Hypotheses for changing to a phase of lower type

Let $j > 1$ be the phase number.

(A3) The phase type τ_j is defined (according to Definition 4.3.1) and $\tau_j < \tau_{j-1}$.

(B3) At x_{j-1} we have

- (i) $(-1)^{|\mathcal{M}^{(\tau_j)}|+1} \det \mathcal{C}^{(\tau_j)} > \delta$, and
- (ii) $E_b^{(\tau_j)}(x_{j-1}) > 0$ for $b = 2, \ldots, \tau_{j-1} - \tau_j + 1$.

(C3) For $b = 1, \ldots, \left|\mathcal{M}^{(\tau_j)}\right|$ we have $q_b^{(\tau_j)}(x_{j-1}) > 0$.

(D3) For $w \in \mathcal{M}^{(\tau_j)} \backslash \{\tau_j\}$ we have $\Delta^{(\mathrm{P})} \hat{y}_w(x_{j-1}) > 0$.

We now prove that the hypotheses (A3)–(D3) imply the original hypotheses (A)–(F) when the Independent Types Property holds.

Lemma 5.5.8. *Assume that $j > 1$ and that the Independent Types Property holds for a phase of type τ_j. If (A3)–(D3) hold, then there is a phase j of type τ_j and hypotheses (A)–(F) are satisfied for phase j.*

Proof. By Corollary 4.3.2 we have

$$(x_{j-1}, \hat{y}_0(x_{j-1}), \ldots, \hat{y}_m(x_{j-1})) \in D(\delta).$$

So now consider the functions in the set $\widehat{\mathcal{B}}^{(\tau_j)}(\delta)$. By Corollary 4.3.2 we have $\hat{y}_{\tau_j} > 0$; while by (B3) (i) we have $(-1)^{\left|\mathcal{M}^{(\tau_j)}\right|+1} \det \mathcal{C}^{(\tau_j)} - \delta > 0$. The functions $q_b^{(\tau_j)}$ are positive at x_{j-1} by (C3). Finally by (B3) (ii) and Lemma 5.5.6 we have $E_b^{(\tau_j)} > 0$ for $b = 2, \ldots, \left|\mathcal{M}^{(\tau_j)}\right|$. Therefore hypothesis (A) is satisfied. Moreover, each function in the set $\widehat{\mathcal{B}}^{(\tau_j)}(\delta)$ is positive at x_{j-1}. Thus hypotheses (B), (D), (E), and (F) are all satisfied. The remaining hypothesis, (C), is satisfied by (D3) and Corollary 4.3.2. $\square$

Analysing deprioritised algorithms based on prioritised algorithms is a standard technique, which we call the deprioritised approach. The theory presented in this and the previous chapter allows the deprioritised approach to be applied to a large class of prioritised algorithms. This theory extends the work of Wormald [61]. In the next chapter we use the results of Chapter 4 and Chapter 5 to analyse some algorithms on random d-in d-out digraphs.

Chapter 6

Algorithms on Random Regular Digraphs

In this chapter we demonstrate the theory of Chapter 4 and Chapter 5 by applying the deprioritised approach to two algorithms for random d-in d-out digraphs (for fixed d). The first algorithm, called DominatingSet and given in Section 6.1, is an extension of the algorithm DominatingSet2 presented in Chapter 3. DominatingSet is currently the best algorithm for finding dominating sets of random d-in d-out digraphs. By analysing the deprioritised algorithm based on DominatingSet, we obtain the best known a.a.s. upper bounds on the domination number of random d-in d-out digraphs for $d = 2, 3, 4$. In particular, we improve upon the upper bound from analysing DominatingSet2 [42] for random 2-in 2-out digraphs.

In Section 6.2, we introduce the algorithm PathDominatingSet which finds 2-path dominating sets of random d-in d-out digraphs. Applying the deprioritised approach to PathDominatingSet, we determine the best known a.a.s. upper bounds on the 2-path domination numbers for $d \in \{2, \dots, 6\}$. Although quite a simple algorithm, PathDominatingSet is interesting as the deprioritised algorithm based on PathDominatingSet has a phase of type 0. (Note that a phase of type 0 is not the same as a preprocessing subphase.) As far as we are aware, this is the first case of a deprioritised algorithm having a phase of type 0.

d	DominatingSet	Best Previous
2	$0.38069n$	$0.39856n$
3	$0.32269n$	$0.57143n$
4	$0.28409n$	$0.55556n$

Table 6.1.1: Asymptotically almost sure upper bounds on the domination number of a random d-in d-out digraph.

6.1 Dominating Sets

We now present a prioritised algorithm for finding dominating sets of random d-in d-out digraphs. This algorithm, called DominatingSet, is an extension of DominatingSet2 presented in Chapter 3. Applying the theory of Chapter 4 and Chapter 5 to DominatingSet, we obtain the best known (a.a.s.) upper bounds on the domination number for a random d-in d-out digraph (when $d = 2, 3, 4$). These and previous bounds from the analysis of DominatingSet2 [42] (for $d = 2$) and Theorem 2.2.1 (for $d = 3, 4$) are given in Table 6.1.1.

We start this section by defining the operations of DominatingSet. Next the priorities of these operations are given and the definition of the algorithm is completed. We then determine, from the definition of the operations, the differential equations whose solutions describe the behaviour of deprioritised algorithm based on DominatingSet. Theorem 5.4.1 is applied in detail for $d = 2$, where we see that the sequence of phase types is $2, 3, 4, 5, 6$. In Section 6.1.4 we show that the Independent Types Property is satisfied for all phase types. So we are able to use the alternative hypotheses (A2)–(D2) for phases two to five. The application of Theorem 5.4.1 proceeds similarly for larger d and so is not given. When we apply the deprioritised approach to PathDominatingSet in Section 6.2, we will use the alternative hypotheses (A3)–(D3).

6.1.1 The operations and their priorities

The operations of DominatingSet are defined similarly to the operations of DominatingSet2 in Chapter 3. Recall that each operation of DominatingSet2 begins by randomly selecting a vertex of a given degree pair; we call this vertex the *processed* vertex. Also the type of an operation is determined by the degree pair of the processed vertex. DominatingSet has two classes of operations based on whether the degree pair (p, q) of the processed vertex satisfies $p > q$ or $p \leq q$. Operations processing vertices of degree pairs (p, q) with $p > q$ are straightforward generalisations of the operations of DominatingSet2 to arbitrary d. The remaining operations are more complicated.

As with DominatingSet2, the algorithm DominatingSet constructs a digraph G and a set D that becomes a dominating set for G at the end of the algorithm. The operations of DominatingSet are designed so that after each operation:

(a) the processed vertex is either in D or is an out-neighbour of a vertex in D, and

(b) every vertex of degree pair (d, d) is either in D or is an out-neighbour of a vertex in D.

Property (a) ensures that the algorithm terminates and (b) ensures that D is a dominating set of G at the end of the algorithm. We also define each operation to expose as many edges as possible, so that the dominating set returned by the algorithm DominatingSet is small.

First we consider operations that process vertices of degree pairs (p, q) with $p > q$. Such an operation performs the following steps in order:

(i) a vertex v of degree pair (p, q) is selected u.a.r.,

(ii) the vertex v is added to D,

(iii) the remaining $(d-p)$ free in-points and $(d-q)$ free out-points associated with v are exposed,

(iv) the free points associated with the $(d-q)$ new out-neighbours of v are exposed, and

(v) any vertices, other than v and the out-neighbours of v, that become saturated during the operation are added to D (such vertices are called accidental saturates).

That is, we select v, add v to D, and then call the procedure Saturate with v; as we did in DominatingSet2.

We now define operations that process vertices of degree pairs (p, q) with $p \leq q$. Such an operation begins by selecting a vertex u of degree pair (p, q) uniformly at random. Then the $(d-p)$ free in-points associated with u are exposed to obtain a multi-set $w_1, \ldots, w_{d-p}$ of new in-neighbours of u. A vertex v from $u, w_1, \ldots, w_{d-p}$ is chosen by comparing the degree pairs of the vertices from before the exposure of the in-points associated with u. Let (r, s) be the minimum degree pair of the vertices $w_1, \ldots, w_{d-p}$ with respect to the reverse lexicographic ordering (which we denote by $\preccurlyeq$). If $(p, q) \preccurlyeq (r, s)$, then we let $v = u$; otherwise we select v u.a.r. from the vertices of $w_1, \ldots, w_{d-p}$ with degree pair (r, s). We then add v to D and call Saturate with v. Note that, if $v \neq u$, then u is an out-neighbour of v and so u is saturated during the call to the procedure Saturate.

We now define the type of each operation and the priorities of the operation types. The type of an operation is determined by the degree pair of the vertex processed by that operation. So we also define types for degree pairs. In particular, the *type* of a degree pair is an integer in the set $\{0, \ldots, (d+1)^2 - 1\}$ such that

(i) the degree pairs $(0, 0)$ and (d, d) have type 0 and $(d+1)^2 - 1$, respectively,

(ii) the type of (p, q) is less than the type of (r, s) whenever $p + q < r + s$,

(iii) for degree pairs (p, q) and (r, s) with $p+q = r+s$, $p \leq q$, and $r > s$, the type of (p, q) is less than the type of (r, s),

(iv) for (p, q) and (r, s) with $p + q = r + s$, $p > q$, and $r > s$, the type of (p, q) is less than the type of (r, s) if and only if $p < r$, and

(v) for degree pairs (p, q) and (r, s) with $p+q = r+s$, $p \leq q$, and $r \leq s$, the type of (p, q) is less than the type of (r, s) if and only if $p > r$.

Clearly the type of a degree pair is well-defined. Now an operation has type τ if the degree pair of the vertex processed by that operation also has type τ. Then we define an operation of type τ_1 to have higher priority than an operation of type τ_2 if $\tau_1 > \tau_2$. The types of degree pairs for DominatingSet (for $d = 2, 3$) are given in Figure 6.1.1.

The priorities of the operation types are chosen so that operations which expose a large number of edges are performed more often than operations which expose a small number of edges. Somewhat counter-intuitively, this requires assigning higher priority to operations that expose fewer edges. However, the higher priority means that vertices of the corresponding degree pair do not build up, thus reducing the number of such operations in the long run.

Having defined the priorities of the operation types, we may now complete the definition of the algorithm DominatingSet. DominatingSet begins with the empty pairing and proceeds via a sequence of operations until no operation may be performed. The type of the next operation performed by DominatingSet is the type of highest priority that may be performed. So DominatingSet finishes when there are only vertices of degree pair (d, d). Hence DominatingSet returns a d-in d-out (multi-)digraph G together with a dominating set for G.

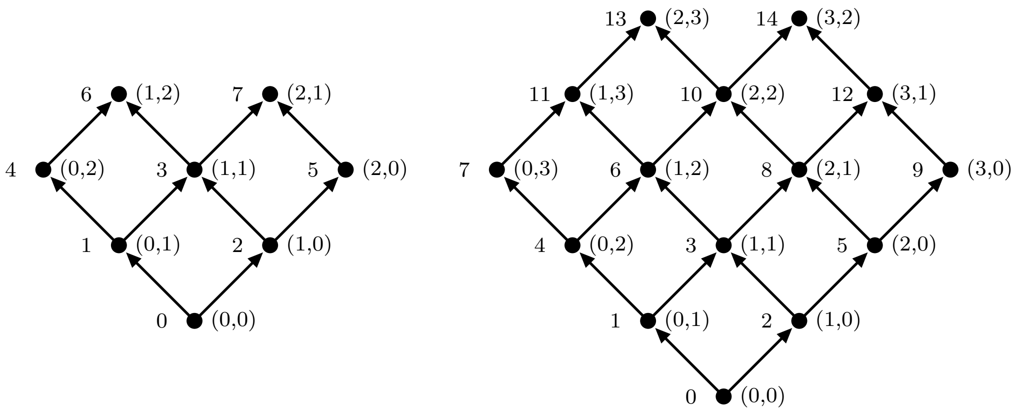


Figure 6.1.1: The types of operations displayed on the DPPD for DominatingSet for $d = 2, 3$ (with the vertex (d, d) not shown). The type of an operation is displayed next to the degree pair of the vertex processed by that operation.

