Master Thesis

Random walk algorithms for SAT

Author(s):
Giurgiu, Andrei

Publication Date:
2009

Permanent Link:
https://doi.org/10.3929/ethz-a-005939758

Rights / License:
In Copyright - Non-Commercial Use Permitted
Random Walk Algorithms for SAT

Master Thesis
Andrei Giurgiu
September 2009

Advisors: Prof. Dr. Emo Welzl, Robin Moser
Department of Computer Science, ETH Zürich
Abstract

Schöning [14] discovered a simple local-search randomized algorithm for solving k-SAT, which still remains one of the fastest algorithms known so far. We recall the analysis of this algorithm using random walks on the integers, and then introduce a modified version and generalize the analysis so that it applies for the modification as well. Schöning’s algorithm chooses an assignment for the variables at random, and then, as long as it is not satisfying, picks a violated clause and flips the value of a variable from that clause. If after a linear number of iterations, a satisfying assignment is still not found, the procedure is restarted. The modification consists of randomly flipping each variable that appears in the chosen violated clause. In both Schöning’s algorithm and the modified version the part describing exactly which violated clause to select is left unspecified. We call that part selection rule. We produce worst-case sets of formulas which show that the analysis is tight, for two different selection rules. In the final chapter, we consider computationally unbounded selection rules that ensure a polynomial runtime of Schöning’s algorithm, and we prove that such selection rules exist for some classes of formulas. During our investigations, we also develop more general combinatorial tools which may prove useful for analyzing algorithms using random walks.
Contents

1 Introduction
  1.1 Motivation And Outline ........................................ 1
  1.2 Notation and Terminology ....................................... 2
  1.3 Markov Chains .................................................. 4
  1.4 Schöning’s Algorithm ............................................ 4
  1.5 Recent Results for SAT .......................................... 5
  1.6 The Analysis of Schöning’s Algorithm ......................... 6

2 A Variant of Schöning’s Algorithm
  2.1 The Algorithm .................................................. 13
  2.2 The Coupling Argument ......................................... 14
  2.3 Analysing a Particular Kind of Markov Chains ............... 15
  2.4 Schöning’s Algorithm as a Special Case ....................... 26
  2.5 Analysis of the Modified Algorithm ......................... 26
  2.6 Relation to Schöning’s algorithm ............................. 28

3 Worst-Case Analysis
  3.1 General Treatment ............................................... 31

iii
3.2 The “Devil” Selection Rule For Schöning’s Algorithm . . . . . . 38
3.3 The “Random” Selection Rule For Schöning’s Algorithm . . . 41
3.4 The “Devil” Selection Rule For the Modified Algorithm . . . 46
3.5 The “Random” Selection Rule For the Modified Algorithm . . 48

4 Selection Rules of Type “Angel” 55
4.1 More About Selection Rules . . . . . . . . . . . . . . . . . . . . 55
4.2 Types of Angels . . . . . . . . . . . . . . . . . . . . . . . . . . . 57
4.3 Relations Between Types of Angels . . . . . . . . . . . . . . . . 58
4.4 Angels For Some Restricted Classes of Formulas . . . . . . . . 64
   The Formulas Used For the Worst-case Scenario . . . . . . . . 64
   The Circular-XOR Formula . . . . . . . . . . . . . . . . . . . . . 67

5 Conclusions and Open Problems 73

A Probability Theory 75
A.1 Chernoff Bounds . . . . . . . . . . . . . . . . . . . . . . . . . . 75
A.2 Azuma’s Inequality . . . . . . . . . . . . . . . . . . . . . . . . . 76

Bibliography 77
Chapter 1

Introduction

1.1 Motivation And Outline

The problem of boolean formula satisfiability (SAT) is of central importance in Computer Science, being probably the most famous example for NP-completeness. The ability to easily reduce a huge variety of problems to SAT implies that efficient algorithms for SAT may be usable for other NP-hard problems as well. Moreover, SAT is not only extremely important from a theoretical point of view, but has countless direct practical applications as well, for example in scheduling or hardware design and verification, to name only a few. For this reason, in the last few decades there has been a constant search for faster and faster algorithms for solving satisfiability.

The last major improvement in this direction was made by Schöning \cite{14}, who proposed a simple randomized algorithm that finds a satisfying truth assignment for a 3-CNF formula $F$ with $n$ variables in $O(1.334^n)$. The proof of this relies on random walks, as we will see further in this chapter. It works by maintaining a current assignment, and each time flipping the value of one variable in a violated clause, in order to get closer to the satisfying assignment. If after polynomially many steps, an assignment has not been found, the current assignment is resampled, and the procedure repeated.

In the second chapter of this thesis, we will consider a modified version of Schöning’s algorithm, which is similar to the algorithm proposed by Robin Moser \cite{7} as a constructive version of Lovász Local Lemma. This means that for a formula for which each clause has variables in common with at most $2^{k-2}$ other clauses, we know that the algorithm finds a satisfying assignment in polynomial time. This is of course not the case if we consider general formulas, and our purpose here is to apply (roughly) the same analysis as is the case of Schöning’s algorithm for this modified version. The goal is to
1. Introduction

determine exactly how this modified algorithm behaves running on $k$-CNF formulas.

In the third chapter, we show that the upper bounds that we found for Schöning’s algorithm and the modified version are indeed tight. In both algorithms, the variables that get flipped appear always in a single clause that is violated by the current assignment. The algorithms do not specify which violated clause is chosen at each point since this is irrelevant for proving the upper bound, that is, even if an adversary is actively trying to keep the algorithm away from the satisfying assignment by always choosing the worst clauses. In this chapter we analyze such a “devilish” selection rule, and also a random selection rule, in which the violated clause is chosen at random. In both these cases, and for both algorithms, we prove that for some specific formulas, the algorithms are as slow as possible, and thereby that the analysis is tight.

In the last chapter we consider “beneficent” selection rules, which are such that at each point the algorithm picks a good clause, and “guides” the algorithm towards a satisfying assignment. If it manages to do so in polynomial time, then we call it an angel. We do not yet know whether there is such an angel that works for all formulas (the so-called universal angel). We define multiple types of angels and show that some of them are equivalent, and that in some cases they do not need to be exceedingly complex.

1.2 Notation and Terminology

This thesis will primarily follow the notation used in the lecture notes of Prof. Emo Welzl at ETH Zürich for the course “Boolean Satisfiability – Combinatorics and Algorithms” [15].

First of all, based on the universality of CNF (conjunctive normal form), we will consider logical formulas in CNF almost always throughout this material. In order to have a simplified representation for these formulas, we introduce the set notation.

Let $V$ be a finite set, consisting of $n$ boolean variables. The set $\overline{V}$ is the set of negations of the variables in $V$. For all variables $x \in V$, there is an element $\overline{x} \in \overline{V}$, which is the negation of $x$. Furthermore, this correspondence is bijective, so $\overline{V}$ has exactly $n$ elements. Moreover, $V$ and $\overline{V}$ need to be disjoint. The set $V \cup \overline{V}$ is called the set of literals, with elements of $V$ being positive literals and elements of $\overline{V}$ negative literals. We define negations for negative literals as well, so the negation of $\overline{x}$ will be $x$. 
Given a general CNF formula with the usual (logical) form

$$m \bigwedge_{i=1}^{k_i} \bigvee_{j=1}^{u_{i,j}}$$

where $u_{i,j}$ are literals, the same formula in the set notation will have the form

$$\{C_1, \ldots, C_m\}, \text{ where } C_i = \{u_{i,1}, \ldots, u_{i,k_i}\}.$$

In this notation, the clauses $\bigvee_{j=1}^{k_i} u_{i,j}$ are denoted by a set $C_i$ of the corresponding literals. The CNF formula is then represented by the set of $m$ clauses. The set of variables that appear inside $F$ as positive or negative literals is denoted by $\text{vbl}(F)$.

In the rest of this thesis, we will consider formulas with clauses of the same size $k$, which we will call $k$-CNF formulas.

Given a variable set $V$, an assignment $\alpha$ on $V$ is a mapping $V \to \{0,1\}$. The set of all assignments on $V$ will then be denoted by $\{0,1\}^V$. For $\overline{x} \in \overline{V}$, we define $\alpha(\overline{x}) = 1 - \alpha(x)$. An assignment satisfies a clause if at least one literal that appears in that clause maps to 1 under the assignment. An assignment satisfies a formula if it satisfies all its clauses. A formula is satisfiable if there exists an assignment that satisfies it.

We will extensively use the notion of Hamming distance between two assignments $\alpha$ and $\beta$ on $V$, defined as the number of variables in $V$ for which $\alpha$ and $\beta$ have different values:

$$d(\alpha, \beta) = \sum_{x \in V} |\alpha(x) - \beta(x)|.$$

Given a formula $F$ and an assignment $\alpha$ on $\text{vbl}(F)$, the set of violated clauses is denoted by $\text{vlt}(F, \alpha)$. If $F$ is satisfied by $\alpha$, then $\text{vlt}(F, \alpha)$ is naturally empty.

We will now introduce some notations that we will use in the analysis of specific $k$-CNF formulas. We denote by $1_V$ the assignment that maps 1 to all variables in $V$. Also, we define clause types as follows: the type of a clause is the number of positive literals that appear in that clause. The set of violated clauses of type $i$ of $F$ by the assignment $\alpha$ is denoted by $\text{vlt}_i(F, \alpha)$. 
1.3 Markov Chains

Markov chains are an important tool for randomized algorithms. We will use them extensively throughout this thesis, mostly in the form of random walks on the integers. Intuitively, we can think of a Markov chain as a variable that changes in at discrete time steps, the value at a certain time step depending only on the value at the previous time step.

**Definition 1.1** Let \( \{X_i\}_{i \geq 0} \) be a sequence of real-valued random variables. The sequence \( \{X_i\}_{i \geq 0} \) is a Markov chain if the following property, which we call the Markov property, holds for all \( i \geq 0 \):

\[
Pr[X_{i+1} = x_{i+1} \mid X_1 = x_1, \ldots, X_i = x_i] = Pr[X_{i+1} = x_{i+1} \mid X_i = x_i].
\]

The quantities \( Pr[X_{i+1} = x_{i+1} \mid X_i = x_i] \) will be referred to as transition probabilities.

In other words, the value of the chain at a certain time step depends only on the value at the previous time step. If additionally, the transition probabilities are invariant, then the Markov chain is time-homogeneous.

**Definition 1.2** A Markov chain is time-homogeneous, if for all \( i, i' \geq 0 \), and all \( a, b \) we have

\[
Pr[X_{i+1} = a \mid X_i = b] = Pr[X_{i'+1} = a \mid X_{i'} = b].
\]

Typically, we will use time-homogenous Markov chains to approximate the evolution of particular values throughout the execution of an algorithm.