6.1.2 The differential equations

In this section we determine the system of differential equations whose solution describes the deprioritised algorithm based on DominatingSet. First we derive functions that approximate the expected change in the random variables due to a single operation of DominatingSet. We are interested in the random variables $Z_{(i,j)}$ counting the number of vertices of degree pair (i, j), and $Z = |D|$ which counts the number of vertices that have been added to the dominating set. Later, when we apply Theorem 5.4.1, we use the random variables Y_i for $i = 0, \ldots, (d + 1)^2$; so we define $Y_i = Z_{(p,q)}$ where (p, q) has type i (for $i = 0, \ldots, (d+1)^2 - 1$) and $Y_{(d+1)^2} = Z$. However, we determine the differential equations using the more natural random variables $Z_{(i,j)}$ and D.

We start by determining functions $f^{(r)}_{(i,j)}$ and $f^{(r)}$ such that, for $0 \leq i, j \leq d$ and $r \in \{0, \ldots, (d + 1)^2 - 2\}$,

$$f^{(r)}_{(i,j)}(t/n, Z_{(0,0)}(t)/n, \ldots, Z_{(d,d)}(t)/n, Z(t)/n) + o(1)$$

is the expected change in $Z_{(i,j)}$ due to an operation of type r at time t, and

$$f^{(r)}(t/n, Z_{(0,0)}(t)/n, \dots, Z_{(d,d)}(t)/n, Z(t)/n) + o(1)$$

is the expected change in Z due to an operation of type r at time t. These functions correspond to the functions $f_{i,r}$ of Definition 4.1.1. Note that for now the underlying random process is that defined by the prioritised algorithm DominatingSet.

Consider an operation that chooses vertex v to add to D (so v is not an accidental saturate). During this operation there are six sorts of vertices:

- vertices that have no associated free points exposed,
- the vertex u processed by the operation,
- the vertex v which is added to D (which may be u),
- vertices, other than u, that are out-neighbours of v and so have all their associated free points exposed, called *rems*,
- vertices, other than u, v and the out-neighbours of v, that have an associated in-point exposed, called *in-incs*, and
- vertices, other than u, v and the out-neighbours of v, that have an associated out-point exposed, called *out-incs.*

Notice that the operations are defined using free points (either in or out) selected uniformly at random. So let

$$\rho = \sum_{0 \le i,j \le d} (d-i) Z_{(i,j)} = \sum_{0 \le i,j \le d} (d-j) Z_{(i,j)}$$

be the number of free in-points (which equals the number of free out-points). Then the functions $f^{(r)}_{(i,j)}$ and $f^{(r)}$ are rational functions of polynomials in the random variables $Z_{(i,j)}$ divided by powers of ρ. Hence the polynomial H of Definition 4.1.1 is equal to ρ. Since more than one edge may be exposed during an operation, the random variables $Z_{(i,j)}$ change during an operation. However, a constant number of edges are exposed (as d is fixed) and so the random variables only change by a constant amount. As $H = \rho = \Omega(n)$ while the deprioritised algorithm is analysed, the value of $Z_{(i,j)}/\rho$ during an operation is within $o(1)$ of its value at the start of the operation. So when determining the functions $f^{(r)}_{(i,j)}$ and $f^{(r)}$, we treat each $Z_{(i,j)}$ as fixed throughout each operation.

Now let $\mathbb{P}_{\text{in}}(w \in V_{(i,j)})$ be the probability that a vertex w, selected via a free in-point chosen uniformly at random, has degree pair (i,j). Define $\mathbb{P}_{\text{out}}(w \in V_{(i,j)})$ similarly. Then

$$\mathbb{P}_{\text{in}}(w \in V_{(i,j)}) = (d-i)Z_{(i,j)}/\rho \quad \text{and} \quad \mathbb{P}_{\text{out}}(w \in V_{(i,j)}) = (d-j)Z_{(i,j)}/\rho.$$

In-incs and Out-incs

For convenience we extend the definition of $Z_{(i,j)}$ so that $Z_{(i,j)} = 0$ for $i < 0$ or $j < 0$. The expected change in $Z_{(i,j)}$ due to an in-inc w is $\text{In}_{(i,j)} + o(1)$ where

$$\text{In}_{(i,j)} = \mathbb{P}_{\text{in}}(w \in V_{(i-1,j)}) - \mathbb{P}_{\text{in}}(w \in V_{(i,j)}) = ((d+1-i)Z_{(i-1,j)} - (d-i)Z_{(i,j)})/\rho.$$

Similarly, the expected change in $Z_{(i,j)}$ due to an out-inc w is $\text{Out}_{(i,j)} + o(1)$ where

$$\text{Out}_{(i,j)} = \mathbb{P}_{\text{out}}(w \in V_{(i,j-1)}) - \mathbb{P}_{\text{out}}(w \in V_{(i,j)}) = ((d+1-j)Z_{(i,j-1)} - (d-j)Z_{(i,j)})/\rho.$$

Rems

A rem is a vertex, other than the processed vertex, that is an out-neighbour of the vertex v added to D. Let w be a rem. Contributions to the expected change in $Z_{(i,j)}$ from calling Saturate with w come from three sources: w moving to $V_{(d,d)}$, in-incs from exposing the free out-points associated with w, and out-incs from exposing the free in-points associated with w. Let F_{in} and F_{out} be the number of free in-points and out-points associated with w (respectively) before the edge (v,w) is added, and let $\delta_{a,b}$ be the usual Kronecker delta.

Then the expected change in $Z_{(i,j)}$ due to a rem is $\text{Rem}_{(i,j)} + o(1)$ where

$$\begin{aligned}\text{Rem}_{(i,j)} &= \delta_{i,d}\delta_{j,d} - \mathbb{P}_{\text{in}}(w \in V_{(i,j)}) + \mathbb{E}(F_{\text{in}} - 1)\,\text{Out}_{(i,j)} + \mathbb{E}(F_{\text{out}})\,\text{In}_{(i,j)} \\ &= \delta_{i,d}\delta_{j,d} - (d-i)Z_{(i,j)}/\rho \\ &\quad + \frac{1}{\rho}\left[\sum_{p=0}^{d}\sum_{q=0}^{d}(d-p-1)(d-p)Z_{(p,q)}\right]\text{Out}_{(i,j)} \\ &\quad + \frac{1}{\rho}\left[\sum_{p=0}^{d}\sum_{q=0}^{d}(d-q)(d-p)Z_{(p,q)}\right]\text{In}_{(i,j)}.\end{aligned}$$

Operations

There are two distinct sorts of operations as described above. Consider an operation of type r that processes a vertex u of degree pair (p,q). Let v be the vertex added to D. Then there are two cases to consider.

Case $p > q$: In this case $v = u$ always. So there are $d-q$ rems and $d-p$ out-incs from exposing the free points associated with u. Also u moves from $V_{(p,q)}$ to $V_{(d,d)}$. Therefore the expected change in $Z_{(i,j)}$ is $\text{Op}r_{(i,j)} + o(1)$ where

$$\text{Op}r_{(i,j)} = \delta_{i,d}\delta_{j,d} - \delta_{i,p}\delta_{j,q} + (d-q)\text{Rem}_{(i,j)} + (d-p)\text{Out}_{(i,j)}.$$

Case $p \le q$: In this case, either $v = u$ or v is an in-neighbour of u obtained by exposing a free in-point associated with u. Asymptotically almost surely the in-neighbours of u are distinct (as we do not consider the very end of the algorithm), so this is assumed for the following analysis. Let $\chi_p = \{(r,s) \in \mathbb{Z}^2 : 0 \le r,s \le d\}^{d-p}$. For $\mathbf{z} = (z_1, \dots, z_{d-p}) \in \chi_p$, let $\mathcal{E}_{\mathbf{z}}$ be the event that, for $a = 1, \dots, d-p$, the vertex w_a has degree pair z_a. Then for $\mathbf{z} \in \chi_p$, we define $\gamma_{(i,j)}(\mathbf{z})$ such that $\gamma_{(i,j)}(\mathbf{z}) + o(1)$ is the expected change in $Z_{(i,j)}$ conditioned on $\mathcal{E}_{\mathbf{z}}$.

We determine $\gamma_{(i,j)}(\mathbf{z})$ by considering the contributions to the expected change in $Z_{(i,j)}$ due to the vertices u, v, and $w_1, \dots, w_{d-p}$. If w_a is not added to D, then the contribution of w_a to the change in $Z_{(i,j)}$ is $\delta_{r_a,i}\delta_{s_a,j-1} - \delta_{r_a,i}\delta_{s_a,j}$. Let (r,s) be the minimum of $\{(r_a,s_a) : a = 1, \dots, d-p\}$ with respect to the reverse lexicographic ordering. Define $\lambda_{\mathbf{z}} = 1$ if $v = u$, and $\lambda_{\mathbf{z}} = 0$ if $u \neq v$. Then we take

$$\begin{aligned}\gamma_{(i,j)}(\mathbf{z}) = \quad & \delta_{i,d}\delta_{j,d} - \delta_{p,i}\delta_{q,j} + \left[\sum_{w_a \neq v} (\delta_{r_a,i}\delta_{s_a,j-1} - \delta_{r_a,i}\delta_{s_a,j})\right] \\ & + \lambda_{\mathbf{z}}(d-q)\mathrm{Rem}_{(i,j)} \\ & + (1-\lambda_{\mathbf{z}})[(d-s-1)\mathrm{Rem}_{(i,j)} + (d-r)\mathrm{Out}_{(i,j)} \\ & \qquad + (d-q)\mathrm{In}_{(i,j)} + \delta_{i,d}\delta_{j,d} - \delta_{i,r}\delta_{j,s}].\end{aligned}$$

The probability that $\mathbf{z}$ is the sequence of degree pairs corresponding to $w_1, \ldots, w_{d-p}$ is

$$\prod_{a=1}^{d-p} \mathbb{P}_{\mathrm{out}}(w_a \in V_{(r_a,s_a)}) + o(1).$$

Hence the expected change in $Y_{(i,j)}$ is $\mathrm{Op}r_{(i,j)} + o(1)$ where

$$\mathrm{Op}r_{(i,j)} = \sum_{\mathbf{z} \in \chi_p} \left(\gamma_{(i,j)}(\mathbf{z}) \prod_{a=1}^{d-p} \mathbb{P}_{\mathrm{out}}(w_a \in V_{(r_a,s_a)})\right).$$

Changes in the Size of the Dominating Set

The expected change in the size of the dominating set due to any operation is one plus the expected number of accidental saturates. Accidental saturates are either in-incs or out-incs; they are never rems. Consider an operation of type r that processes a vertex u of degree pair (p, q). Then again we need to consider two cases.

Case $p > q$: The expected change in Z is $\mathrm{dom}_r + o(1)$ where

$$\mathrm{dom}_r = 1 + (d-q)(\mathrm{Rem}_{(d,d)} - 1) + (d-p)\mathrm{Out}_{(d,d)}.$$

Case $p \leq q$: We determine a function dom_r such that the expected change in Z is $\mathrm{dom}_r + o(1)$. This function takes a similar form to $\mathrm{Op}r_{(i,j)}$ for $p \leq q$. In particular we take

$$\mathrm{dom}_r = 1 + \sum_{\mathbf{z} \in \chi_p} \left(\sigma(\mathbf{z}) \prod_{a=1}^{d-p} \mathbb{P}_{\mathrm{out}}(w_a \in V_{(r_a,s_a)})\right)$$

where

$$\sigma(\mathbf{z}) = \left[\sum_{w_a \neq v} \delta_{r_a,d}\delta_{s_a,d-1}\right] + \lambda_{\mathbf{z}}(d-q)(\mathrm{Rem}_{(d,d)} - 1)$$
$$+ (1-\lambda_{\mathbf{z}})\left[(d-s-1)(\mathrm{Rem}_{(d,d)} - 1) + (d-r)\mathrm{Out}_{(d,d)} + (d-q)\mathrm{In}_{(d,d)}\right]$$

and $\lambda_{\mathbf{z}}$ is defined as above.