1.4 Schöning’s Algorithm

We now present the randomized algorithm introduced by Schöning in \([14]\). Given a satisfiable \( k \)-CNF formula \( F \), it finds one satisfying assignment in expected time \( O \left( \left( \frac{2^{k-1}}{k} \right)^n \text{poly}(n) \right) \), where \( n \) is the number of variables in the formula. The algorithm simply consists of repeating function \( \text{Sch}(F, 3n) \) (see below) until a satisfying assignment is found. We can also use this algorithm to determine with arbitrarily small error probability whether a \( k \)-CNF formula is satisfiable or not.

As can be easily seen from the algorithm listing, the function \( \text{Sch}(F, t) \) picks an assignment at random and simply repeats for at most \( t \) times the following sequence of actions: it picks an unsatisfied clause (which is guaranteed
to exist if the current assignment is not satisfying), takes one variable at random from that clause, and flips the bit corresponding to that variable in the current assignment. If at any step, a satisfying assignment is found, the procedure terminates successfully. If after $t$ steps it has still encountered no satisfying assignment, the procedure finishes signalling this.

Note that we do not specify the way to select a violated clause at every step. For the algorithm to be fully described, there needs to be a rule according to which the violated clause is picked, which we call selection rule. In order to obtain the bounds in this chapter, the selection rule is irrelevant, since the result will hold for all such rules.

As we have mentioned before, this algorithm needs to be repeated for a large number of times in order to obtain a reasonable probability of success. In this section, we will prove an upper bound for the runtime of the algorithm for any $k$-CNF formula, and leave for a later chapter the discussion of whether this bound is tight. We state here the result that will be proven at the end of this introductory chapter.

**Theorem 1.3** In expectation, for a satisfiable $k$-CNF formula $F$, a satisfying assignment will be found after repeating $Sch(F, 3n)$ at most $\frac{3}{2} \left( \frac{2^{(k-1)} - 1}{k} \right)^n$ times. The probability that it does not find a satisfying assignment after $\frac{3T}{2} \left( \frac{2^{(k-1)} - 1}{k} \right)^n$ is at most $e^{-T}$.

**Procedure** $Sch(F, t)$

```plaintext
\begin{align*}
\alpha &\in \text{rand} \{0,1\}^{\text{vbl}(F)}; \\
\text{if } \alpha \text{ satisfies } F \text{ then} &\quad \text{return } (\text{true}, \alpha) \\
&\quad i = 0; \\
\text{while } i < t \text{ do} &\quad C \in \text{vlt}(F, \alpha) \text{ (use any selection rule);} \\
&\quad v \in \text{rand vbl } (C); \\
&\quad \alpha(v) \leftarrow 1 - \alpha(v); \\
&\quad \text{if } \alpha \text{ satisfies } F \text{ then} &\quad \text{return } (\text{true}, \alpha) \\
&\quad i \leftarrow i + 1; \\
&\quad \text{return } (\text{false}, \text{nil})
\end{align*}
```

1.5 Recent Results for SAT

Schöning’s algorithm is very similar to a randomized algorithm by Papadimitriou [8], which solved 2-SAT in expected quadratic time.
Prior to Schöning’s algorithm, the fastest solution to 3-SAT was the PPSZ algorithm due to Paturi et al. [9], which has a runtime of $O(1.447^n)$. This was a direct improvement from a previous algorithm also by Paturi et al. [10], which runs in $O(1.588^n)$ time. After Schöning’s algorithm surfaced, there have been successive improvements to $O(1.331^n)$ by Hofmeister et al. [3], $O(1.328^n)$ by Rolf [11], and to $O(1.324^n)$ by Iwama and Tamaki [4]. The last paper introduces an algorithm that is a combination of both the PPSZ algorithm and Schöning’s. The best result known so far is due to [12], who improved the result for 3-SAT to $O(1.322^n)$. Iwama and Tamaki [4], have also presented the best solution so far to 4-SAT, running in $O(1.474^n)$ time. For $k$-SAT, where $k > 4$, the current best solution is still given by the PPSZ algorithm [9].

Many of the algorithms mentioned so far have been derandomized, but as such they perform much worse than their randomized counterparts. The best deterministic algorithm to date is by Scheder [13], and runs in $O(1.465^n)$ for 3-CNF formulas.

### 1.6 The Analysis of Schöning’s Algorithm

Since the algorithm consists of independent runs of $\text{Sch}(F, 3n)$, we will first compute the probability of success for one single run. Also it is obvious that if $F$ is not satisfiable, then the procedure returns $false$.

In order to determine the probability of success, given that $F$ is satisfiable, we do the following thought experiment: we alter the algorithm in such a way that its behaviour stays the same, but it computes additional values that are easier to handle than the current assignment $\alpha$. The modified algorithm is given below as Procedure Sch-Coupling, and, of course, is not meant to be useful for any practical purposes, but is a mere tool that allows us to understand the mechanics of Schöning’s algorithm.

The only thing that we require is that for a satisfiable formula $F$, and for identical internal random bits, if we allow the same number of time steps $t$, the two algorithms perform the same number of iterations. Also, this extended algorithm is allowed to use any extra information, including the actual satisfying assignments for $F$, as long as the above condition is fulfilled. Since we assume that the formula $F$ is satisfiable, let $\alpha^*$ be one satisfying assignment, which is supplied as a parameter to the algorithm.

The main idea of this auxiliary algorithm is that it contains an internal variable $X$, which is changed at every step, using the same randomness source as the original algorithm. The main idea is that $X$ is always at least as large as the Hamming distance between the current assignment $\alpha$ and $\alpha^*$.
**Remark 1.4** Note that we can fix an arbitrary ordering of \( V \) in the beginning. Note that for any clause \( C \), there exists at least one literal in that clause which is satisfied by \( \alpha^* \). Clearly, this happens because \( C \) is satisfied by \( \alpha^* \). It makes sense then to define \( \xi(C, \alpha^*) \) as the variable corresponding to the smallest such literal.

---

**Procedure** Sch-Coupling(\( F, t, \alpha^* \))

\[
\begin{align*}
\alpha & \in \text{rand } \{0,1\}^{\text{vbl}(F)}; \\
X & = d(\alpha, \alpha^*); \\
\text{if } \alpha \text{ satisfies } F \text{ then} \\
\quad \text{return } (true, \alpha) \\
\text{else} \\
\quad i & = 0; \\
\text{while } i < t \text{ do} \\
\quad C & \in \text{vlt}(F, \alpha) \text{ (use any selection rule)}; \\
\quad v & \in \text{rand } \text{vbl}(C); \\
\quad \alpha(v) & \leftarrow 1 - \alpha(v); \\
\quad \text{if } v = \xi(C, \alpha^*) \text{ then} \\
\quad \quad X & \leftarrow X - 1 \\
\quad \text{else} \\
\quad \quad X & \leftarrow X + 1 \\
\quad \quad i & \leftarrow i + 1; \\
\text{if } \alpha \text{ satisfies } F \text{ then} \\
\quad \text{return } (true, \alpha) \\
\text{return } (false, \text{nil})
\end{align*}
\]

We have introduced the additional variable \( X \), but it can be easily seen that this does not change the overall flow of the program. Let \( X_i \) and \( a_i \) be the values of \( X \) and \( \alpha \) at each step \( i \). Furthermore, let \( i_{\text{max}} \) be the maximum value of \( i \) reached by the algorithm, or equivalently, the number of steps performed by the algorithm. This can clearly be less than \( t \), since the algorithm might find a satisfying assignment early and terminate. Then we have the following lemma.

**Lemma 1.5** For all possible runs of Sch-Coupling, and for all \( i \) such that \( 0 \leq i \leq i_{\text{max}} \), we have that \( d(\alpha_i, \alpha^*) \leq X_i \).

**Proof** We use induction. For \( i = 0 \), the claim holds, since \( X_0 = d(\alpha_0, \alpha^*) \) is guaranteed by the initialization of \( X \). For \( 0 < i \leq i_{\text{max}} \), we assume that the claim holds for \( i - 1 \). Clearly \( \alpha_{i-1} \) is not satisfying, and thus an unsatisfied clause \( C \) is picked.

If the random variable \( v \) is picked such that \( v = \xi(C, \alpha^*) \), then it must be the case that \( \alpha_{i-1}(v) = 1 - \alpha^*(v) \). This implies that \( d(\alpha_i, \alpha^*) = d(\alpha_{i-1}, \alpha^*) - 1 \), i.e. the Hamming distance decreases. Since \( X_i = X_{i-1} - 1 \), we have that the condition \( d(\alpha_i, \alpha^*) \leq X_i \) is maintained, thereby proving the lemma if \( v = \xi(C, \alpha^*) \).
1. Introduction

If \( v \neq \xi(C, a^*) \), we have that \( X_i = X_{i-1} + 1 \), and \( d(a_i, a^*) = d(a_{i-1}, a^*) \pm 1 \), from which the claim follows.

This lemma naturally entails the fact that if \( X_{i_{\text{max}}} = 0 \), it must be that the algorithm has already reached a satisfying assignment (namely \( a^* \)). It is clear then that \( \Pr [X_{i_{\text{max}}} = 0] \leq \Pr [\alpha_{i_{\text{max}}} \text{ satisfies } F] \).

We now make the following observation, by examining the probability with which \( X \) increases and decreases: If \( j > 0 \) and \( 0 \leq i < i_{\text{max}} \), then we have \( \Pr [X_{i+1} = j - 1 \mid X_i = j] = 1/k \), and \( \Pr [X_{i+1} = j + 1 \mid X_i = j] = 1 - 1/k \).

Also, it may be interesting to note that while \( d(\alpha, a^*) \) is upperbounded by \( n \), \( X \) is not. This happens because \( d(\alpha, a^*) \) may decrease when \( v \neq \xi(C, a^*) \), but in this situation \( X \) always increases. Since at each step where \( X \) is nonzero, there is always a probability that \( X \) increases, \( X \) can even grow arbitrarily large.

By extending these equations to all \( i \geq 0 \) (this implies introducing infinitely many extra random variables \( X_i \) for \( i > i_{\text{max}} \)), we obtain a Markov chain with transition probabilities as defined above. We introduce \( q = 1/k \) to ease the calculations, and many of the following results will hold for all \( 0 < q < 1/2 \).

To simplify notation, let \( Y \) be a random variable that represents the number of steps that are needed to reach 0, i.e. \( Y \) is the least non-negative integer \( i \) such that \( X_i = 0 \). In case such \( i \) does not exist, we set \( Y = \infty \). We are interested in the quantity \( \Pr [Y \leq 3n] \), i.e. the probability that after at most \( 3n \) steps the algorithm has reached a satisfying assignment.

In order to compute it, we use the approach from [15], which we reproduce here for completeness. In a later chapter, we will describe a different method to compute the results from the next two lemmas.

**Lemma 1.6** For all \( j \leq 0 \) and \( 0 < q < 1/2 \) we have that

\[
\Pr [Y < \infty \mid X_0 = j] = \left( \frac{q}{1 - q} \right)^j \quad \text{and}
\]

\[
E [Y \mid X_0 = j, Y < \infty] = \frac{j}{1 - 2q}.
\]

**Proof** We prove the first part first. Clearly the lemma holds for \( j = 0 \), so we will assume \( j > 0 \). We view \( X \) as a random walk on the non-negative integers. As we have already seen, at each step \( X \) will decrease with probability \( q \) and increase with probability \( 1 - q \). We consider the paths that start in \( j \) and reach 0, where we make exactly \( i \) steps that increase \( X \) (and thus \( i + j \)
steps that decrease \( X \), and reach 0 on the first time in the last step. The number of such paths, according to the ballot theorem (cf. \([2]\)) is \( \binom{2i+j}{i} \frac{j}{2i+j} \).

We weigh each path with its probability of occurrence
\[
P_q(i, j) = (1 - q)^i q^{i+j},
\]
and obtain the probability that the random walk eventually reaches 0 given that it starts in state \( j \):

\[
\Pr [Y < \infty \mid X_0 = j] = \sum_{i \leq 0} \binom{2i+j}{i} \frac{j}{2i+j} P_q(i, j) = q^j \sum_{i \leq 0} \binom{2i+j}{i} \frac{j}{2i+j} (q(1 - q))^i = q^j \left( B_2(q(1 - q)) \right)^j,
\]
where \( B_2(z) \) is the generalized binomial series given by

\[
B_2(z) = \sum_i \binom{2i+1}{i} \frac{z^i}{2i+1} = \frac{1 - \sqrt{1 - 4z}}{2z},
\]
and according to \([1]\), we have that for all \( r \geq 1 \),

\[
(B_2(z))^r = \sum_i \binom{2i+r}{i} \frac{r z^i}{2i+r}.
\]

Replacing the closed form of \( B_2(z) \) in Equation 1.1, we obtain

\[
\Pr [Y < \infty \mid X_0 = j] = q^j \left( \frac{1 - \sqrt{1 - 4q(1 - q)}}{2q(1 - q)} \right) = q^j \left( \frac{1}{1 - q} \right)^j.
\]

Proceeding in a somewhat similar manner, we get for the expectation

\[
\mathbb{E}[Y \mid X_0 = j, Y < \infty] =
\]

\[
\frac{1}{\Pr [Y < \infty \mid X_0 = j]} \sum_{i \geq 0} (2i+j) \frac{j}{2i+j} P_q(i, j)
\]

\[
= \frac{j q^j}{\Pr [Y < \infty \mid X_0 = j]} \sum_{i \geq 0} \binom{2i+j}{i} (q(1 - q))^i
\]

\[
= \frac{j q^j}{q^j (B_2(q(1 - q)))} \sqrt{1 - 4q(1 - q)}
\]

\[
= \frac{j}{1 - 2q},
\]

\((1.3)\)
using the fact (see [1]) that for all \( r > 0, \)

\[
\frac{B_2(z)^r}{\sqrt{1-4z}} = \sum_i \binom{2i + r}{i} z^i. \quad \Box
\]

We now provide the calculation of \( \Pr[Y \leq 0] \) and of \( \mathbb{E}[Y \mid Y \leq \infty] \) as it is done in [15], but in a more generalized setting, since we will use this result in the next chapter as well.

**Lemma 1.7** Let \( Y \) and \( X_0 \) be two integer-valued random variables, subject to the following conditions:

- \( \Pr[X_0 = j] = \frac{1}{2^n} \binom{n}{j}; \)

- \( \Pr[Y < \infty \mid X_0 = j] = \lambda^j, \) for some \( \lambda < 1; \)

- \( \mathbb{E}[Y \mid X_0 = j, Y < \infty] = cj, \) for some fixed \( c. \)

Then the following identities hold:

- \( \Pr[Y < \infty] = \left(\frac{1+\lambda}{2}\right)^n, \)

- \( \mathbb{E}[Y \mid Y < \infty] = \frac{c\lambda}{1+\lambda} n. \)

**Proof** Using Bayes' rule and the Binomial Theorem, we have that

\[
\Pr[Y < \infty] = \sum_{0 \leq j \leq n} \Pr[X_0 = j] \Pr[Y < \infty \mid X_0 = j]
\]

\[
= \frac{1}{2^n} \sum_{0 \leq j \leq n} \binom{n}{j} \lambda^j
\]

\[
= \frac{1}{2^n} (1 + \lambda)^n = \left(\frac{1+\lambda}{2}\right)^n.
\]
Using the law of conditional probabilities, we obtain
\[
\mathbb{E}[Y \mid Y < \infty] = \sum_{i \geq 0} i \cdot \Pr[Y = i \mid Y < \infty]
\]
\[
= \frac{1}{\Pr[Y < \infty]} \sum_{i \geq 0} i \cdot \Pr[Y = i]
\]
\[
= \frac{1}{(1+\lambda)^n} \sum_{i \geq 0} i \cdot \sum_{j \geq 0} \Pr[X_0 = j] \Pr[Y = i \mid X_0 = j]
\]
\[
= \left(\frac{2}{1+\lambda}\right)^n \sum_{i \geq 0} i \cdot \sum_{j \geq 0} \frac{1}{2^n} \binom{n}{j} \Pr[Y = i \mid X_0 = j]
\]
\[
= \frac{1}{(1+\lambda)^n} \sum_{j \geq 0} \binom{n}{j} \sum_{i \geq 0} i \cdot \Pr[Y = i \mid X_0 = j]
\]
\[
= \frac{1}{(1+\lambda)^n} \sum_{j \geq 0} \binom{n}{j} \mathbb{E}[Y \mid Y < \infty, X_0 = j] \Pr[Y < \infty \mid X_0 = j]
\]
\[
= \frac{1}{(1+\lambda)^n} \sum_{j \geq 0} \binom{n}{j} cj\lambda^j.
\]
We now use the well known combinatorial identity \(\binom{n}{j} = \frac{n(n-1)}{j(j-1)}\) and the Binomial Theorem, giving
\[
\mathbb{E}[Y \mid Y < \infty] = \frac{cn\lambda}{(1+\lambda)^n} \sum_{j \geq 1} \binom{n-1}{j-1} \lambda^j
\]
\[
= \frac{cn\lambda}{(1+\lambda)^n} (1+\lambda)^{n-1}
\]
\[
= \frac{cn\lambda}{1+\lambda}. \quad \square
\]

**Lemma 1.8** Using the same conditions as in the previous lemma, we have that for all \(\gamma \geq 1\),
\[
\Pr\left[Y \leq \gamma \frac{c\lambda n}{1+\lambda}\right] > \left(1 - \frac{1}{\gamma}\right) \left(\frac{1+\lambda}{2}\right)^n.
\]

**Proof** Markov’s inequality states the following:
\[
\Pr[Y > \gamma \mathbb{E}[Y \mid Y < \infty] \mid Y < \infty] < \frac{1}{\gamma}.
\]

Since the event \(Y \leq \gamma \frac{c\lambda n}{1+\lambda}\) implies the event \(Y < \infty\), we have
\[
\Pr\left[Y \leq \gamma \frac{c\lambda n}{1+\lambda}\right] = \Pr\left[Y \leq \gamma \frac{c\lambda n}{1+\lambda} \mid Y < \infty\right] \cdot \Pr[Y < \infty]
\]
\[
\geq \left(1 - \frac{1}{\gamma}\right) \left(\frac{1+\lambda}{2}\right)^n. \quad \square
1. Introduction

To apply the result above to Schöning’s algorithm, we use \( c_q^{\text{sch}} = \frac{1}{1-2q}, \lambda_q^{\text{sch}} = \frac{q}{1-q}, \) \( \frac{\lambda_q^{\text{sch}}}{1+\lambda_q^{\text{sch}}} = q, \) and \( \gamma = 3, \) and finally obtain

\[
\Pr \left[ Y \leq 3n \frac{qn}{1-2q} \right] > \frac{2}{3} \left( \frac{1}{2(1-q)} \right)^n .
\]

Since \( q = \frac{1}{k} \) and \( k \geq 3, \) we obtain \( \frac{q}{1-2q} \leq 1 \) and thus

\[
\Pr [ Y \leq 3n ] \geq \Pr \left[ Y \leq 3n \frac{qn}{1-2q} \right] > \frac{2}{3} \left( \frac{k}{2(k-1)} \right)^n .
\]

Since \( Y \leq 3n \) implies a \( X_{3n} = 0, \) it also means, using Lemma 1.5 that Schöning’s Algorithm finds a satisfying assignment. Thus the probability that a satisfying assignment is found within \( 3n \) steps exceeds \( \frac{2}{3} \left( \frac{k}{2(k-1)} \right)^n . \)

We are now able to prove an upper bound on the amount of times we need to run Schöning’s algorithm so that a satisfying assignment is found with probability approaching 1. We will look now at a general and simple result, which will prove helpful in the next chapters as well.

**Lemma 1.9** Given an arbitrary randomized procedure that succeeds with probability \( p, \) we consider the algorithm consisting of repeatedly and independently invoking the procedure until success is obtained. Then the expected number of invocations is \( 1/p, \) and after \( T/p \) invocations success is guaranteed with probability greater than \( 1 - e^{-T}, \) for any \( T > 0. \)

**Proof** Let \( Z \) be a random variable that counts the number of invocations needed until a satisfying assignment is found. Clearly \( Z \) is geometrically distributed with parameter \( p. \) The first part of the theorem is now clear, since \( E[Z] = 1/p. \)

To verify the second part of the claim, we compute the probability that there are at least \( T/p \) invocations of the procedure. Clearly, using the definition of \( p, \) we have that \( \Pr [ Z > T/p ] = (1-p)^{T/p}. \) Using the identity \( (1-x)^{1/x} < e^{-1}, \) we conclude that

\[
\Pr [ Z > T/p ] < e^{-T} .
\]

Applying the previous lemma, we obtain the proof to Theorem 1.3.
A Variant of Schöning’s Algorithm

2.1 The Algorithm

We consider a modification of Schöning’s algorithm, where instead of flipping a random variable in a chosen dissatisfied clause, we flip each variable that appear in that clause independently with probability $0 < \rho < 1$. As before the algorithm will work on $k$-CNF formulas. The modified algorithm is given as procedure Sch-Variant below. The purpose of this chapter is to apply the equivalent of Schöning’s analysis to find an upper bound for the time complexity of this modified algorithm.

```
Procedure Sch-Variant(F, t, \rho)
    \alpha \in \text{rand}\{0,1\}^{\text{vbl}(F)};
    \text{if } \alpha \text{ satisfies } F \text{ then}
        \text{return } (\text{true}, \alpha)
    i = 0;
    \text{while } i < t \text{ do}
        C \in \text{vlt}(F, \alpha) \text{ (use any selection rule)};
        \text{for all } v \in \text{vbl}(C) \text{ do}
            b \in \text{rand}\{0,1\};
            \text{if } b \leq \rho \text{ then}
                \alpha(v) \leftarrow 1 - \alpha(v)
            \text{if } \alpha \text{ satisfies } F \text{ then}
                \text{return } (\text{true}, \alpha)
            i \leftarrow i + 1;
        \text{return } (\text{false}, \text{nil})
```
2. A Variant of Schöning’s Algorithm

2.2 The Coupling Argument

As before, let \( a^* \) be a particular assignment which satisfies \( F \), and let \( d = d_H(\alpha^*, \alpha) \) be the Hamming distance between \( \alpha^* \) and the current assignment that the algorithm is looking at, \( \alpha \). We present below the auxiliary coupling algorithm Sch-Variant-Coupling, which is somewhat similar to Sch-Coupling, which was used in the introductory chapter. We consider as in the Schöning case some ordering on \( V \), and for some clause \( C \), the function \( \xi(C, \alpha^*) \) as defined in Remark 1.4. If the algorithm decides to flip the variable \( \xi(C, \alpha^*) \), then the internal variable \( X \) is decremented by 1, and whenever it decides to flip some other variable, \( X \) is incremented by 1. The fact that both algorithms when run on the same formula and the same random bits perform the same number of iterations is evident. What remains to be proven is the invariant relation between \( X \) and \( d(\alpha, \alpha^*) \).