The Functions $f^{(r)}_{(i,j)}$ and $f^{(r)}$

To obtain the functions $f^{(r)}_{(i,j)}$ and $f^{(r)}$, we scale the random variables by setting $Z_{(i,j)}(t) = nz_{(i,j)}(t/n)$ and $Z(t) = nz(t/n)$. Then we write $\mathrm{Op}r_{(i,j)}$ and dom_r in terms of $z_{(i,j)}$ (for $0 \le i, j \le d$) and z. (Of course, when we apply Theorem 5.4.1 we use the scaled variables y_i defined by $Y_i(t) = ny_i(t/n)$ for $i = 0, \ldots, (d+1)^2$). First we have

$$\begin{aligned}
\mathrm{In}_{(i,j)} &= \frac{(d+1-i)z_{(i-1,j)} - (d-i)z_{(i,j)}}{s}, \\
\mathrm{Out}_{(i,j)} &= \frac{(d+1-j)z_{(i,j-1)} - (d-j)z_{(i,j)}}{s}, \text{ and} \\
\mathrm{Rem}_{(i,j)} &= \delta_{i,d}\delta_{j,d} - (d-i)z_{(i,j)}/s \\
&\quad + (1/s)\left(\sum_{p=0}^{d}\sum_{q=0}^{d}(d-p-1)(d-p)z_{(p,q)}\right)\mathrm{Out}_{(i,j)} \\
&\quad + (1/s)\left(\sum_{p=0}^{d}\sum_{q=0}^{d}(d-p)(d-q)z_{(p,q)}\right)\mathrm{In}_{(i,j)}
\end{aligned}$$

where $s = \sum_{p=0}^{d}\sum_{q=0}^{d}(d-p)z_{(p,q)}$. The other equations follow from those above.

The Differential Equations

We use the theory of Chapter 4 (in particular Section 4.3) to define differential equations whose solutions, as we prove later, describe the random variables of the deprioritised algorithm based on DominatingSet. Recall that the differential equations for a given phase are obtained from the type distribution of that phase and the functions $f^{(r)}_{(i,j)}$ and $f^{(r)}$. The type distribution for a phase of type τ is given by Definition 4.2.1 with $\mathcal{O} = \mathcal{M}^{(\tau)}$, where $\mathcal{M}^{(\tau)}$ is the irreducible type set for a phase of type τ (as defined in Definition 4.2.5). We determine the irreducible type sets in the next section using the DPPD method described in Section 4.2.4.

The differential equations relate to the deprioritised algorithm based on DominatingSet. So we now consider the random process $\{G_t\}$ defined by the deprioritised algorithm. Recall that $p_r^{(\tau)}$ is the probability of performing an operation of type r during a phase of type τ. So we obtain the differential equations

$$\frac{dz_{(i,j)}}{dx} = \sum_{r \in \mathcal{M}^{(\tau)}} p_r^{(\tau)} f^{(r)}_{(i,j)}$$

and

$$\frac{dz}{dx} = \sum_{r \in \mathcal{M}^{(\tau)}} p_r^{(\tau)} f^{(r)}.$$

The first function approximates $\mathbb{E}(Z_{(i,j)}(t+1) - Z_{(i,j)}(t) \mid G_0, \ldots, G_t)$, while the second approximates $\mathbb{E}(Z(t+1) - Z(t) \mid G_0, \ldots, G_t)$.

The solutions to the differential equations $dz_{(i,j)}/dx$ and dz/dx describe the random variables $Z_{(i,j)}$ and Z during the deprioritised algorithm. However, it is more convenient to solve the differential equations

$$\frac{dz_{(i,j)}}{dz} = \frac{dz_{(i,j)}}{dx} \Big/ \frac{dz}{dx}.$$

So we do not determine (approximations to) the values x_j for $j = 1, \ldots, K$. Instead we determine approximations to $z_{(i,j)}(x_j)$ and $z(x_j)$. Of course we are still able to check the hypotheses to Theorem 5.4.1, as the functions $f^{(r)}_{(i,j)}$ and $f^{(r)}$ do not depend on the variable x.

6.1.3 Applying the deprioritised approach

We wish to apply the deprioritised approach, as described in Chapter 4 and Chapter 5, to the analysis of DominatingSet. So we must show that DominatingSet is a deprioritisable algorithm. We also need to determine the irreducible type sets. Then Theorem 5.4.1 may be applied.

Consider the definition of deprioritisable algorithms given in Definition 4.1.1. Parts (i), (ii), and (iv) of Definition 4.1.1 follow from the definition of DominatingSet. The previous section shows that part (iii) is satisfied (with the polynomial $H = s$ and integers $k = (d+1)^2 - 1$ and $m = (d+1)^2$). Notice that in each operation, since d is fixed, the number of edges exposed is bounded; hence part (v) is satisfied. Since the random variables count vertices, part (vi) is satisfied with $M = 1$. Finally, part (vii) is satisfied as $Y_0(0) = n$ and $Y_i(0) = 0$ for $i \geq 1$. Therefore DominatingSet is a deprioritisable algorithm.

6.1.4 Determining the irreducible type sets

We determine the irreducible type sets using the DPPD method described in Section 4.2.4. Recall the definition of $B_{(i,j)}$-determined from Section 4.2.4. It will be more convenient to use the variables $z_{(p,q)}$ rather than the variables y_i. To translate between the two sets of variables, we define $\nu(p,q) = i$ where i is the type of (p,q) (as in Section 6.1.1) and $\mathrm{dp}(\nu(p,q)) = (p,q)$. Note that, as for DominatingSet2, when $(i,j) \neq (d,d)$ we have

$$B_{(i,j)} = \{(i-1,j),(i,j-1)\} \cap \{(p,q) \,:\, 0 \leq p,q \leq d\}.$$

Therefore $f^{(r)}_{(i,j)}$ is $B_{(i,j)}$-determined if

(a) every term in the numerator of $f^{(r)}_{(i,j)} + \delta_{\nu(i,j),r}$ contains at least one variable in the set $\{z_{(i,j)}, z_{(i-1,j)}, z_{(i,j-1)}\}$, and

(b) for all $(p,q) \in B_{(i,j)}$, there exists a term of the numerator of $f^{(r)}_{(i,j)} + \delta_{\nu(i,j),r}$ containing the variable $z_{(p,q)}$ but no variables in the set

$$\{z_{\mathrm{dp}(j)} \,:\, j = \tau+1, \ldots, k\}.$$

Note that we do not need to consider the function $f^{(r)}$ as $f^{(r)}$ corresponds to the expected change in Y_m and $m > k$.

We will show that $f^{(r)}_{\mathrm{dp}(i)}$ is $B_{\mathrm{dp}(i)}$-determined for all $i \in \{0, \dots, k\}$ and $r \in \{0, \dots, k\}$. Then, by Lemma 4.2.11, we may determine the irreducible type sets for all phase types; moreover, the Independent Types Property will hold for phases of all types.

Fix $(i, j) \neq (d, d)$ and consider r such that $\mathrm{dp}(r) = (p, q)$ with $p > q$. Then

$$f^{(r)}_{(i,j)} + \delta_{\nu(i,j),r} = (d - q)\mathrm{Rem}_{(i,j)} + (d - p)\mathrm{Out}_{(i,j)}.$$

Notice that every term of the numerators of $\mathrm{In}_{(i,j)}$, $\mathrm{Out}_{(i,j)}$, and $\mathrm{Rem}_{(i,j)}$ involves $z_{(i,j)}$ or $z_{(i-1,j)}$, or $z_{(i,j-1)}$. Therefore part (a) of the definition of $B_{(i,j)}$-determined is satisfied. Next we show that part (b) is satisfied. Let $(s, t) \in B_{(i,j)}$. From the definition of $\mathrm{Rem}_{(i,j)}$, it is clear that the numerator of $\mathrm{Rem}_{(i,j)}$ contains a term involving only $z_{(s,t)}$ and $z_{(0,0)}$. Notice that the function $H = s$ contains the term $z_{(0,0)}$. Thus if we multiply the numerator of $\mathrm{Rem}_{(i,j)}$ by any power of H, the result still contains a term involving only $z_{(s,t)}$ and $z_{(0,0)}$. Hence part (b) is satisfied.

Now consider r such that $\mathrm{dp}(r) = (p, q)$ with $p \leq q$. Then

$$\begin{aligned} f^{(r)}_{(i,j)} + \delta_{\nu(i,j),r} = \quad & \mathrm{Rem}_{(i,j)} \left[(d-p)\mathbb{P}(u = v) + (d - s - 1)\mathbb{P}(u \neq v)\right] \\ & + \mathrm{Out}_{(i,j)} \left[(d - r)\mathbb{P}(u \neq v)\right] + \mathrm{In}_{(i,j)} \left[(d - q)\mathbb{P}(u \neq v)\right] \\ & + \sum_{\mathbf{z} \in \chi_p} \left[\left(\sum_{w_a \neq v} \delta_{r_a,i}\delta_{s_a,j-1} - \delta_{r_a,i}\delta_{s_a,j} \right) - (1 - \lambda_{\mathbf{z}})\delta_{r,i}\delta_{s,j} \right] \mathbb{P}(\mathcal{E}_{\mathbf{z}}) \end{aligned}$$

where $\mathbb{P}(u = v) = \sum_{\mathbf{z} \in \chi_p} \lambda_{\mathbf{z}} \mathbb{P}(\mathcal{E}_{\mathbf{z}})$ and $\mathbb{P}(u \neq v) = \sum_{\mathbf{z} \in \chi_p} (1 - \lambda_{\mathbf{z}})\mathbb{P}(\mathcal{E}_{\mathbf{z}})$. Notice that the sum

$$\left(\sum_{w_a \neq v} \delta_{r_a,i}\delta_{s_a,j-1} - \delta_{r_a,i}\delta_{s_a,j} \right) - (1 - \lambda_{\mathbf{z}})\delta_{r,i}\delta_{s,j}$$

is non-zero only when $\mathrm{dp}(w_a) = (i, j)$ or $\mathrm{dp}(w_a) = (i, j - 1)$ for some a. In this case the numerator of

$$\mathbb{P}(\mathcal{E}_{\mathbf{z}}) = \prod_{a=1}^{d-p} \mathbb{P}_{\mathrm{out}}(w_a \in V_{(r_a,s_a)})$$

contains either the variable $z_{(i,j)}$ or the variable $z_{(i,j-1)}$. Hence part (a) of the definition of $B_{(i,j)}$-determined is satisfied.

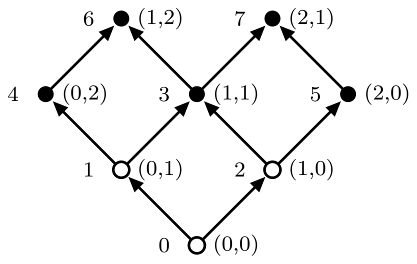


Figure 6.1.2: The coloured DPPD for a phase of type 2 of DominatingSet for $d = 2$.

We show part (b) as we did above; so let $(s, t) \in B_{(i,j)}$. Notice that

$$\mathrm{Rem}_{(i,j)}\left[(d-p)\mathbb{P}(u=v) + (d-s-1)\mathbb{P}(u \neq v)\right]$$

contains the term $Cz_{(p,q)}z_{(0,0)}^{d-p+1}$ for some $C \neq 0$. This is the only way such a term can be obtained and so part (b) is satisfied.

Therefore $f_{\mathrm{dp}(i)}^{(r)}$ is $B_{\mathrm{dp}(i)}$-determined for all $i \in \{0, \ldots, k\}$ and $r \in \{0, \ldots, k\}$. Thus, for all phase types, we may determine the irreducible type sets using the DPPD method and the Independent Types Property holds.

We determine the irreducible type sets for DominatingSet for $d = 2$ only. Figure 6.1.2 gives the coloured DPPD for a phase of type 2. Using the DPPD method, we see that the irreducible type set for a phase of type 2 is $\{2, 3, 4, 5\}$. Table 6.1.2 gives the irreducible type sets for phases of each type.

Finally we must choose the type of the first phase of the deprioritised algorithm based on DominatingSet. We take this type to be the type of the first phase of DominatingSet. Consider the first operation of DominatingSet, which has type 0. Asymptotically almost surely the first operation creates vertices of degree pairs $(2, 2)$, $(1, 0)$, and $(0, 1)$ only. Thus the second operation a.a.s. has type 2 and so phase one of DominatingSet has type 2. Hence the deprioritised algorithm based

Phase Type	Irreducible Type Sets
0	$\{0,1,2\}$
1	$\{1,2,3,4\}$
2	$\{2,3,4,5\}$
3	$\{3,4,5,6,7\}$
4	$\{4,5,6,7\}$
5	$\{5,6,7\}$
6	$\{6,7\}$
7	$\{7\}$

Table 6.1.2: The operations types that may occur during a phase of a given type.

on DominatingSet also begins with a phase of type 2. The types of any subsequent phases can be determined from the application of Theorem 5.4.1, as described in Section 4.3.

6.1.5 Numerical analysis for $d = 2$

We only give details for the case $d = 2$. The analysis for larger d proceeds similarly, but with many more phases (the number of phases appears to grow quadratically with d). We check the hypotheses to Theorem 5.4.1 with solutions to the differential equations obtained (non-rigorously) using the fourth order Runge-Kutta method. Note that these hypotheses depend on the parameter δ. We leave δ unspecified and extend the numerical solutions as far as possible while the hypotheses are satisfied for some $\delta > 0$. We may then take δ sufficiently small so as to include each point of the numerical solutions.

First we show that the initial conditions $(0, \hat{y}_0(0), \dots, \hat{y}_9(0))$ lie in $D(\delta)$. Recall that

$$\hat{y}_0(0) = \lim_{n\to\infty} Y_0(0)/n = 1 \quad \text{and} \quad \hat{y}_i(0) = \lim_{n\to\infty} Y_i(0)/n = 0 \quad (\text{for } i = 1, \dots, 9).$$

As we have exact values for $\hat{y}_i(0)$, the function values we calculate for phase one are also exact. Thus $H(0, \hat{y}_0(0), \ldots, \hat{y}_9(0)) = 2$ and so

$$(0, \hat{y}_0(0), \ldots, \hat{y}_9(0)) \in D(\delta).$$

We start the analysis with a phase of type 2.

Checking hypotheses for phase 1 of type 2

Hypothesis (A) To satisfy hypothesis (A) we need to show that

$$(x_0, \hat{y}_0(x_0), \ldots, \hat{y}_9(x_0))$$

lies in the closure of $V^{(2)}(\delta)$. Above we showed that

$$(x_0, \hat{y}_0(x_0), \ldots, \hat{y}_9(x_0)) \in D(\delta),$$

show it just remains to show that the functions

$$-\det \mathcal{C}^{(2)} - \delta,\ \hat{y}_2,\ q_b^{(2)} \text{ for } b = 1, 2, 3, 4, \text{ and } E_b^{(2)} \text{ for } b = 2, 3, 4,$$

are all non-negative at $x_0 = 0$. Calculating the values of these functions numerically at $(x_0, \hat{y}_0(x_0), \ldots, \hat{y}_9(x_0))$ we find

$$-\det \mathcal{C}^{(2)} - \delta = 1,\ \hat{y}_2 = 0,$$
$$q_b^{(2)} = \delta_{b,1} \text{ for } b = 1, 2, 3, 4, \text{ and } E_b^{(2)} = 1 \text{ for } b = 2, 3, 4.$$

Therefore hypothesis (A) is satisfied.

Hypothesis (B) We need to show that each function in $\widehat{\mathcal{B}}^{(2)}(\delta)$ has positive growth at x_0 during a phase of type 2. Any function that is positive at x_0 has positive growth at x_0. Thus we only need to consider the functions $\hat{y}_2$ and $q_b^{(2)}$ for $b = 2, 3, 4$. Using automatic differentiation [38] we calculate the derivatives of these functions and find

$$\Delta^{(2)} \hat{y}_2(x_0) = 3 \quad \text{and} \quad \Delta^{(2)} q_b^{(2)}(x_0) = \begin{cases} 21, & \text{if } b = 2 \\ 4.5, & \text{if } b = 3 \\ 6, & \text{if } b = 4. \end{cases}$$

Therefore hypothesis (B) is satisfied.

Hypothesis (C) We must show that $\hat{y}_i$ (for $i = 2, 3, 4, 5$) has positive growth at x_0 during a preprocessing subphase. Since $\hat{y}_i(x_0) = 0$ for each i, we must consider the derivatives (as calculated in a preprocessing subphase) of $\hat{y}_i$ for each i. We have

$$\Delta^{(\mathrm{P})}\hat{y}_2(x_0) = 4, \ \Delta^{(\mathrm{P})}\hat{y}_i(x_0) = 0 \text{ for } i = 3, 4, 5$$

and

$$(\Delta^{(\mathrm{P})})^2\hat{y}_i(x_0) = \begin{cases} 32, & \text{if } i = 3 \\ 8, & \text{if } i = 4, 5. \end{cases}$$

Therefore hypothesis (C) is satisfied.

Hypothesis (D) We must show that the functions $-\det \mathcal{C}^{(2)} - \delta$ and $q_b^{(2)}$ (for $b = 1, 2, 3, 4$) have positive growth at x_0 during a preprocessing subphase. We have already shown that $-\det \mathcal{C}^{(2)} - \delta$ and $q_1^{(2)}$ are positive at x_0 and so have positive growth. We consider the derivatives at x_0 of $q_b^{(2)}$ for $b = 2, 3, 4$:

$$\Delta^{(\mathrm{P})}q_b^{(2)}(x_0) = \begin{cases} 28, & \text{if } b = 2 \\ 6, & \text{if } b = 3 \\ 8, & \text{if } b = 4 \end{cases}$$

Therefore hypothesis (D) is satisfied.

Hypothesis (E)

For hypothesis (A) we showed that $E_b^{(2)}(x_0)$ is positive for $b = 2, 3, 4$. Therefore hypothesis (E) is also satisfied.

Hypothesis (F)

Each function of $\widehat{\mathcal{B}}^{(2)}(\delta)$ has positive growth during a phase of type 2 of order 0 or 1, as shown when considering hypothesis (B). For hypotheses (C) and (D), we showed that the functions with positive growth of order 1 during a phase of type 2 also have positive growth of order 1 during the preprocessing subphase. Thus hypothesis (F) is satisfied by Lemma 5.3.4.

Computing numerical solutions to the differential equations $dy_i/dy_9 = dz_{\mathrm{dp}(i)}/dz$ (for $i = 0, \ldots, 8$) we find that $\hat{y}_9(x_1) = z(x_1)$ satisfies

$$0.0422191 < \hat{y}_9(x_1) < 0.0422192.$$

The solutions exit $V^{(2)}(\delta)$ at the boundary $q_1^{(2)} = E_2^{(2)} = 0$. Thus there is a next phase and this phase has type 3. We check the hypotheses for phase 2 at $\hat{x}_1$ such that $\hat{y}_9(\hat{x}_1) = 0.0422191$, instead of at x_1. Earlier, in Section 6.1.4, we showed that the Independent Types Property hold for phases of all types. Hence we may use the alternative hypotheses (A2)–(D2).

Checking hypotheses for phase 2 of type 3

Hypothesis (A2) As mentioned above.

Hypothesis (B2)

(i) $\det \mathcal{C}^{(3)}(\hat{x}_1) = 0.82592\cdots$

(ii) $(\Delta^{(3)})^2 \hat{y}_3(\hat{x}_1) = 20.91221\cdots$

(iii) $\Delta^{(\mathrm{P})}\Delta^{(3)}\hat{y}_3(\hat{x}_1) = 25.34147\cdots$

Hypothesis (C2) Note that $\mathcal{M}^{(3)} = \{3,4,5,6,7\}$ and that $\mathcal{M}^{(2)} = \{2,3,4,5\}$.

(i) $\Delta^{(\mathrm{P})}q_4^{(3)}(\hat{x}_1) = 2.25177\cdots$ and $\Delta^{(\mathrm{P})}q_5^{(3)}(\hat{x}_1) = 2.25177\cdots$

(ii) $\Delta^{(\mathrm{P})}\Delta^{(3)}q_4^{(3)}(\hat{x}_1) = 46.43098\cdots$ and $\Delta^{(\mathrm{P})}\Delta^{(3)}q_5^{(3)}(\hat{x}_1) = 35.21154\cdots$

(iii) $(\Delta^{(3)})^2 q_4^{(3)}(\hat{x}_1) = 38.31563\cdots$ and $(\Delta^{(3)})^2 q_5^{(3)}(\hat{x}_1) = 29.05715\cdots$

(iv) $(\Delta^{(\mathrm{P})})^2 \hat{y}_6(\hat{x}_1) = 3.11359\cdots$ and $(\Delta^{(\mathrm{P})})^2 \hat{y}_7(\hat{x}_1) = 3.05130\cdots$

Hypothesis (D2)

(i) $$q_b^{(3)}(\hat{x}_1) = \begin{cases} 0.54965\cdots, & \text{if } b = 1 \\ 0.14315\cdots, & \text{if } b = 2 \\ 0.13310\cdots, & \text{if } b = 3 \end{cases}$$

(ii) $$\Delta^{(\mathrm{P})}\hat{y}_i(\hat{x}_1) = \begin{cases} 0.88740\cdots, & \text{if } i = 3 \\ 0.22521\cdots, & \text{if } i = 4 \\ 0.21848\cdots, & \text{if } i = 5 \end{cases}$$

Computing numerical solutions to the differential equations we find that $\hat{y}_9(x_2)$ satisfies

$$0.1379476 < \hat{y}_9(x_2) < 0.1379477.$$

The solutions exit $V^{(3)}(\delta)$ at the boundary $q_1^{(3)} = E_2^{(3)} = 0$. Thus there is a next phase and this phase has type 4. We check the hypotheses for phase 3 at $\hat{x}_2$ such that $\hat{y}_9(\hat{x}_2) = 0.1379476$ instead of x_2.

Checking hypotheses for phase 3 of type 4

Hypothesis (A2) As mentioned above.

Hypothesis (B2)

(i) $\det \mathcal{C}^{(4)}(\hat{x}_2) = 0.83502\cdots$

(ii) $(\Delta^{(4)})^2\hat{y}_4(\hat{x}_2) = 3.98914\cdots$

(iii) $\Delta^{(\mathrm{P})}\Delta^{(4)}\hat{y}_4(\hat{x}_2) = 1.69585\cdots$

Hypothesis (C2) Note that $\mathcal{M}^{(4)} = \{4,5,6,7\} \subset \mathcal{M}^{(3)}$; so there is nothing to check.

Hypothesis (D2)

(i) $$q_b^{(4)}(\hat{x}_2) = \begin{cases} 0.32644\cdots, & \text{if } b = 1 \\ 0.25305\cdots, & \text{if } b = 2 \\ 0.14588\cdots, & \text{if } b = 3 \\ 0.10964\cdots, & \text{if } b = 4 \end{cases}$$

(ii) $$\Delta^{(\mathrm{P})}\hat{y}_i(\hat{x}_2) = \begin{cases} 0.52984\cdots, & \text{if } i = 4 \\ 0.48551\cdots, & \text{if } i = 5 \\ 0.22935\cdots, & \text{if } i = 6 \\ 0.21036\cdots, & \text{if } i = 7 \end{cases}$$

Computing numerical solutions to the differential equations we find that $\hat{y}_9(x_3)$ satisfies

$$0.2334262 < \hat{y}_9(x_3) < 0.2334263.$$

The solutions exit $V^{(4)}(\delta)$ at the boundary $q_1^{(4)} = E_2^{(4)} = 0$. Thus there is a next phase and this phase has type 5. We check the hypotheses for phase 4 at $\hat{x}_3$ such that $\hat{y}_9(\hat{x}_3) = 0.2334262$.