Procedure Sch-Variant-Coupling(\( F, t, \rho, \alpha^* \))

\[
\begin{align*}
\alpha &\in_{\text{rand}} \{0,1\}^{\text{vbl}(F)}; \\
X &= d(\alpha, \alpha^*); \\
\text{if } \alpha \text{ satisfies } F \text{ then} \\
&\quad \text{return (true, } \alpha \text{)} \\
i &= 0; \\
\text{while } i < t \text{ do} \\
&\quad C \in \text{vlt}(F, \alpha) \text{ (use any selection rule);} \\
&\quad \text{for all } v \in \text{vbl}(C) \text{ do} \\
&\quad & b \in_{\text{rand}} [0,1]; \\
&\quad & \text{if } b \leq \rho \text{ then} \\
&\quad & \quad \alpha(v) \leftarrow 1 - \alpha(v); \\
&\quad & \quad \text{if } v = \xi(C, \alpha^*) \text{ then} \\
&\quad & \quad \quad X \leftarrow X - 1 \\
&\quad & \quad \text{else} \\
&\quad & \quad \quad X \leftarrow X + 1 \\
&\quad & \quad i \leftarrow i + 1; \\
&\quad & \text{if } \alpha \text{ satisfies } F \text{ then} \\
&\quad & \quad \text{return (true, } \alpha \text{)} \\
&\quad \text{return (false, nil)}
\end{align*}
\]

We use the same meaning as in the previous chapter for \( X_i, \alpha_i \) and \( i_{\text{max}} \).

Lemma 2.1 For all possible runs of Sch-Variant-Coupling, and for all \( i \) such that \( 0 \leq i \leq i_{\text{max}} \), we have that \( d(\alpha_i, \alpha^*) \leq X_i \).

Proof We use induction, and the base case is trivial, as before. For \( 0 < i \leq i_{\text{max}} \), we assume that the claim holds for \( i - 1 \). Since \( \alpha_{i-1} \) is not satisfying, an unsatisfied clause \( C \) is picked. All variables in \( C \) are flipped independently.
with probability $\rho$, and the variables are flipped sequentially. We have to prove that the property is invariant after each flip. Let $a$ be the current assignment, $a'$ the assignment after the flip, and likewise $X$ and $X'$. Let $v$ be the variable that is currently being flipped.

If $v = \xi(C, a^*)$, then it must be the case that $a(v) = 1 - a^*(v)$. This implies that $d(a', a^*) = d(a, a^*) - 1$, i.e. the Hamming distance decreases. Since $X' = X - 1$, this implies that the condition $d(a, a^*) \leq X$ is maintained.

If $v \neq \xi(C, a^*)$, we have that $X' = X + 1$, and $d(a', a^*) = d(a, a^*) \pm 1$.

We thus conclude that in all cases, the difference between $X$ and $d(a', a^*)$ can only increase, thus proving the claim. $\square$

To prove that $X$ has the properties of a Markov chain, we need the following lemma.

**Lemma 2.2** In any run of SchCoupling, we have the following for all $i \geq 1$:

- If $X_{i-1} = 0$, then $X_i = 0$,
- If $X_{i-1} > 0$, then for all $\delta \in \mathbb{Z}$, and for all $j \geq 1$

$$
\Pr[X_i = j + \delta \mid X_{i-1} = j] = \binom{k-1}{1+\delta} \rho^{2+\delta}(1-\rho)^{k-2-\delta} + \binom{k-1}{\delta} \rho^{\delta} (1-\rho)^{k-\delta}.
$$

**Proof** The case $X_{i-1} = 0$ is clearly obvious. For the rest, we notice that in order to obtain $X_i = X_{i-1} + \delta$, one of two cases must occur: either $\xi(C, a^*)$ and $\delta + 1$ other variables are flipped (and there are $\binom{k-1}{1+\delta}$ ways to choose these variables), or $\xi(C, a^*)$ is not flipped, but $\delta$ other variables are (there are $\binom{k-1}{\delta}$ ways to choose them). $\square$

Note that the values of $\delta$ for which probability of transition is nonzero are $-1, \ldots, n - 1$.

### 2.3 Analysing a Particular Kind of Markov Chains

In the following section, we will limit our discussion to a family of Markov chains, of which the one introduced in the last section is a special case. We need to derive some properties that will enable us to establish some results similar to Lemma 1.6 from the previous chapter.

Basically we consider the Markov chains whose state space consists of the nonnegative integers, and have the following three properties:
2. A Variant of Schöning’s Algorithm

- once the state 0 is reached, the Markov chain remains in that state,
- from a nonzero state \( j \), the reachable states are \( j - 1, \ldots, j + k - 1 \), for some fixed \( k > 0 \),
- the transition probabilities are the same for all nonzero starting states (we will call this property state-space homogeneity).

To make this precise, we introduce the following definitions:

**Definition 2.3** The transition set \( \Delta \) is the set \( \{-1, \ldots, k - 1\} \).

**Definition 2.4** The transition function \( p : \Delta \rightarrow [0,1] \) is any function that satisfies the condition \( \sum_{\delta \in \Delta} p(\delta) = 1 \).

![Diagram of possible transitions from a state \( j \).](image)

**Definition 2.5** Given a transition set \( \Delta \), and a transition function \( p \), let \( \mathcal{W}^{(p)} = \{X_i^{(p)}\}_{i \geq 0} \) be a Markov chain whose state space is the set of nonnegative integers, and whose transition probabilities are given as follows:

\[
Pr \left[ X_{i+1}^{(p)} = \delta \mid X_i^{(p)} = 0 \right] = \begin{cases} 1 & \text{if } \delta = 0, \\ 0 & \text{otherwise}; \end{cases}
\]

\[
Pr \left[ X_{i+1}^{(p)} = j + \delta \mid X_i^{(p)} = j \geq 1 \right] = \begin{cases} p(\delta) & \text{if } \delta \in \Delta, \\ 0 & \text{otherwise}. \end{cases}
\]  

**Definition 2.6** In this section, we will restrict ourselves to the case where the transition function \( p \) satisfies the property \( 0 < p(-1) < p(1) + \ldots + p(k-1) \), which we will call positive bias; in other words, we consider transition functions where it is more likely to pick a positive transition than to pick \(-1\). It is easy to see that the transition functions that we have considered so far fulfill this property.

Given a random walk \( \mathcal{W}^{(p)} \) and a starting state \( j \), we are now interested in the following two problems:

- What is the probability that the random walk eventually reaches state 0?
• Given that \( W^p \) reaches state 0, what is the expected time it takes to do so?

We introduce now some concepts that will help us find an answer to these questions. First, we want to be able to record the transitions made by an execution of the random walk \( W^p \) starting in state \( j \), in the event that it reaches state 0. We can manage this by considering transition strings, which correspond to particular executions of the random walk.

**Definition 2.7** A transition string is a finite sequence of transitions, i.e. any element \( \pi \) from \( \Delta^* \) (see Definition 2.3). We will also use the term string to mean the same thing in an unambiguous context.

We are interested in the amount a transition string “moves” in the state space, so we introduce the displacement of a transition string as the difference between the starting state and the state reached at the end of the string.

**Definition 2.8** The displacement \( \sigma(\pi) \) of a transition string \( \pi \) is given by the negated sum of all its elements:

\[
\sigma(\pi) = -\sum_{i=1}^{||\pi||} \pi_i.
\]

Clearly, a transition string \( \pi \) can be seen as a trace of a particular execution of the algorithm, starting in state \( \sigma(\pi) \) and ending in state 0. However, not all transition strings correspond to actual executions. More specifically, such “invalid” transition strings are those which correspond to executions that arrive earlier in state 0. Note that the previous statement also takes care of transcripts of executions that would reach negative states. We want to exclude these “invalid” strings, so we will add the following condition: allow only those transition strings whose displacement is greater than the displacement of any of their prefixes. Since the displacement of a prefix of a transition string indicates the state that has been reached after executing that prefix, the condition that ensures that the all states except the final one are positive.

**Definition 2.9** An allowed transition string \( \pi \) is a transition string such that for any proper prefix \( \pi' \) of \( \pi \) we have that \( \sigma(\pi) > \sigma(\pi') \).

The space of allowed transition strings of displacement \( j \) will be denoted by \( S_j \). If we consider allowed transition strings of length \( m \), we will denote them by \( S_j^m \).

Clearly, the only allowed transition string of displacement 0 is the empty string \( \varepsilon \). The following two lemmas establish some useful properties of spaces of allowed transition strings.
2. A Variant of Schöning’s Algorithm

**Proposition 2.10** The following identity involving spaces of allowed transition strings of length $m$ holds for all $j \geq 1$ and $m \geq 1$:

$$S_j^m = \bigcup_{\delta \in \Delta} \delta S_{j+\delta}^{m-1},$$

where $\delta S_{j+\delta}^{m-1}$ denotes the set of strings obtained by concatenating $\delta$ with all strings from $S_{j+\delta}^{m-1}$.

**Lemma 2.11** For all $j \geq 1$,

$$S_j = \bigcup_{\delta \in \Delta} \delta S_{j+\delta}.$$

**Proof** From the definitions of spaces of allowed transition strings we get that $S_j = \bigcup_{m \geq 0} S_j^m$. Using the fact that all strings in $S_j$ are nonempty and the previous remark, we have

$$S_j = \bigcup_{m \geq 1} \bigcup_{\delta \in \Delta} \delta S_{j+\delta}^{m-1} = \bigcup_{\delta \in \Delta} \bigcup_{m \geq 1} S_{j+\delta}^{m-1} = \bigcup_{\delta \in \Delta} \delta S_{j+\delta}. \quad \square$$

Another useful observation is that a random walk starting in state $j$ and reaching 0 also passes through all points between $j$ and 0. This leads to the following lemma.