Checking hypotheses for phase 4 of type 5

Hypothesis (A2) As mentioned above.

Hypothesis (B2)

(i) $\det \mathcal{C}^{(5)}(\hat{x}_3) = 1.14587\cdots$

(ii) $(\Delta^{(5)})^2 \hat{y}_4(\hat{x}_3) = 2.05665\cdots$

(iii) $\Delta^{(\mathrm{P})}\Delta^{(5)}\hat{y}_5(\hat{x}_3) = -3.77295\cdots$

Hypothesis (C2) Note that $\mathcal{M}^{(5)} = \{5, 6, 7\} \subset \mathcal{M}^{(4)}$; so there is nothing to check.

Hypothesis (D2)

(i) $q_b^{(5)}(\hat{x}_3) = \begin{cases} 0.36307\cdots, & \text{if } b = 1 \\ 0.35304\cdots, & \text{if } b = 2 \\ 0.42975\cdots, & \text{if } b = 3 \end{cases}$

(ii) $\Delta^{(\mathrm{P})}\hat{y}_i(\hat{x}_3) = \begin{cases} 0.52817\cdots, & \text{if } i = 5 \\ 0.79990\cdots, & \text{if } i = 6 \\ 0.62516\cdots, & \text{if } i = 7 \end{cases}$

Computing numerical solutions to the differential equations we find that $\hat{y}_9(x_4)$ satisfies

$$0.3746430 < \hat{y}_9(x_4) < 0.3746431.$$

The solutions exit $V^{(5)}(\delta)$ at the boundary $q_1^{(5)} = E_2^{(5)} = 0$. Thus there is a next phase and this phase has type 6. We check the hypotheses for phase 5 at $\hat{x}_4$ such that $\hat{y}_9(\hat{x}_4) = 0.3746430$.

Checking hypotheses for phase 5 of type 6

Hypothesis (A2) As mentioned above.

Hypothesis (B2)

(i) $\det \mathcal{C}^{(6)}(\hat{x}_4) = 0.67586\cdots$

(ii) $(\Delta^{(6)})^2 \hat{y}_6(\hat{x}_4) = 1.40866\cdots$

(iii) $\Delta^{(\mathrm{P})}\Delta^{(6)}\hat{y}_6(\hat{x}_4) = -693.54375\cdots$

Hypothesis (C2) Note that $\mathcal{M}^{(6)} = \{6, 7\} \subset \mathcal{M}^{(5)}$; so there is nothing to check.

Hypothesis (D2)

(i) $q_b^{(6)}(\hat{x}_4) = \begin{cases} 0.35556\cdots, & \text{if } b = 1 \\ 0.32030\cdots, & \text{if } b = 2 \end{cases}$

(ii) $\Delta^{(\mathrm{P})}\hat{y}_i(\hat{x}_4) = \begin{cases} 2.02426\cdots, & \text{if } i = 6 \\ 1.59804\cdots, & \text{if } i = 7 \end{cases}$

Computing numerical solutions to the differential equations we find that $\hat{y}_9(x_5)$ satisfies

$$0.3806845 < \hat{y}_9(x_5) < 0.3806846.$$

The solutions exit $V^{(5)}(\delta)$ at the boundary $\hat{y}_0 = 0$. Thus the algorithm has finished. Therefore an a.a.s. upper bound on the domination number of a random 2-in 2-out digraph is $0.38069n$.

d	PathDominatingSet	Best Previous
2	$0.55283n$	$0.71379n$
3	$0.48912n$	$0.61594n$
4	$0.43531n$	$0.54383n$
5	$0.39262n$	$0.48840n$
6	$0.35813n$	$0.44433n$

Table 6.2.1: Asymptotically almost sure upper bounds on the 2-path domination number of a random d-in d-out digraph.

6.2 Analysing 2-Path Dominating Set Algorithms

We introduce a prioritised algorithm, called PathDominatingSet, that finds a 2-path dominating set of a random d-in d-out digraph. Applying the deprioritised approach to PathDominatingSet, we obtain a.a.s. upper bounds on the 2-path domination number as given in Table 6.2.1; previous upper bounds, from Theorem 2.2.8, are also given.

The algorithm PathDominatingSet is designed rather differently than DominatingSet. As PathDominatingSet proceeds, vertices are added to a set B. The operations are defined so that, after each operation, each edge is incident with at least one vertex in B. Thus, after each operation, every vertex with in-degree at least one and out-degree at least one is either in B or has both an in-neighbour and an out-neighbour in B. The algorithm finishes when there are no vertices of in-degree zero or out-degree zero; at which point B is a 2-path dominating set. Note that PathDominatingSet does not return a d-in d-out digraph. In fact, the fewer points exposed by PathDominatingSet, the better PathDominatingSet has performed. The digraph returned by PathDominatingSet can be completed to a random d-in d-out digraph by exposing the remaining free points.

6.2.1 The operations and their priorities

Each operation processes a vertex of degree pair (p, q) where $p = 0$ or $q = 0$. The type of an operation is determined by the degree pair of the processed vertex: so there are $2d + 1$ types of operations. An operation that processes a vertex of degree pair (p, q) proceeds as follows:

(i) a vertex v is degree pair (p, q) is selected uniformly at random;

(ii) if $p = 0$, then

 (a) a free in-point associated with v is exposed to obtain an in-neighbour w_1,

 (b) the vertex w_1 is added to B, and

 (c) the free points associated with w_1 are exposed;

(iii) if $q = 0$, then

 (a) a free out-point associated with v is exposed to obtain an out-neighbour w_2,

 (b) the vertex w_2 is added to B, and

 (c) the free points associated with w_2 are exposed.

This completes the definition of the operation.

We define the priorities of the operation types by also defining types for degree pairs. In particular, the degree pair (p, q) has type $2(p + q) - \delta_{p>q}$ where $\delta_{p>q} = 1$ if $p > q$ and $\delta_{p>q} = 0$ if $p \leq q$. Then the type of an operation is the type of the degree pair of the vertex processed by that operation. An operation of type τ_1 has higher priority than an operation of type τ_2 if $\tau_1 > \tau_2$. We can now complete the definition of PathDominatingSet: the algorithm begins with the empty pairing and proceeds via a sequence of operations, until no operation may be performed; at each step, the type of the next operation is the type of highest priority that may be performed.

6.2.2 The differential equations

We now determine differential equations whose solution describes the deprioritised algorithm approximating PathDominatingSet. Again we require that d be fixed. The differential equations are determined using random variables $Z_{(i,j)}$, which count the vertices with degree pair (i,j), and $Z = |B|$. Later, when applying Theorem 5.4.1, we rename these random variables Y_i for $i = 0, \dots, (d+1)^2$. In particular, for (p,q) with $p = 0$ or $q = 0$, we let $Z_{(p,q)} = Y_i$ where (p,q) has type i, as described in Section 6.2.1. We also let $Z = Y_{(d+1)^2}$. The random variables $Z_{(p,q)}$ for $p, q > 0$ can be renamed in any way.

We start by determining functions that approximate the expected change in the random variables due to a single operation of PathDominatingSet. So in this section (Section 6.2.2), the underlying random process is defined by the prioritised algorithm PathDominatingSet. As in Section 6.1.2, we determine functions $f^{(r)}_{(i,j)}$ and $f^{(r)}$, for $0 \le i, j \le d$ and $r \in \{0, \dots, 2d\}$, such that

$$f^{(r)}_{(i,j)}(t/n, Z_{(0,0)}(t)/n, \dots, Z_{(d,d)}(t)/n, Z(t)/n) + o(1)$$

is the expected change in $Z_{(i,j)}$ due to an operation of type r at time t, and

$$f^{(r)}(t/n, Z_{(0,0)}(t)/n, \dots, Z_{(d,d)}(t)/n, Z(t)/n) + o(1)$$

is the expected change in Z due to an operation of type r at time t.

The analysis is similar to that of DominatingSet; the main difference is that there are no rems. We again use the functions $\mathbb{P}_{\text{in}}(w \in V_{(i,j)})$, $\mathbb{P}_{\text{out}}(w \in V_{(i,j)})$, $\text{In}_{(i,j)}$, and $\text{Out}_{(i,j)}$, as defined in Section 6.1.2, to define $f^{(r)}_{(i,j)}$ and $f^{(r)}$.

Now consider an operation that processes a vertex v of degree pair (p,q). If $p = 0$, then we expose a free in-point associated with v to obtain a new in-neighbour w_1. Let F_{in} and F_{out} be the number of free in-points and free out-points associated with w_1 respectively, before the edge from w_1 to v is exposed. Then the expected change in $Z_{(i,j)}$ due to exposing the free points associated with w_1 is $\Psi^{\text{in}}_{(i,j)} + o(1)$ where

$$\Psi^{\text{in}}_{(i,j)} = \delta_{i,d}\delta_{j,d} - \mathbb{P}_{\text{out}}(w_1 \in V_{(i,j)}) + \mathbb{E}(F_{\text{in}})\text{Out}_{(i,j)} + \mathbb{E}(F_{\text{out}} - 1)\text{In}_{(i,j)}$$

and

$$\mathbb{E}(F_{\text{in}}) = \sum_{r=0}^{d}\sum_{s=0}^{d}(d-r)\mathbb{P}_{\text{out}}(w_1 \in V_{(r,s)})$$

and

$$\mathbb{E}(F_{\text{out}} - 1) = \sum_{r=0}^{d}\sum_{s=0}^{d}(d-s-1)\mathbb{P}_{\text{out}}(w_1 \in V_{(r,s)}).$$

Similarly, if $q = 0$, then we expose an out-point associated with v to obtain a new out-neighbour w_2. The expected change in $Z_{(i,j)}$ due to exposing the free points associated with w_2 is $\Psi^{\text{out}}_{(i,j)} + o(1)$ where

$$\begin{aligned}\Psi^{\text{out}}_{(i,j)} &= \delta_{i,d}\delta_{j,d} - \mathbb{P}_{\text{in}}(w_2 \in V_{(i,j)}) \\ &\quad + \left(\sum_{r=0}^{d}\sum_{s=0}^{d}(d-r-1)\mathbb{P}_{\text{in}}(w_2 \in V_{(r,s)})\right)\text{Out}_{(i,j)} \\ &\quad + \left(\sum_{r=0}^{d}\sum_{s=0}^{d}(d-s)\mathbb{P}_{\text{in}}(w_2 \in V_{(r,s)})\right)\text{In}_{(i,j)}.\end{aligned}$$

The change in $Z_{(i,j)}$ from exposing the free points associated with v is

$$\begin{cases} \delta_{i,1}\delta_{j,1} - \delta_{i,0}\delta_{j,0} & \text{if } p = q = 0, \\ \delta_{i,1}\delta_{j,q} - \delta_{i,0}\delta_{j,q} & \text{if } p = 0 \text{ and } q > 0, \\ \delta_{i,p}\delta_{j,1} - \delta_{i,p}\delta_{j,0} & \text{if } p > 0 \text{ and } q = 0. \end{cases}$$

Now consider an operation of type r that processes a vertex of degree pair (p, q). Then expected change in $Z_{(i,j)}$ due to an operation of type r is $\text{Op}r_{(i,j)} + o(1)$ where

$$\text{Op}r_{(i,j)} = \begin{cases} \delta_{i,1}\delta_{j,1} - \delta_{i,0}\delta_{j,0} + \Psi^{\text{in}}_{(i,j)} + \Psi^{\text{out}}_{(i,j)} & \text{if } p = q = 0, \\ \delta_{i,1}\delta_{j,q} - \delta_{i,0}\delta_{j,q} + \Psi^{\text{in}}_{(i,j)} & \text{if } p = 0 \text{ and } q > 0, \\ \delta_{i,p}\delta_{j,1} - \delta_{i,p}\delta_{j,0} + \Psi^{\text{out}}_{(i,j)} & \text{if } p > 0 \text{ and } q = 0. \end{cases}$$

The expected change in Z due to an operation of type r is $\text{dom}_r + o(1)$ where

$$\text{dom}_r = \delta_{p,0} + \delta_{q,0}.$$

The Functions $f^{(r)}_{(i,j)}$ and $f^{(r)}$

The functions $f^{(r)}_{(i,j)}$ and $f^{(r)}$ are obtained by scaling the random variables $Z_{(i,j)}$ and P. In particular, we set $Z_{(i,j)}(t) = nz_{(i,j)}(t/n)$ and $Z(t) = nz(t/n)$ and then write

$\mathrm{Op}r_{(i,j)}$ and dom_r in terms of $z_{(i,j)}$ and z. We use the scaled variables y_i such that $Y_i(t) = ny_i(t/n)$ (for $i = 0, \ldots, (d+1)^2$) when applying the deprioritised approach.

The differential equations

The differential equations $dz_{(i,j)}/dx$ and dz/dx are obtained using the theory of Chapter 4, as described for DominatingSet. Recall that now the underlying random process $\{G_t\}$ is defined by the deprioritised algorithm based on PathDominatingSet. Again we have $dz_{(i,j)}/dx$ approximating $\mathbb{E}(Z_{(i,j)}(t+1) - Z_{(i,j)}(t) \,|\, G_0, \ldots, G_t)$ while dz/dx approximating $\mathbb{E}(Z(t+1) - Z(t) \,|\, G_0, \ldots, G_t)$. As in Section 6.1, it is more convenient to solve the differential equations $dz_{(i,j)}/dz$.

6.2.3 Applying the deprioritised approach

We apply the deprioritised approach to PathDominatingSet, as described in Chapter 4 and Chapter 5. So in this section we show that PathDominatingSet is a deprioritisable algorithm and determine the irreducible type sets for each phase. We then apply Theorem 5.4.1 for $d = 2$.

Recall the definition of deprioritisable algorithms from Definition 4.1.1. From the definition of PathDominatingSet, we see that parts (i), (ii), and (iii) are satisfied. Part (iii) is satisfied (with the polynomial $H = s$ and integers $k = 2d$ and $m = (d+1)^2$) as described in the previous section. The number of edges exposed by each operation is bounded, so part (v) is satisfied. Part (vi) is satisfied with $M = 1$, as the random variables count vertices. Finally, we have $Y_0(0) = n$ and $Y_i(0) = 0$ for $i \geq 1$, so part (vii) is satisfied. Therefore PathDominatingSet is a deprioritisable algorithm.

6.2.4 Determining the irreducible type sets

The irreducible type sets are determined using the DPPD method described in Section 4.2.4. Recall the definition of $B_{(i,j)}$-determined from Section 4.2.4. It is more convenient to use the variables $z_{(p,q)}$. Recall that the functions ν and dp are used to translate between the variables $z_{(p,q)}$ and y_i. We define $\nu(p,q) = i$ where i is the type of (p,q) (as in Section 6.2.1) and $\mathrm{dp}(\nu(p,q)) = (p,q)$. Now, for $(i,j) \neq (d,d)$ and $(i,j) \neq (1,1)$, we have

$$B_{(i,j)} = \{(i-1,j),(i,j-1)\} \cap \{(p,q) \,:\, 0 \leq p,q \leq d\};$$

when $(i,j) = (1,1)$ we have $B_{(1,1)} = \{(0,1),(1,0),(0,0)\}$.

As for DominatingSet, for (i,j) corresponding to an operation type, the function $f^{(r)}_{(i,j)}$ is $B_{(i,j)}$-determined if

(a) every term in the numerator of $f^{(r)}_{(i,j)} + \delta_{\nu(i,j),r}$ contains at least one variable in the set $\{z_{(i,j)}, z_{(i-1,j)}, z_{(i,j-1)}\}$, and

(b) for all $(p,q) \in B_{(i,j)}$, there exists a term of the numerator of $f^{(r)}_{(i,j)} + \delta_{\nu(i,j),r}$ containing the variable $z_{(p,q)}$ but no variables in the set

$$\{z_{\mathrm{dp}(j)} \,:\, j = \tau+1,\ldots,k\}.$$

Now fix (i,j) with $i = 0$ or $j = 0$. Notice that every term of the numerators of $\mathrm{In}_{(i,j)}$ and $\mathrm{Out}_{(i,j)}$ involves $z_{(i,j)}$ or $z_{(i-1,j)}$ or $z_{(i,j-1)}$. Therefore every term of the numerator of $\Psi^{\mathrm{in}}_{(i,j)}$, $\Psi^{\mathrm{in}}_{(i,j)}$, and $f^{(r)}_{(i,j)} + \delta_{\nu(i,j),r}$ involves $z_{(i,j)}$ or $z_{(i-1,j)}$ or $z_{(i,j-1)}$. So part (a) above is satisfied. It is also clear from the definition of $f^{(r)}_{(i,j)} + \delta_{\nu(i,j),r}$ that for $(p,q) \in B_{(i,j)}$, the numerator of $f^{(r)}_{(i,j)} + \delta_{\nu(i,j),r}$ contains a term involving only the variables $z_{(p,q)}$ and $z_{(0,0)}$. Hence part (b) above is also satisfied. Therefore $f^{(r)}_{\mathrm{dp}(i)}$ is $B_{\mathrm{dp}(i)}$-determined for all $i \in \{0,\ldots,k\}$ and $r \in \{0,\ldots,k\}$. Therefore, we may apply Lemma 4.2.11 for all phase types.

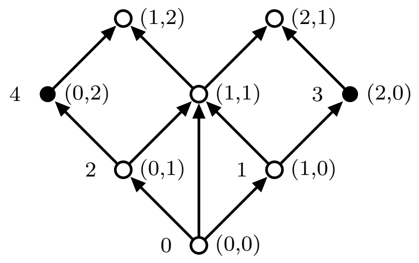


Figure 6.2.1: The coloured DPPD for a phase of type 2 of PathDominatingSet.

Phase Type	Operation Types
0	0,1,2
1	1,2,3
2	2,3,4
3	3,4
4	4

Table 6.2.2: The operations types that may occur during a phase of a given type.

We determine the irreducible type sets for PathDominatingSet for $d = 2$ only. The coloured DPPD for a phase of type 2 is shown in Figure 6.2.1. By Lemma 4.2.11, we have $\mathcal{M}^{(2)} = \{2, 3, 4\}$. Table 6.2.2 gives the irreducible type sets for each phase type when $d = 2$.

To apply Theorem 5.4.1, we must choose the type of the first phase. Consider the prioritised algorithm PathDominatingSet. The first operation of PathDominatingSet has type 0. So after the first operation, a.a.s. there are vertices of degree pairs $(2, 2)$, $(1, 1)$, $(1, 0)$, and $(0, 1)$ only. Therefore the type of the second operation and the first phase of PathDominatingSet is 2. So we take the type of the first phase of the deprioritised algorithm based on PathDominatingSet to be 2.

6.2.5 Numerical analysis for $d = 2$

We now use Theorem 5.4.1 to analyse the deprioritised algorithm based on PathDominatingSet for $d = 2$. For this deprioritised algorithm, the types of the phases decrease and the last phase has type 0. As the Independent Types Property holds for all phase types, we are able to use the alternative hypotheses (A3)–(D3) for phases two and three. We omit the analyses for larger d, as they do not demonstrate anything new. As before we solve the differential equations (non-rigorously) using the fourth order Runge-Kutta method. The numerical solutions are used to check the hypotheses of Theorem 5.4.1. The numerical solutions are extended as far as possible while the hypotheses are satisfied for some $\delta > 0$. We then take δ sufficiently small so as to include each point of the numerical solutions.

Before considering the hypotheses of Theorem 5.4.1, we finish the definition of the random variables Y_i (for $i = 0, \ldots, (d+1)^2$). Recall that for $i = 0, \ldots, 2d$, we have $Y_i = Z_{(p,q)}$ where (p, q) has type i; in particular,

$$Y_0 = Z_{(0,0)},\ Y_1 = Z_{(1,0)},\ Y_2 = Z_{(0,1)},\ Y_3 = Z_{(2,0)},\quad \text{and } Y_4 = Z_{(0,2)}.$$

We also define

$$Y_5 = Z_{(1,1)},\ Y_6 = Z_{(2,1)},\ Y_7 = Z_{(1,2)},\ Y_8 = Z_{(2,2)},\ \text{and } Y_9 = Z.$$

We now check the hypotheses of Theorem 5.4.1. First we show that the initial conditions $(0, \hat{y}_0(0), \ldots, \hat{y}_9(0))$ lie in $D(\delta)$. Recall that

$$\hat{y}_0(0) = \lim_{n\to\infty} Y_0(0)/n = 1 \quad \text{and} \quad \hat{y}_i(0) = \lim_{n\to\infty} Y_i(0)/n = 0 \quad (\text{for } i = 1, \ldots, 9).$$

Since the initial conditions are known exactly, the values of the functions of $\mathcal{B}^{(2)}(\delta)$ are calculated exactly for phase one. Again we have $H(0, \hat{y}_0(0), \ldots, \hat{y}_9(0)) = 2$ and so

$$(0, \hat{y}_0(0), \ldots, \hat{y}_9(0)) \in D(\delta).$$

We start the analysis with a phase of type 2.

Checking hypotheses for phase 1 of type 2

Hypothesis (A) To satisfy hypothesis (A) we need to show that

$$(0, \hat{y}_0(0), \ldots, \hat{y}_9(0))$$

lies in the closure of $V^{(2)}(\delta)$. Above we showed that

$$(0, \hat{y}_0(0), \ldots, \hat{y}_9(0)) \in D(\delta),$$

so it just remains to show that the functions

$$\det \mathcal{C}^{(2)} - \delta,\ \hat{y}_2,\ q_b^{(2)} \text{ for } b = 1, 2, 3, \text{ and } E_b^{(2)} \text{ for } b = 2, 3$$

are non-negative at $x = 0$. Calculating the values of these functions numerically at $(0, \hat{y}_0(0), \ldots, \hat{y}_9(0)) \in D(\delta)$ we find

$$\det \mathcal{C}^{(2)} - \delta = 1,\ \hat{y}_2 = 0,$$
$$q_b^{(2)} = \delta_{b,1} \text{ for } b = 1, 2, 3 \text{ and } E_b^{(2)} = 1 \text{ for } b = 2, 3.$$

Therefore hypothesis (A) is satisfied.

Hypothesis (B) We need to show that each function in $\widehat{\mathcal{B}}^{(2)}(\delta)$ has positive growth at x_0 during a phase of type 2. Any function that is positive has positive growth. Thus we only need to consider the functions $\hat{y}_2$ and $q_b^{(2)}$ for $b = 2, 3$. Using automatic differentiation [38] we calculate the derivatives of these functions and find

$$\Delta^{(2)} \hat{y}_2(x_0) = 1 \quad \text{and} \quad \Delta^{(2)} q_b^{(2)}(x_0) = \begin{cases} 0.5, & \text{if } b = 2 \\ 1, & \text{if } b = 3 \end{cases}$$

Therefore hypothesis (B) is satisfied.

Hypothesis (C) We must show that $\hat{y}_i$ (for $i = 2, 3, 4$) has positive growth at x_0 during a preprocessing subphase. Since $\hat{y}_i(x_0) = 0$ for $i = 2, 3, 4$, we must consider the derivatives of $\hat{y}_i$ (as calculated in a preprocessing subphase) for $i = 2, 3, 4$. We have

$$\Delta^{(\mathrm{P})} \hat{y}_2(x_0) = 3,\ \Delta^{(\mathrm{P})} \hat{y}_i(x_0) = 0 \text{ for } i = 3, 4$$

and

$$(\Delta^{(\mathrm{P})})^2\hat{y}_i(x_0) = 4.5 \text{ for } i = 3, 4.$$

Therefore hypothesis (C) is satisfied.