**Lemma 2.12** For all $j \geq 1$ we have that

$$S_j = S_1 S_{j-1}.$$

Furthermore, a string $a \in S_j$ splits uniquely into strings $a' \in S_1$ and $b' \in S_{j-1}$, making $S_j$ isomorphic to the cartesian product $S_1 \times S_{j-1}$.

**Proof** Let $a \in S_1$, $b \in S_{j-1}$. Clearly $\sigma(ab) = \sigma(a) + \sigma(b) = j$, and it can be easily seen that this is an allowed string. Thus $ab \in S_j$.

For the other direction, let $a \in S_j$. There exists a minimal prefix $a'$ (which is obviously unique) of $a$ such that $\sigma(a') = 1$. Then $a = a'b'$, with $a' \in S_1$ and $b' \in S_{j-1}$. \quad \square

Now we define precisely the way in which transition strings are related to the random walk:

**Definition 2.13** The random transition string $D$ is a random variable with values in $\bigcup_{j \geq 0} S_j \cup \{\omega\}$, with the outcomes defined as follows. For any nonnegative $j$, in case the random walk $W^{(p)}$ starting in state $j$ reaches state 0, let $m$ be the first time step when it does so; then the outcome will be the transition string (from $S_j$) consisting of the quantities $X_{i+1} - X_i$ for $0 \leq i < m$. Otherwise, the outcome will be the special symbol $\omega$. 

18
Remark 2.14 The probability of any outcome $\pi \in \mathcal{S}_j$ of $D$ is given by:

$$Pr[D = \pi \mid X_0 = j] = \prod_{i=1}^{\mid \pi \mid} p(\pi_i).$$

Remark 2.15 The stopping time $Y$ (defined in the first chapter as the first time step $i$ when $X_i = 0$) has the following property:

$$Y = \begin{cases} \infty & \text{if } D = \omega, \\ |D| & \text{otherwise}. \end{cases}$$

Notation. In order to make calculations easier, we introduce a shorthand notation for the quantities we want to evaluate:

$$f_j := Pr[Y < \infty \mid X_0 = j], \quad (2.2)$$
$$g_j := E[Y \mid Y < \infty, X_0 = j]. \quad (2.3)$$

Remark 2.16 Clearly $f_j$ can be expressed in the following manner:

$$f_j = Pr[D \neq \omega \mid X_0 = j] = \sum_{\pi \in \mathcal{S}_j} Pr[D_j = \pi] = \sum_{\pi \in \mathcal{S}_j} \prod_{i=1}^{\mid \pi \mid} p(\pi_i). \quad (2.4)$$

Lemma 2.17 There exists $\lambda \in (0, 1]$ such that for all $j \geq 1$, we have that $f_j = \lambda f_{j-1}$.

Proof Applying Remark 2.16 two times, and Lemma 2.12, we obtain

$$f_j = \sum_{\pi \in \mathcal{S}_j} \prod_{i=1}^{\mid \pi \mid} p(\pi_i)$$
$$= \sum_{\pi' \in \mathcal{S}_1} \sum_{\pi'' \in \mathcal{S}_{j-1}} \left( \prod_{i=1}^{\mid \pi' \mid} p(\pi'_i) \right) \cdot \left( \prod_{i=1}^{\mid \pi'' \mid} p(\pi''_i) \right)$$
$$= \sum_{\pi' \in \mathcal{S}_1} \left( \prod_{i=1}^{\mid \pi' \mid} p(\pi'_i) \right) \sum_{\pi'' \in \mathcal{S}_{j-1}} \left( \prod_{i=1}^{\mid \pi'' \mid} p(\pi''_i) \right)$$
$$= f_1 \cdot f_{j-1}.$$ 

We set $\lambda = f_1$ and the claim follows. □

Corollary 2.18 There exists $\lambda \in (0, 1]$ such that for all $j \geq 1$, $f_j = \lambda^j$. 

19
In order to exclude the case where \( \lambda = 1 \), we will prove that \( f_j \) gets arbitrarily close to 0 for growing \( j \).

**Notation.** In the proof of this result we will consider strings that do not contain the zero transitions, and we will use the following shorthands:

\[
\begin{align*}
R_j^m &= S_j^m \cap (\Delta \setminus \{0\})^*, \\
R_j &= S_j \cap (\Delta \setminus \{0\})^*.
\end{align*}
\]

**Definition 2.19** We note that in any string \( \pi \in S_j^m \), we can drop the 0 transitions, and obtain a string \( \pi' \) from \( R_j \), which is uniquely defined. We will say that \( \pi \) reduces to \( \pi' \).

**Proposition 2.20** Given a string \( \pi' \in R_j^l \), where \( l > 0 \) and \( j > 0 \), there are exactly \( (m-1) \) strings in \( S_j^m \) that reduce to \( \pi' \), for \( m \geq l \).

**Proof** In a string of length \( m \) exactly \( m - l \) positions have to be 0, and the others are determined by \( \pi' \). However, a string is not allowed if it contains a 0 on the last position. There are thus \( (m-1) \) many ways to choose the positions of the zeroes, and thus the claim follows. \( \square \)

**Lemma 2.21** We have that

\[
\lim_{j \to \infty} f_j = 0.
\]

**Proof** For any positive \( j \), Remark 2.16 and the fact that \( S_j = \cup_{m \geq 1} S_j^m \), we have

\[
f_j = \sum_{m \geq 1} \sum_{\pi \in S_j^m} \prod_{i=1}^{m} p(\pi_i).
\]

Using Proposition 2.20, we can group the strings that reduce to the same element of \( R_j \):

\[
f_j = \sum_{m \geq 1} \sum_{l=1}^{m} \sum_{\pi' \in R_j^l} \binom{m-1}{m-l} p(0)^{m-l} \prod_{i=1}^{l} p(\pi_i').
\]

Regrouping the terms and modifying the summation indices, and using the combinatorial identity \( \binom{m-1}{m-l} = \binom{m}{l-1} \), we get

\[
f_j = \sum_{l \geq 1} \sum_{\pi' \in R_j^l} \left( \prod_{i=1}^{l} p(\pi_i') \right) \sum_{m \geq l} \binom{m-1}{l-1} p(0)^{m-l}. \tag{2.5}
\]
Let $\gamma(l)$ denote the value of the innermost sum. In order to find a simpler formula for it, introduce the variable $M = m - l$, obtaining

$$\gamma(l) = \sum_{M \geq 0} \binom{M + l - 1}{l - 1} p(0)^M.$$ 

We use the well known combinatorial identity $\sum_{h \geq 0} \binom{n + h}{h} z^h = \frac{1}{1 - z} - \frac{1}{1 - p(0)}$, and obtain $\gamma(l) = \frac{1}{(1 - p(0))}$. We replace this value in equation 2.5. Also, we use the observation that $R^l_j$ is empty for $l < j$, and so it makes sense to consider summing only over the indices $l \geq j$:

$$f_j = \sum_{l \geq j} \sum_{\pi' \in R^l_j} \frac{1}{(1 - p(0))} \prod_{i=1}^l p(\pi'_i).$$ 

Moving some terms and regrouping factors, we get

$$f_j = \sum_{l \geq j} \sum_{\pi' \in R^l_j} \prod_{i=1}^l \frac{p(\pi'_i)}{1 - p(0)}.$$ 

(2.6)

Note that the innermost sum also represents the probability that the reduced random walk (i.e. the corresponding random walk where transitions that remain in the same state are not possible, and the other transitions are renormalized) reaches state 0 in exactly $l$ steps. We can thus define $W'$ as the random walk on the nonnegative integers, with transition probabilities $p'(\delta) = p(\delta)/(1 - p(0))$, for $\delta \in \Delta \setminus \{0\}$. Introducing $E_{j,l}$ as the event that number of steps needed for $W'$ to reach state 0 from state $j$ is exactly $l$, we have

$$\Pr[E_{j,l}] = \sum_{\pi' \in R^l_j} \prod_{i=1}^l \frac{p(\pi'_i)}{1 - p(0)}$$

and Equation 2.6 becomes

$$f_j = \sum_{l \geq j} \Pr[E_{j,l}].$$

(2.7)

The key to the rest of the proof relies in the observation that in a sequence of transitions starting at $j$ and ending in 0, we need at least $(l + j)/2$ transitions of type $-1$. This is so, because the other at most $\lceil (l - j)/2 \rceil$ transitions will
A Variant of Schöning’s Algorithm

go away from state 0, and it is easy to see that in case there are more such transitions, it would not be possible for the random walk to reach 0. We can conclude that the probability that $W'$ reaches state 0 from state $j$ in $l$ steps is upper bounded by the probability that in a reduced string (not necessarily allowed) of length $l$ at least $(l + j)/2$ transitions are of type $-1$. We can even relax this restriction, and just require that more than $l/2$ transitions are of type $-1$.

It is clear that the number $B_l$ of transitions of type $-1$ in a random string of length $l$ is a sum of $l$ independent Bernoulli variables, each yielding 1 with probability $p(-1)$. Then the required bound can be determined by applying Chernoff bounds. Also, the hypothesized condition $p(-1) < p(1) + \ldots + p(k - 1)$ implies that $p'(-1) < 1/2$.

Applying Chernoff bounds (see Appendix A.1) gives us the following result:

$$
\Pr \left[ E_{j,l} \right] \leq \Pr \left[ B_l > \frac{l}{2} \right] < e^{-2l \left( \frac{1}{2} - p'(-1) \right)^2}.
$$

We have proved so far the following upper bound for $f_j$ (using Equations 2.7 and 2.8):

$$
f_j < \sum_{l \geq j} e^{-2l \left( \frac{1}{2} - p'(-1) \right)^2}.
$$

Setting $\alpha = e^{-2 \left( \frac{1}{2} - p'(-1) \right)^2}$, which is clearly less than 1, we obtain

$$
f_j < \sum_{l \geq j} \alpha^l = \frac{\alpha^j}{1 - \alpha}.
$$

This implies that $f_j$ tends exponentially to 0 as $j$ increases, so the claim follows. \hfill \Box

What remains to be done in order to find a form for $f_j$ is computing the value of $\lambda$. The following lemma enables us to state that $\lambda$ is a root of a certain polynomial.

**Lemma 2.22** We have the following identity involving $f_j$, for all $j \geq 1$:

$$
f_j = \sum_{\delta \in \Delta} p(\delta) f_{j+\delta}.
$$

**Proof** First use Remarks 2.14 and 2.16:

$$
f_j = \sum_{\pi \in \mathcal{S}_j} \Pr \left[ D_j = \pi \right] = \sum_{\pi \in \mathcal{S}_j} \prod_{i=1}^{\vert \pi \vert} p(\pi_i).
$$


Analysing a Particular Kind of Markov Chains

We can split the first transition from the string $\pi$ by using Lemma 2.11, giving

$$f_j = \sum_{\delta \in \Delta} \sum_{\pi' \in S_{j+\delta}} p(\delta)^{|\pi'|} \prod_{i=1}^{\pi'_i} p(\pi'_i).$$

Regrouping, and using Equation 2.10, we have

$$f_j = \sum_{\delta \in \Delta} p(\delta) f_{j+\delta}. \quad \square$$

It is clear now that $\lambda$ is a root of the polynomial

$$p(-1) + (p(0) - 1)x + p(1)x^2 + \ldots + p(k-1)x^k = 0,$$

lying in the interval $(0,1]$.

**Lemma 2.23** The polynomial

$$h(x) = \sum_{\delta=-1}^{k-1} p(\delta)x^{\delta+1} - x$$

has exactly one zero in the interval $(0,1)$, if $p$ is a positively biased (see Definition 2.6) transition function.

**Proof** We first note that since $p$ is a transition functions, 1 is always a zero of $h(x)$, as all the values of $p$ sum up to 1. We notice that $h(x) + x$ has only positive coefficients, and is thus convex on the interval $(0,\infty)$. Since a root of $h(x)$ must satisfy $h(x) + x = x$, and the left hand side is convex, while the right hand side is linear, we conclude that there must be at most two zeroes in $(0,\infty)$, one of which is 1. We will now show that if $p$ is positively biased, then $h'(1) > 0$; this together with the fact that $h(0)$ is positive, will complete the proof, since $h$ is continuous.

In order to show that $h'(1) > 0$, we take the polynomial $h_2(x) = h(x)/x$, and we notice that

$$h'_2(1) = \left[ \frac{h'(x)x - h(x)}{x^2} \right]_{x=1} = h'(1).$$

Computing the value of $h'_2(1)$ and using the positive bias property,

$$h'_2(1) = -p(-1) + p(1) + 2p(2) + \ldots + (k-1)p(k-1) > 0. \quad \square$$
Now we will compute an expression for $g_j = E [ Y \mid Y < \infty, X_0 = j ]$.

**Lemma 2.24** For all $j \geq 1$, we have $g_j f_j = \sum_{\delta \in \Delta} p(\delta) g_{j+\delta} f_{j+\delta} + f_j$.

**Proof** Using the definition of expectation, we have

$$E [ Y \mid Y < \infty, X_0 = j ] = \sum_{\pi \in S_j} |\pi| \Pr [ D = \pi \mid D \neq \omega, X_0 = j ] .$$

Since $D = \pi$ as an event implies $D \neq \omega$, we conclude that

$$\Pr [ D = \pi \mid D \neq \omega, X_0 = j ] = \frac{\Pr [ D = \pi \mid X_0 = j ]}{\Pr [ D \neq \omega, X_0 = j ]} .$$

Obviously, $\Pr [ D \neq \omega \mid X_0 = j ] = f_j$, and so using Remark 2.14, we have

$$g_j = \sum_{\pi \in S_j} |\pi| \prod_{i=1}^{\pi} \frac{p(\pi_i)}{f_j} . \tag{2.11}$$

Splitting the space $S_j$ into $S_j^m$ for all $m$, we have

$$g_j f_j = \sum_{m \geq 1} m \sum_{\pi \in S_j^m} \prod_{i=1}^{\pi} p(\pi_i) . \tag{2.12}$$

As in the case of $f_j$, we split the first transition from $\pi$ using Proposition 2.10, obtaining

$$g_j f_j = \sum_{m \geq 1} m \sum_{\delta \in \Delta} \sum_{\pi' \in S_j^{m-1}} \prod_{i=1}^{\pi'} p(\pi_i') .$$

Regrouping, and setting $m' = m - 1$, we obtain

$$g_j f_j = \sum_{\delta \in \Delta} p(\delta) \sum_{m \geq 1} (m - 1 + 1) \sum_{\pi' \in S_j^{m-1}} \prod_{i=1}^{m-1} p(\pi_i')$$

$$= \sum_{\delta \in \Delta} p(\delta) \sum_{m' \geq 0} (m' + 1) \sum_{\pi' \in S_j^{m'}} \prod_{i=1}^{m'} p(\pi_i')$$

$$= \sum_{\delta \in \Delta} p(\delta) \sum_{m' \geq 0} m' \sum_{\pi' \in S_j^{m'}} \prod_{i=1}^{m'} p(\pi_i') + \sum_{\delta \in \Delta} p(\delta) \sum_{m' \geq 0} \sum_{\pi' \in S_j^{m'}} \prod_{i=1}^{\pi'} p(\pi_i').$$
Analysing a Particular Kind of Markov Chains

Note that the first term is just \( \sum_{\delta \in \Delta} p(\delta) g_{j+\delta} f_{j+\delta} \), using Equation 2.12. In the second term, we use the fact that \( S_{j+\delta} = \bigcup_{m' \geq 0} S_{j+\delta}^{m'} \), obtaining

\[
\sum_{m' \geq 0} \sum_{\pi' \in S_{j+\delta}^{m'}} |\pi'| \prod_{i=1}^{\pi'} p(\pi'_i) = \sum_{\pi' \in S_{j+\delta}} |\pi'| \prod_{i=1}^{\pi'} p(\pi'_i).
\]

We notice that this is equal to \( f_j \), using equation 2.10, and thus the expression for \( g_j f_j \) is

\[
g_j f_j = \sum_{\delta \in \Delta} p(\delta) g_{j+\delta} f_{j+\delta} + f_j.
\]

Using this intermediary result, we are now able to compute the closed form of \( g_j \).

**Lemma 2.25** For all \( j \geq 1 \), we have that \( g_j = c \cdot j \), where

\[
c = \left( -\sum_{\delta \in \Delta} \delta p(\delta) \lambda^\delta \right)^{-1}.
\]

**Proof** Using Equation 2.11 and Lemma 2.12, we obtain

\[
g_j f_j = \sum_{\pi' \in \Delta_1} \sum_{\pi'' \in \Delta_{j-1}} (|\pi'| + |\pi''|) \prod_{i=1}^{\pi'} p(\pi'_i) \cdot \prod_{i=1}^{\pi''} p(\pi''_i)
\]

\[
= \left( \sum_{\pi' \in \Delta_1} |\pi'| \prod_{i=1}^{\pi'} p(\pi'_i) \right) \left( \sum_{\pi'' \in \Delta_{j-1}} \prod_{i=1}^{\pi''} p(\pi''_i) \right) +
\]

\[
+ \left( \sum_{\pi'' \in \Delta_{j-1}} |\pi''| \prod_{i=1}^{\pi''} p(\pi''_i) \right) \left( \sum_{\pi' \in \Delta_1} \prod_{i=1}^{\pi'} p(\pi'_i) \right)
\]

\[
= g_1 f_1 f_{j-1} + g_{j-1} f_{j-1} f_1.
\]

Since \( f_j = f_1 \cdot f_{j-1} \), we get the simple expression \( g_j = g_1 + g_{j-1} \), or \( g_j = j \cdot g_1 \).

Thus, we have \( c = g_1 \), and use Lemma 2.24 to determine its value:

\[
c j f_j = \sum_{\delta \in \Delta} p(\delta) c \cdot (j + \delta) \cdot f_{j+\delta} + f_j.
\]

We replace \( f_j \) by \( \lambda^j \), and obtain

\[
c j = \sum_{\delta \in \Delta} p(\delta) c \cdot (j + \delta) \cdot \lambda^\delta + 1.
\]
Since $\sum_{\delta \in \Delta} p(\delta)\lambda^\delta = 1$, we can reduce the last equation to

$$0 = \sum_{\delta \in \Delta} \delta p(\delta)c\lambda^\delta + 1,$$

and thus finish the proof.

### 2.4 Schöning’s Algorithm as a Special Case

We can of course apply the results from the last section for the analysis of Schöning’s original algorithm. Then we choose $\Delta = \{-1,0,1\}$, $p(-1) = q$, $p(0) = 0$ and $p(1) = 1 - q$. Then $\lambda_q^{\text{sch}}$ has to be a solution of the equation

$$q - x + (1 - q)x^2 = 0.$$

It can be easily checked that the solutions are 1 and $\frac{q}{1-q}$, and we deduce that $\lambda_q^{\text{sch}} = \frac{q}{1-q}$, which is the value we obtained in the introductory chapter. To determine $c_q^{\text{sch}}$, we apply Lemma 2.25 and obtain

$$c_q^{\text{sch}} = \left( p(-1)\frac{1}{\lambda_q^{\text{sch}}} - p(1)\lambda_q^{\text{sch}} \right)^{-1} = \left( \frac{1-q}{q} - (1-q)\frac{q}{1-q} \right)^{-1} = \frac{1}{1-2q},$$

which also coincides with our result from the previous chapter.

### 2.5 Analysis of the Modified Algorithm

For a given $k \geq 3$ and $\rho \in (0,1)$, we consider the Markov chain corresponding to the transition function

$$p_{\rho,k}(\delta) = \binom{k-1}{1+\delta}(1-\rho)^{k-2-\delta} + \binom{k-1}{\delta}\rho^\delta(1-\rho)^{k-\delta}.$$

Computing $\lambda_{\rho,k}^{\text{var}} = \Pr[Y < \infty \mid X_0 = 1]$ amounts to determining the single root of the polynomial

$$p(-1) + (p(0) - 1)x + p(1)x^2 + \ldots + p(k-1)x^{k} = 0. \quad (2.13)$$

In general, this is not straightforward, since calculating zeros of polynomials in general can only be done if their degree is at most 4. Thus, we are
able to find closed forms for $k = 3, 4$ or $5$, since one root is always $1$, and we can divide the polynomial by $x - 1$. However, the roots can be easily approximated numerically for any $k$.

We give here the detailed calculation of $\lambda_{\rho,3}$, which amounts basically to solving a second degree equation.

**Proposition 2.26** For any $\rho \in (0,1)$, we have

$$\lambda_{\rho,3} = \frac{-2\rho^2 - 3\rho + 2 + \sqrt{5\rho^2 - 8\rho + 4}}{2\rho(1 - \rho)}$$

**Proof** The transition function is given by

$$p_{\rho,3}(-1) = (1 - \rho)^2,$$
$$p_{\rho,3}(0) = (1 - \rho)^3 + 2\rho^2(1 - \rho),$$
$$p_{\rho,3}(+1) = 2\rho(1 - \rho)^2 + \rho^3,$$
$$p_{\rho,3}(+2) = \rho^2(1 - \rho).$$

(2.14)

Then $\lambda_{\rho,3}$ is a solution to

$$p_{\rho,3}(+2)x^3 + p_{\rho,3}(+1)x^2 + (p_{\rho,3}(0) - 1)x + p_{\rho,3}(-1) = 0.$$ 

As we already know that $1$ is a solution, we divide the polynomial by $x - 1$ and obtain

$$p_{\rho,3}(+2)x^2 + (p_{\rho,3}(+2) + p_{\rho,3}(+1))x + p_{\rho,3}(-1) = 0.$$ 

By replacing the transition probabilities with their values, we get

$$\rho^2(1 - \rho)x^2 + (\rho^2(1 - \rho) + 2\rho(1 - \rho)^2 + \rho^3)x + \rho(1 - \rho)^2 = 0.$$ 

Dividing by $\rho$ and rearranging the terms, we have

$$\rho(1 - \rho)x^2 + (2\rho^2 - 3\rho + 2)x + (1 - \rho)^2 = 0.$$ 

The nonnegative solution, is what we are looking for, namely

$$\lambda_{\rho,3} = \frac{-2\rho^2 - 3\rho + 2 + \sqrt{5\rho^2 - 8\rho + 4}}{2\rho(1 - \rho)}.$$ 

□
In particular, for \( \rho = 1/2 \) (which corresponds to resampling uniformly all variables that appear in the unsatisfied clause \( C \)), we obtain \( \lambda^{\text{var}}_{1/2,3} = \sqrt{5} - 2 \).