Hypothesis (D) We must show that the functions $\det \mathcal{C}^{(2)} - \delta$ and $q_b^{(2)}$ (for $b = 1, 2, 3$) have positive growth at x_0 during a preprocessing subphase. We have already shown that $\det \mathcal{C}^{(2)} - \delta$ and $q_1^{(2)}$ are positive at x_0 and so have positive growth. We consider the derivatives of $q_b^{(2)}$ at x_0 for $b = 2, 3$:

$$\Delta^{(\mathrm{P})}q_2^{(2)}(x_0) = 1.5 \text{ and } \Delta^{(\mathrm{P})}q_3^{(2)}(x_0) = 3.$$

Therefore hypothesis (D) is satisfied.

Hypothesis (E) For hypothesis (A), we showed that $E_b^{(2)}(x_0)$ is positive for $b = 2, 3$. Therefore hypothesis (E) is satisfied.

Hypothesis (F) Each function in $\widehat{\mathcal{B}}^{(2)}(\delta)$ has positive growth of order 0 or 1, as shown for hypothesis (B). For hypotheses (C) and (D), we showed that the functions with positive growth of order 1 during a phase of type 2, also have positive growth of order 1 during the preprocessing subphase. Thus hypothesis (F) is satisfied by part (ii) of Lemma 5.3.4.

Computing numerical solutions to the differential equations we find that $\hat{y}_9(x_1)$ satisfies

$$0.4030707 < \hat{y}_9(x_1) < 0.4030708.$$

The solutions exit $V^{(2)}(\delta)$ at the boundary $\hat{y}_2 = q_3^{(2)} = 0$ (by Corollary 5.5.3). Thus there is a next phase. Let $\hat{x}_1$ be such that $\hat{y}_9(\hat{x}_1) = 0.4030707$. Then we have $\hat{y}_1(\hat{x}_1) = 0.061195\cdots$, so phase two has type 1. We check the hypotheses for phase two at $\hat{x}_1$ instead of at x_1.

As we showed earlier, the Independent Types Property holds for a phase of type 1. Therefore, by Lemma 5.5.8, we satisfy hypotheses (A)–(F) by satisfying hypotheses (A3)–(D3).

Checking hypotheses for phase 2 of type 1

Hypothesis (A3) As mentioned above.

Hypothesis (B3)

(i) $\det \mathcal{C}^{(1)}(\hat{x}_1) = 0.71401\cdots$

(ii) $E_2^{(1)}(\hat{x}_1) = 0.53792\cdots$

Hypothesis (C3)

- $$q_b^{(1)}(\hat{x}_1) = \begin{cases} 0.53792\cdots, & \text{if } b = 1 \\ 0.09610\cdots, & \text{if } b = 2 \\ 0.07998\cdots, & \text{if } b = 3 \end{cases}$$

Hypothesis (D3)

- $\Delta^{(\mathrm{P})}\hat{y}_2(\hat{x}_1) = 0.47819\cdots$ and $\Delta^{(\mathrm{P})}\hat{y}_3(\hat{x}_1) = 0.17173\cdots$

Computing numerical solutions to the differential equations we find that $\hat{y}_9(x_2)$ satisfies

$$0.4858464 < \hat{y}_9(x_2) < 0.4858465.$$

The solutions exit $V^{(3)}(\delta)$ at the boundary $\hat{y}_1 = q_3^{(1)} = 0$ (by Corollary 5.5.3). Thus there is a next phase. Let $\hat{x}_2$ be such that $\hat{y}_9(\hat{x}_2) = 0.4858464$. Then we have $\hat{y}_0(\hat{x_2}) = 0.04454\cdots$, so phase three has type 0. We check the hypotheses for phase three at $\hat{x}_2$ instead of at x_2.

As we showed earlier, the Independent Types Property holds for a phase of type 0. Therefore, by Lemma 5.5.8, we satisfy hypotheses (A)–(F) by satisfying hypotheses (A3)–(D3).

Checking hypotheses for phase 3 of type 0

Hypothesis (A3) As mentioned above.

Hypothesis (B3)

(i) $\det \mathcal{C}^{(0)}(\hat{x}_2) = 1.04690\cdots$

(ii) $E_2^{(0)}(\hat{x}_2) = 0.62187\cdots$

Hypothesis (C3)

- $q_b^{(0)}(\hat{x}_2) = \begin{cases} 0.62187\cdots, & \text{if } b = 1 \\ 0.21251\cdots, & \text{if } b = 2, 3 \end{cases}$

Hypothesis (D3)

- $\Delta^{(\mathrm{P})}\hat{y}_1(\hat{x}_2) = \Delta^{(\mathrm{P})}\hat{y}_2(\hat{x}_2) = 0.25469\cdots$

Computing numerical solutions to the differential equations we find that $\hat{y}_9(x_3)$ satisfies

$$0.5528359 < \hat{y}_9(x_3) < 0.5528360.$$

The solutions exit $V^{(3)}(\delta)$ at the boundary $\hat{y}_0 = 0$. Thus the algorithm has finished. Therefore an a.a.s. upper bound on the domination number of a random 2-in 2-out digraph is $0.55283n$.

Chapter 7

Conclusion

Prioritised algorithms, of the type considered in this thesis, have been applied to many combinatorial problems. However, such algorithms are difficult to analyse. The deprioritised approach has proved to be an effective alternative. In this thesis we have extended earlier work by Wormald [61], so that the deprioritised approach may be applied to a larger class of prioritised algorithms: for example, algorithms for random d-in d-out digraphs (where d is fixed). Of course, further development is still possible and we now consider some possibilities for further work.

Some of the work in this thesis is specific to designing algorithms based on prioritised algorithms. For example, determining which types of operations occur during a phase and the expected proportions in which they occur. However we may also be interested in algorithms that choose the types of operations to perform using a probability distribution more generally. For such algorithms (unprioritised algorithms, perhaps?) the type distribution would not come from a prioritised algorithm but would be defined in some other way. The work in this thesis may be helpful in analysing unprioritised algorithms.

To analyse a deprioritised algorithm, as described by Theorem 5.4.1, a large number of conditions must be checked. Although automatic differentiation makes checking these conditions easy, it would be worthwhile to reduce the number of conditions.

Indeed, the conditions seem rather artificial, as they do not directly relate to the algorithms being analysed; so we would like to know whether or not they are necessary. Recently, Shi and Wormald [55] have introduced a method that may allow us to prove Theorem 5.4.1 assuming fewer hypotheses. Alternatively, there may be simple global conditions on a prioritised algorithm which ensure that the changes of phase of the corresponding deprioritised algorithm proceed smoothly.

Most of the hypotheses to Theorem 5.4.1 concern the change of phase. These hypotheses are required because the type distribution is discontinuous at the changes of phase. Using an idea of Shi and Wormald, we could modify our type distribution with a smoothing parameter so that the type distribution changes continuously during the execution of the algorithm. Letting the parameter tend to zero, the modified type distribution should approach the original type distribution. Thus the functions $\hat{y}_i$ describe the behaviour of the deprioritised algorithm. Such an approach should yield a theorem that is easier to apply than Theorem 5.4.1.

When developing the theory presented in this thesis, there were a number of choices regarding the hypotheses and definitions. For example, the functions $\hat{y}_i$ which describe the random variables of the deprioritised algorithm were defined using $\mathcal{M}^{(\tau)}$-clutches, where $\mathcal{M}^{(\tau)}$ are the irreducible type sets. However, we could also have defined $\hat{y}_i$ using $\{\tau, \dots, k\}$-clutches, provided we slightly change the hypotheses. In particular, we would need to allow the functions $p_r^{(\tau)}$ defining the type distribution to be zero throughout a phase; this could be shown using the results of Section 4.2.2 or with ad hoc methods. Another restriction we made is that the change in the random variables due to an operation must be bounded by a constant. This restriction could be eased to allow the change in the random variables to be bounded by a function that is $o(n)$. Indeed, earlier analyses of prioritised algorithms allowed for this possibility [60]. There are many possible variations of Theorem 5.4.1; we give the version that we have found most useful.

Finally we ask the question: do deprioritised algorithms perform better, worse, or the same as prioritised algorithms? In the (few) cases when both a prioritised algorithm and the corresponding deprioritised algorithm have been analysed, the results have been the same [30, 31]. So it seems that in using the deprioritised approach we obtain a simpler analysis which leads to results as good as those obtained by analysing prioritised algorithms.

Symbol Index

$A^{(\tau)}$	The domain describing a phase of type τ, 72
$\mathcal{B}^{(\tau_j)}(\delta)$	The set of functions defining the domain on which phase j of the deprioritised algorithm is analysed, 86
$\widehat{\mathcal{B}}^{(\tau_j)}(\delta)$	The subset of functions of $\mathcal{B}^{(\tau_j)}(\delta)$ that may have positive growth of positive order, 86
$\mathcal{C}(t)$	The clutch matrix of an $\mathcal{O}$-clutch at step t, 52
$\mathcal{C}^{(\tau)}(t)$	The clutch matrix for $\mathcal{M}^{(\tau)}$-clutch at step t during a phase of type τ, 56
deg-pair(u)	The degree-pair of the vertex u, 17
$\Delta^{(\mathrm{P})}$	The differential operator for a preprocessing subphase, 87
$\Delta^{(\tau)}$	The differential operator for a phase of type τ, 87
$\mathcal{DG}_{n,d}$	The model of random d-in d-out digraphs, 28
$DG(P)$	The multi-digraph obtained from the pairing P, 30
dp	A function relating variables to degree pairs, 67
$\mathcal{DP}_{n,d}$	The pairing model for random d-in d-out digraphs, 30
$\dfrac{dy_i}{dx}$	The differential equations whose solutions describe the scaled random variables of the deprioritised algorithm, 71
$\mathbb{E}(\cdot)$	The expectation of a random variable, 24
$E_b^{(\tau)}$	Functions relating to $\Phi_b^{(\tau)}(t)$, 72

ϵ_j	The scaled length of the preprocessing subphase of phase j, 82
$f_{i,r}$	Functions describing the expected change in the random variables due to an operation, 46
$f_{i,r}(t)$	The value of the function $f_{i,r}$ at step t of an algorithm, 48
$f_{i,r}(x)$	The value of the function $f_{i,r}$ in terms of some continuous functions, 48
$\overleftarrow{\gamma}(G)$	The absorption number of the digraph G, 20
$\overrightarrow{\gamma}(G)$	The domination number of the digraph G, 18
$\overrightarrow{\gamma}_{\text{path}}(G)$	The 2-path domination number of the digraph G, 20
$g_{i,r}$	The numerator of the function $f_{i,r}$, 46
H	A function for which H^{ℓ_i} is the denominator of $f_{i,r}$, 46
in-deg(u)	The in-degree of the vertex u, 17
ix(r, S)	The index of r in S, 60
K	The number of phases of the deprioritised algorithm, 73
$L^{(\tau)}(\delta)$	A domain on which the differential equations are Lipschitz, 72
$\mathcal{L}_h(x, \epsilon_j)$	A lower bound on the function h during the preprocessing subphase, 91
M	An upper bound on the scaled random variables, 47
$\mathcal{M}^{(\tau)}$	The irreducible type set for a phase of type τ, 59
$N_{\text{in}}(u)$	The set of in-neighbours of the vertex u, 16
$N_{\text{out}}(u)$	The set of out-neighbours of the vertex u, 16
ν	A function relating degree pairs to variables, 67