Also, \( c^{\text{var}}_{\rho,3} \) can be determined using Lemma 2.11. To keep formulas simple, we will use \( \lambda \) to mean \( \lambda^{\text{var}}_{\rho,k} \) and \( c \) to mean \( c^{\text{var}}_{\rho,k} \). Then we can readily apply Lemmas 1.6, 1.7 and 1.8 from the introductory chapter, and get

\[
\Pr \left[ Y_{\rho,k} \leq \gamma \frac{c \lambda n}{1 + \lambda} \right] > \left( 1 - \frac{1}{\gamma} \right) \left( \frac{1 + \lambda}{2} \right)^n.
\]

We choose the maximum allowed number of iterations in Sch-Variant to be \( t = \gamma \frac{c \lambda n}{1 + \lambda} \) (which is in any case linear in \( n \)), and then using Lemma 1.9, we are able to conclude the analysis with the following theorem.

**Theorem 2.27** In expectation, for a satisfiable \( k \)-CNF formula \( F \), a satisfying assignment will be found after repeating Sch-Variant(\( F, t \)) at most \( \frac{\gamma}{\gamma - 1} \left( \frac{2}{1 + \gamma} \right)^n \) times, where \( t = \gamma \frac{c \lambda n}{1 + \lambda}, \gamma > 1 \) is arbitrary, and \( \lambda \) is the unique solution of Equation 2.13 in the interval \((0,1)\). The probability that it does not find a satisfying assignment after \( T \) times the expectation steps is at most \( e^{-T} \).

**Corollary 2.28** For \( q = 1/2 \) and \( k = 3 \), the expected runtime of the algorithm is \( O(\varphi^n) \), where \( \varphi = \frac{\sqrt{5} + 1}{2} \) is the golden ratio.

### 2.6 Relation to Schöning’s algorithm

We will now show how to obtain with Sch-Variant a time complexity that is close to that of Schöning’s Algorithm, by considering \( \rho \) arbitrarily close to 0.

**Remark 2.29** For \( k = 3 \), using l’Hôpital’s rule, we have that

\[
\lim_{\rho \to 0} \lambda_{\rho,k} = \lim_{\rho \to 0} \frac{-(2\rho^2 - 3\rho + 2) + \sqrt{5\rho^2 - 8\rho + 4}}{2\rho(1 - \rho)} = \lim_{\rho \to 0} \frac{1}{2 - \rho} \left( -(4\rho - 3) + \frac{10\rho - 8}{2\sqrt{5\rho^2 - 8\rho + 4}} \right) = \frac{1}{2}.
\]

We can also see this clearly in Figure 2.2, where we have plotted \( \lambda_{\rho,k} \) for \( k \) between 3 and 6.

In this case, the expected time it takes for the algorithm to reach a satisfying assignment approaches \( O((4/3)^n \text{poly}(n)) \), which corresponds to Schöning’s algorithm. This also makes sense intuitively, for the following reason: whenever all variables that are present in the chosen not satisfied clause are
flipped with a very small probability $q$, if we disregard the cases when no variable is flipped at all, then with the highest probability only a single variable was flipped — which is also the case in Schöning’s algorithm.

We also show how it works intuitively for $k > 3$. To see this, we examine the transition probabilities

$$p_{q,k}(\delta) = \binom{k-1}{1+\delta} \rho^{2+\delta}(1-\rho)^{k-2-\delta} + \binom{k-1}{\delta} \rho^{\delta}(1-\rho)^{k-\delta}.$$

As $\rho$ tends to 0, the highest order term contains $\rho^0(1-\rho)^k$, and appears in $p(0)$. The next highest order term contains $\rho(1-\rho)^{k-1}$, and corresponds to that which appears in both $p(-1)$ and $p(+1)$. Furthermore, we have $p(+1) = (k-1)p(-1) + o(\rho^2)$ by just examining the terms with $\rho^2$ and their coefficients. We get the following equation, whose coefficients differ by at most $o(\rho^2)$ from the original:

$$p_{q,k}(-1) - kp_{q,k}(-1)x + (k-1)p_{q,k}(-1)x^2 = 0.$$

The factor $p_{q,k}(-1)$ clearly reduces, and we obtain the equation $1 - kx + (k-1)x^2 = 0$, whose solution is $\lambda = 1/(k-1)$, which is exactly what we obtain in the Schöning case.
In the previous chapters, we have provided a general upper bound on the number of necessary runs for both Schöning’s algorithm and the modified version. Lower bounds depend, however, on the choice of a certain selection rule. In this chapter, we consider the selection rules “Devil” and “Random” from the introductory chapter, and observe their behaviour on particular formulas. We show that on these formulas, the lower bound on the number of repeats necessary for both algorithms differs from the upper bound by only a polynomial factor, which implies that our analysis from the previous chapters was tight.

The formulas that we will consider in this chapter will have one single satisfying assignment, namely the one which maps all variables to 1. Thus, the Hamming distance to the satisfying assignment at any one point will be given by the number of 0’s in the current assignment. As in the previous chapter, we will want to couple this Hamming distance to the state of a particular Markov chain. The difference will be that in this case, we want to ensure that the current state in the Markov chain is always at most (not at least) as large as the Hamming distance. This will imply that the transition probabilities may be different, depending on the current state in the Markov chain, i.e. the Markov chain will not be homogenous (in the sense we have defined in the last chapter). The first part of this chapter will be devoted to developing some tools that handle more general Markov chains.

### 3.1 General Treatment

We will now discuss some properties of an even more generalized type of the Markov Chains than the ones considered in Section 2.3. To be more exact, we consider chains that are not necessarily homogenous. We generally allow
all possible transitions between states that correspond to nonzero values of a generalized transition function, defined below.

**Definition 3.1** A generalized transition function is a function \( \tilde{p} : \mathbb{N}_0 \times \mathbb{N}_0 \to (0,1) \), with the following two properties:

- for all \( j \geq 0 \), \( \sum_{i \geq 0} \tilde{p}(j,i) = 1 \),
- \( \tilde{p}(0,0) = 1 \).

We also state now a useful additional property:

**Definition 3.2** A generalized transition function \( \tilde{p} \) is called regular at \( j \), where \( j > 0 \), if \( \tilde{p}(j',j'') = 0 \) for all \( j' > j \) and \( j'' < j \).

In other words, \( \tilde{p} \) is regular at \( j \) if the random walk cannot jump over \( j \) when going from large states to small states.

**Definition 3.3** The Markov chain associated to the generalized transition function \( \tilde{p} \) is given by the Markov chain \( \mathcal{W}(\tilde{p}) = \{ X_t(\tilde{p}) \}_{t \leq 0} \), whose state space consists of the nonnegative integers, and whose transition probabilities are

\[
Pr \left[ X_{t+1}(q) = i \mid X_t(q) = j \right] = \tilde{p}(j,i).
\]

Clearly the Markov property (cf. Definition 1.1) and the time homogeneity property (cf. Definition 1.2) are both satisfied. Also, a transition function \( p \) (cf. Definition 2.4) can be easily transformed into a generalized one \( \tilde{p} \) by setting \( \tilde{p}(j,i) = p(i-j) \) if \( j > 0 \) and \( i-j \in \Delta \). Then clearly \( \mathcal{W}(p) \) is the same Markov chain as \( \mathcal{W}(\tilde{p}) \).

Just as in the previous chapter, we will first create some methods to transform representations of possible executions of the random walk. Instead of recording the transitions, we will find it more useful to record the states themselves that the random walk passes through. Thus, we will consider strings of states, which we will call **transcripts**, and define them as follows:

**Definition 3.4** For any \( t \geq 0 \), a \( t \)-transcript w.r.t. \( \tilde{p} \) is a string \( s = s_0 \ldots s_t \) containing nonnegative integers, such that for all \( 0 \leq i < t \) we have that \( \tilde{p}(s_i,s_{i+1}) > 0 \).

We denote by \( T_{j}^{t} \) the set of all \( t \)-transcripts whose first element is \( j \) and last element is \( j' \). Henceforth, \( j \) will also be called starting state, and \( j' \) ending state for the set of transcripts.

We also introduce the sets \( T_{j}^{t} = \cup_{t \leq 0} T_{j}^{t} \), \( T_{j}^{t} = \cup_{j \leq t} T_{j}^{t} \), and \( T_{j}^{t} = \cup_{t \leq 0} T_{j}^{t} \).
Clearly, the transcripts can only be defined once a generalized transition function \( \tilde{p} \) is given. Since notation is already overloaded, we will assume that \( \tilde{p} \) is clear from context, and only specify it otherwise.

**Notation.** Given a generalized transition probability sequence \( \tilde{p} \) and its associated random walk \( \mathcal{W}(\tilde{p}) \), we want to describe the probability of occurrence of a transcript in \( \mathcal{W}(\tilde{p}) \). We introduce the following notation for the probability of those events, assuming \( \tilde{p} \) is implicit:

\[
\Pr[s_0 \ldots s_t] = \Pr\left[X_t^{(q)} = s_t \land \ldots \land X_1^{(q)} = s_1 \mid X_0^{(q)} = s_0\right].
\]

We extend this notation to sets of strings, so for any \( A \subset T_{j,s}^t \) we define

\[
\Pr[A] = \Pr\left[\bigvee_{s \in A} \left(X_t^{(q)} = s_t \land \ldots \land X_1^{(q)} = s_1\right) \mid X_0^{(q)} = s_0\right].
\]

Note that this can be done only for sets of transcripts that start at the same state \( j \).

**Remark 3.5** Let \( \mathcal{A} \subset T_{j,s}^t \). If \( \mathcal{A} \) is prefix-free, then the events corresponding to each transcript in \( \mathcal{A} \) are disjoint. This implies that

\[
\Pr[A] = \sum_{s \in A} \Pr[s].
\]

**Remark 3.6** Clearly \( T_{j,s}^t \) is prefix-free, since all transcripts contained have the same length. Thus, Remark 3.5 holds when \( A \subset T_{j,s}^t \).

Clearly the probability that the random walk reaches 0 from \( j \) within \( t > 0 \) steps is given by \( \Pr[T_{j,0}^t] \).