$o(c_n)$	The set of functions which divided by c_n are asymptotically zero, 18
$O(c_n)$	The set of functions asymptotically bounded above by c_n, 18
$\Omega(c_n)$	The set of functions asymptotically bounded below by c_n, 18
out-deg(u)	The out-degree of the vertex u, 17
$\mathbb{P}(\cdot)$	The probability of an event, 24
$\Phi_b(t)$	Functions analogous to the expected change in the random variables due to a subclutch, 53
$\Phi_b^{(\tau)}(t)$	Functions analogous to the expected change in the random variables due to a subclutch of an $\mathcal{M}^{(\tau)}$-clutch, 56
$p_r(t)$	Functions related to the expected proportions of operation types, 53
$p_r^{(\tau)}(t)$	Functions which are used to define the type distribution for a phase of type τ at step t, 56
$q_b^{(\tau)}$	Functions relating to $p_r^{(\tau)}$, 72
$D(\delta)$	A domain on which the functions $f_{i,r}$ are Lipschitz, 48
t_j	The first step of the preprocessing subphase of phase j, 82
t'_j	The first step of the main subphase of phase j, 82
$V^{(\tau)}(\delta)$	The domain on which we approximate a phase of type τ, 72
W_{δ,ϵ_j}	The domain on which the preprocessing subphase of phase j is analysed, 82
$Y_0, \ldots, Y_m$	Random variables defined for a deprioritisable algorithm, 46
$\hat{y}_i$	Functions describing the scaled random variables of the de-prioritised algorithm, 73

$y_i^{(j)}$	Functions describing the scaled random variables of the deprioritised algorithm during phase j, 73
$Y_i(t)$	The value of a random variable at step t, 47
$y_i^{(\mathrm{p})}$	Functions approximating the scaled random variables during a preprocessing subphase, 84
$z_i^{(j)}$	Functions obtained by applying Theorem 5.1.1 to the main subphase of phase j, 85
$z_i^{(\mathrm{p})}$	Functions obtained from applying Theorem 5.1.1 to the preprocessing subphase, 83

Bibliography

[1] D. Achlioptas. Setting 2 variables at a time yields a new lower bound for random 3-SAT. In *Proceedings of the 32nd ACM Symposium on Theory of Computing (STOC32)*, pages 28–37. ACM, New York, NY, 2000.

[2] D. Achlioptas. Lower Bounds for Random 3-SAT via Differential Equations. *Theoretical Computer Science*, 265:159–185, 2001.

[3] A.G. Akritas, E.K. Akritas, and G.I. Malaschonok. Various proofs of Sylvester's (determinant) identity. *Mathematics and Computers in Simulation*, 42:585–593, 1996.

[4] R. Albert, A. Barabási, and H. Jeong. Scale-free characteristics of random networks: the topology of the world-wide web. *Physica A*, 281:69–77, 2000.

[5] N. Alon and J.H. Spencer. *The Probabilistic Method.* John Wiley & Sons, Inc., New York, NY, 1992.

[6] V.I. Arnautov. Estimation of the exterior stability number of a graph by means of the minimal degree of the vertices. *Prikl. Math. i Programmirovanie*, 11:3–8, 1974.

[7] J. Aronson, A. Frieze, and B.G. Pittel. Maximum matchings in sparse random graphs: Karp-Sipser revisited. *Random Structures and Algorithms*, 12:111–177, 1998.

[8] J. Bang-Jensen and G. Gutin. *Digraphs: Theory, Algorithms and Applications.* Springer Monographs in Mathematics. Springer-Verlag, New York, NY, 2000.

[9] R. Bar-Yehuda and U. Vishkin. Complexity of finding k-path-free dominating sets in graphs. *Information Processing Letters*, 14:228–232, 1982.

[10] A. Barkauskas and L. Host. Finding efficient dominating sets in oriented graphs. *Congressus Numerantium*, 98:27–32, 1993.

[11] M. Beis, W. Duckworth, and M. Zito. Packing vertices and edges in random regular graphs. *Random Structures and Algorithms*, 32:20–37, 2008.

[12] A. Békéssy, P. Békéssy, and J. Komlós. Asymptotic enumeration of regular matrices. *Studia Scientiarum Mathematicarum Hungarica*, 7:343–353, 1972.

[13] E.A. Bender and E.R. Canfield. The asymptotic number of non-negative integer matrices with given row and column sums. *Journal of Combinatorial Theory, Series A*, 24:296–307, 1978.

[14] C. Berge and M.P. Schützenberger. Juex de Nim et solutions. *Comptes Rendus de l'Académie des Sciences*, 242:1672–1674, 1956.

[15] T. Bohman and A. Frieze. Karp-Sipser on random graphs with a fixed degree sequence. *preprint.*

[16] T. Bohman, A. Frieze, and N.C. Wormald. Avoidance of a giant component in half the edge set of a random graph. *Random Structures and Algorithms*, 25:432–449, 2004.

[17] B. Bollobás. A probabilistic proof of an asymptotic formula for the number of labelled regular graphs. *European Journal of Combinatorics*, 1:311–316, 1980.

[18] W.E. Boyce and R.C. DiPrima. *Elementary Differential Equations and Boundary Value Problems.* John Wiley & Sons, Inc., New York, NY, 2001.

[19] M. Cardei, F. Dai, J. Wu, and S. Yang. Extended Dominating Set and Its Applications in Ad Hoc Networks Using Cooperative Communication. *IEEE Transactions on Parallel and Distributed Systems*, 17:851–864, 2006.

[20] M. Chao and J. Franco. Probabilistic analysis of two heuristics for the 3-satisfiability problem. *SIAM Journal on Computing*, 15:1106–1118, 1986.

[21] M. Chlebík and J. Chlebíková. Approximation hardness of dominating set problems. In *Algorithms—ESA 2004*, volume 3221 of *Lecture Notes in Computer Science*, pages 192–203. Springer, Berlin, 2004.

[22] C. Cooper and A. Frieze. A general model of web graphs. *Random Structures and Algorithms*, 22:311–335, 2003.

[23] C. Cooper, A. Frieze, and M. Molloy. Hamilton cycles in random regular digraphs. *Combinatorics, Probability & Computing*, 3:39–50, 1994.

[24] T. Corman, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms.* The MIT Press, Cambridge, MA, 2001.

[25] J. Diaz, M. Serna, and N.C. Wormald. Bounds on the bisection width for random d-regular graphs. *Theoretical Computer Science*, 382:120–130, 2007.

[26] R. Diestel. *Graph Theory.* Number 173 in Graduate Texts in Mathematics. Springer-Verlag, New York, NY, 1997.

[27] W. Duckworth. *Greedy algorithms and cubic graphs.* PhD thesis, University of Melbourne, 2001.

[28] W. Duckworth. Total Domination of Random Regular Graphs. *Australasian Journal of Combinatorics*, 33:279–289, 2005.

[29] W. Duckworth and B. Mans. Randomised greedy algorithms for finding small k-dominating sets of random regular graphs. *Random Structures and Algorithms*, 27:401–412, 2005.

[30] W. Duckworth and N.C. Wormald. Minimum independent dominating sets of random cubic graphs. *Random Structures & Algorithms*, 21:147–161, 2002.

[31] W. Duckworth and N.C. Wormald. On the independent domination number of random regular graphs. *Combinatorics, Probability and Computing*, 15:513–522, 2006.

[32] A. Frieze. Minimum Paths in Directed Graphs. *Operational Research Quarterly*, 28:339–346, 1977.

[33] A. Frieze and S. Suen. Analysis of two simple heuristics on a random instance of k-SAT. *Journal of Algorithms*, 20:312–355, 1996.

[34] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., New York, NY, 1979.

[35] J. Ghoshal, R. Laskar, and D. Pillone. *Domination in Graphs: Advanced Topics*, pages 401–437. Marcel Dekker Inc., New York, NY, 1998.

[36] R.L. Graham, D.E. Knuth, and O. Patashnik. *Concrete Mathematics: a foundation for computer science.* Addison-Wesley Publishing Company, Reading, MA, 1988.

[37] C. Greenhill, A. Ruciński, and N.C. Wormald. Random Hypergraph Processes with Degree Restrictions. *Graphs and Combinatorics*, 20:319–332, 2004.

[38] A. Griewank. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation.* Frontiers in Applied Mathematics. SIAM, Philadelphia, PA, 2000.

[39] F. Harary. *Graph Theory.* Addison-Wesley Publishing Company, Reading, MA, 1969.

[40] T.W. Haynes, S.T. Hedetniemi, and P.J. Slater, editors. *Domination in Graphs: Advanced Topics.* Marcel Dekker, Inc., New York, NY, 1998.

[41] T.W. Haynes, S.T. Hedetniemi, and P.J. Slater. *Fundamentals of Domination in Graphs.* Marcel Dekker, Inc., New York, NY, 1998.

[42] S. Howe. Dominating sets of random 2-in 2-out directed graphs. *Electronic Journal of Combinatorics*, 15:#R29 (electronic), 2008.

[43] W. Hurewicz. *Lectures on ordinary differential equations.* The MIT Press, Cambridge, MA, 1958.

[44] S. Janson. Random Regular Graphs: Asymptotic Distributions and Contiguity. *Combinatorics, Probability and Computing*, 4:369–405, 1995.

[45] A.C. Kaporis, L.M. Kirousis, and E.G. Lalas. The probabilistic analysis of a greedy satisfiability algorithm. *Random Structures and Algorithms*, 28:444–480, 2006.

[46] R.M. Karp and M. Sipser. Maximum Matchings in Sparse Regular Graphs. In *Proceedings of the 22nd IEEE Symposium on Foundations of Computer Science (FOCS'81)*, pages 364–375. IEEE Computer Society Press, Los Alamitos, CA, 1981.

[47] T.G. Kurtz. Solutions of Ordinary Differential Equations as Limits of Pure Jump Markov Processes. *Journal of Applied Probability*, 7:49–58, 1970.

[48] S. Kutten and D. Peleg. Fast Distributed Construction of Small k-Dominating Sets and Applications. *Journal of Algorithms*, 28:40–66, 1998.

[49] C. Lee. *On the domination number of a digraph.* PhD thesis, Michagon State University, 1994.

[50] B.D. McKay. Asymptotics for 0-1 matrices with prescribed line sums. In *Enumeration and Design*, pages 225–238. Academic Press, 1984.

[51] J. Von Neumann and O. Morgenstern. *Theory of Games and Ecomonic Behaviour.* Princeton University Press, Princeton, 1944.

[52] M.E.J. Newman, D.J. Watts, and S.H. Strogatz. Random graph models of social networks. *PNAS*, 99:2566–2572, 2002.

[53] R.J. Nowakowski, editor. *Games of no chance*, volume 29 of *Mathematical Sciences Research Institute Publications.* Cambridge University Press, Cambridge, 1996.

[54] C. Payan. Sur le nombre d'absorption d'un graphe simple. *Cahiers Centre Études Recherche Opér*, 17:307–317, 1975.

[55] L. Shi and N.C. Wormald. Colouring random 4-regular graphs. *preprint.*

[56] L. Shi and N.C. Wormald. Colouring random regular graphs. *preprint.*

[57] N.C. Wormald. *Some Problems in the Enumeration of Labelled Graphs.* PhD thesis, University of Newcastle, 1978.

[58] N.C. Wormald. Differential Equations for Random Processes and Random Graphs. *The Annals of Applied Probability*, 5:1217–1235, 1995.

[59] N.C. Wormald. Models of Random Regular graphs. In *Surveys in Combinatorics, 1999*, number 267 in London Mathematical Society Lecture Note Series, pages 239–298. Cambridge University Press, 1999.

[60] N.C. Wormald. The differential equation method for random graph processes and greedy algorithms. In *Lectures on Approximation and Randomized Algorithms*, pages 73–155. Polish Scientific Publishers PWN, Warsaw, 1999.

[61] N.C. Wormald. Analysis of greedy algorithms on graphs with bounded degrees. *Discrete Mathematics*, 273:235–260, 2003.

[62] M. Zito. Greedy algorithms for minimisation problems in random regular graphs. *Lecture Notes in Computer Science*, 2161:524–536, 2001.