**Definition 3.7** For \( j \geq j' \geq 0 \), let \( \omega_{j,j'}^t : T_{j,j'}^t \to \{0 \ldots t\} \) be a function defined as the starting index of the largest suffix of \( s \) whose elements are not larger than \( j \).

We now introduce a special class of transcripts, those which do not contain states larger than their starting state.

**Definition 3.8** For \( j \geq j' \geq 0 \), let \( \overrightarrow{T}_{j,j'}^t \) denote the set of transcripts \( s \) such that \( s \in T_{j,j'}^t \) and \( s_i \leq j \) for all \( 0 \leq i \leq t \). We will call these the right bounded transcripts.

**Remark 3.9** We have that if \( \tilde{p} \) is regular at \( j \) then \( \overrightarrow{T}_{j,j'}^t = \left(\omega_{j,j'}^t\right)^{-1}(0) \).

This is so since a right bounded transcript \( s \) has all elements upper bounded by \( j \), and thus \( \omega_{j,j'}^t(s) = 0 \). Generalizing, for some \( 0 \leq i \leq t \), \( \overrightarrow{T}_{j,j'}^{t-i} = \left(\omega_{j,j'}^t\right)^{-1}(i) \), since after position \( \omega_{j,j'}^t(s) \) all elements of \( s \) are at most \( j \).
3. Worst-Case Analysis

Definition 3.10 We define transcript composition in the following way: for arbitrary nonnegative \( j, j', j'' \) and \( t_1, t_2 \geq 0 \), given two transcripts \( a \in T_{j,j}^{t_1} \) and \( b \in T_{j',j''}^{t_2} \), their composition is defined as the transcript \( a \circ b \in T_{j,j''}^{t_1+t_2} \) given by \( a \circ b = a_0 \ldots a_{t_1} b_1 \ldots b_{t_2} \). (This is just like normal concatenation, just that we omit the one extra occurrence of \( j' \).

We also extend transcript composition to operate on sets of transcripts in a straightforward manner: if \( A \subset T_{j,j}^{t_1} \) and \( B \subset T_{j',j''}^{t_2} \), we define

\[
A \circ B = \bigcup_{a \in A} \bigcup_{b \in B} a \circ b.
\]

Proposition 3.11 Let \( a \in T_{j,j}^{t_1} \), \( b \in T_{j',j''}^{t_2} \), and \( s = a \circ b \). Then

\[
\Pr [s] = \Pr [a] \Pr [b].
\]

The former equality can be extended to sets of transcripts. Thus, given that \( A \subset T_{j,j}^{t_1} \) and \( B \subset T_{j',j''}^{t_2} \), we have that

\[
\Pr [A \circ B] = \Pr [A] \Pr [B].
\]

Proof Using the law of conditional probability, we have

\[
\Pr [a \circ b] = \Pr \left[ X^{(q)}_{t_1+1} = b_1, \ldots X^{(q)}_{t_1+t_2} = b_{t_2} \mid X^{(q)}_0 = a_0 \right]
\]

\[
= \Pr \left[ X^{(q)}_{t_1+1} = b_1, \ldots X^{(q)}_{t_1+2} = b_{t_2} \mid X^{(q)}_0 = a_0, X^{(q)}_1 = a_1, \ldots X^{(q)}_{t_1} = a_{t_1} \right].
\]

Using the Markov property, the time-homogeneity property and the fact that \( a_{t_1} = b_0 \), we have that

\[
\Pr \left[ X^{(q)}_{t_1+1} = b_1, \ldots X^{(q)}_{t_1+t_2} = b_{t_2} \mid X^{(q)}_0 = a_0, X^{(q)}_1 = a_1, \ldots X^{(q)}_{t_1} = a_{t_1} \right] =
\]

\[
= \Pr \left[ X^{(q)}_{t_1+1} = b_1, \ldots X^{(q)}_{t_1+t_2} = b_{t_2} \mid X^{(q)}_1 = a_{t_1} \right] =
\]

\[
= \Pr \left[ X^{(q)}_1 = b_1, \ldots X^{(q)}_{t_2} = b_{t_2} \mid X^{(q)}_0 = b_0 \right]
\]

It is then clear that

\[
\Pr [a \circ b] = \Pr \left[ X^{(q)}_1 = b_1, \ldots X^{(q)}_{t_2} = b_{t_2} \mid X^{(q)}_0 = b_0 \right].
\]

\[
\cdot \Pr \left[ X^{(q)}_1 = a_1, \ldots X^{(q)}_{t_1} = a_{t_1} \mid X^{(q)}_0 = a_0 \right]
\]

\[
= \Pr [a] \Pr [b].
\]
For sets of transcripts, we have
\[
\Pr[\mathcal{A} \circ \mathcal{B}] = \sum_{a \in \mathcal{A}} \sum_{b \in \mathcal{B}} \Pr[a \circ b] = \left( \sum_{a \in \mathcal{A}} \Pr[a] \right) \left( \sum_{b \in \mathcal{B}} \Pr[b] \right) = \Pr[\mathcal{A}] \Pr[\mathcal{B}] .
\]

**Proposition 3.12** For \( j \geq j' \geq 0 \), with \( j > 0 \), let \( s \in T_{t}^{j} ; 0 \) such that \( i = \omega_{t}^{j} (s) > 0 \). Then \( s = a \circ (j'', j) \circ b \), such that \( a \in T_{t}^{j''} ; 0 \), and \( b \in \overline{T}_{t}^{j''} ; j \), where \( j'' > j \) is such that \( \tilde{p}(j'', j) > 0 \).

Furthermore, for all \( 0 < i \leq t \), we have that
\[
\left( \omega_{t}^{j} \right)^{-1} (i) = \bigcup_{\substack{j'' > j \\
\tilde{p}(j'', j) > 0}} (T_{t}^{j''-1} \circ (j'', j)) \circ T_{t}^{j-i}.
\]

**Proof** Suppose \( s_{i-1} > j \). Then clearly \( a = s_{0} \ldots s_{i-1} \) and \( b = s_{i} \ldots s_{t} \) satisfy the hypothesis.

Suppose now \( s_{i-1} \leq j \). Let \( i' < i \) be the highest possible index such that \( s_{i'} = j \). Since \( \tilde{p} \) is regular at \( j \), we have that \( s_{k} < j \) for all \( i' < k < i - 1 \). (Otherwise we would need to jump from a state greater than \( j \) to a smaller one.) If this is the case, then \( \omega_{t}^{j} (s) \leq i' \), contradiction.

This also proves the “\( \subset \)” part of Equation 3.1. To prove the other direction, it is sufficient to take \( a \in T_{t}^{j''-1} ; 0 \) and \( b \in T_{t}^{j''-1} ; j \), with \( j'' > 0 \) such that \( \tilde{p}(j'', j) > 0 \) and notice that \( \omega_{t}^{j} (a \circ (j + 1, j) \circ b) = i \).

**Notation.** Given \( s \in T_{t}^{j} ; 0 \), define \( \zeta_{k}(s) = s \circ 0 \ldots 0 \), i.e. the original transcript \( s \), with \( k \) zeroes appended to it.

**Remark 3.13** We have that
\[
Pr[\zeta_{k}(s)] = Pr[s] Pr[0 \ldots 0]^{k+1 \text{ times}} = Pr[s] .
\]

**Proposition 3.14** Given \( j \geq 0 \) and \( 0 < t < t' \), we have that
\[
Pr[\overline{T}_{t}^{j} ; 0] \leq Pr[\overline{T}_{t'}^{j} ; 0] .
\]
Proof As the random walk reaches state 0 once, it will stay there, so the event that a shorter transcript from \(j\) to 0 was generated is clearly included in the event that the same transcript padded with 0’s at the end was generated. Thus, we have

\[
\Pr \left[ \tilde{T}_{j,0}^t \right] = \sum_{s \in \tilde{T}_{j,0}^t} \Pr [s].
\]

Using Remark 3.13 and since \(\zeta_k(s) \in \tilde{T}_{j,0}^{t'}\), we obtain

\[
\Pr \left[ \tilde{T}_{j,0}^t \right] = \sum_{s \in \tilde{T}_{j,0}^t} \Pr [\zeta_k(s)] \leq \sum_{s \in \tilde{T}_{j,0}^{t'}} \Pr [s] = \Pr \left[ \tilde{T}_{j,0}^{t'} \right].
\]

The following lemma relates the probability of the random walk reaching 0 in \(t\) steps to the probability of reaching 0 in \(t\) steps without exceeding the starting state \(j\).

**Lemma 3.15** For any \(t \geq 1\) and \(j > 0\) and any generalized transition function \(\tilde{p}\) regular at \(j\), we have that

\[
\Pr \left[ T_{j,0}^t \right] \leq t \cdot \Pr \left[ \tilde{T}_{j,0}^t \right].
\]

**Proof** We group transcripts from \(T_{j,0}^t\) by the corresponding values returned by the map \(\omega_{j,0}^t\) (which in this case will be always less than \(t\)):

\[
\Pr \left[ T_{j,0}^t \right] = \sum_{0 \leq i < t} \Pr \left[ \left( \omega_{j,0}^t \right)^{-1} (i) \right].
\]

We separate the first group, for which \(\omega_{j,0}^t\) is 0, and use Remark 3.9. For the other groups, we use Proposition 3.12:
We use the (gross) approximation \( \Pr \left[ \bigcup_{j'' > j} \left( T_{j''}^{t-1} \circ \tilde{p}(j'', j) \right) \right] \leq 1. \)

Also, using Proposition 3.14, we deduce that

\[
\Pr \left[ T_{j,0}^t \right] \leq \Pr \left[ T_{j,0}^t \right] \leq \Pr \left[ T_{j,0}^t \right].
\]

(3.3)

We can now substitute the corresponding terms in Equation 3.2, and derive the last step in the proof:

\[
\Pr \left[ T_{j,0}^t \right] \leq \Pr \left[ T_{j,0}^t \right] + \sum_{1 \leq i < t} \Pr \left[ T_{j,0}^t \right] \leq t \cdot \Pr \left[ T_{j,0}^t \right].
\]

We now formulate a lemma that states that if \( \tilde{p} \) is regular at \( j \), then the probability to reach 0 from any state \( j' > j \) within \( t \) steps is upper bounded by the probability to reach 0 from \( j \) in \( t \) steps.

**Lemma 3.16** For any \( t \geq 0 \) and \( j > 0 \) and any generalized transition function \( \tilde{p} \) regular at \( j \), we have that for all \( j' > j \),

\[
\Pr \left[ T_{j',0}^t \right] \leq \Pr \left[ T_{j,0}^t \right].
\]

**Proof** We give a proof by induction. For \( t = 0 \), the statement is trivial, since both probabilities are 0.

If \( t > 0 \), we can split off the first state using the following identity:

\[
T_{j',0}^t = \bigcup_{j'' : \tilde{p}(j', j'') > 0} (j', j'') \circ T_{j',0}^{t-1}.
\]

Using the fact that \( \tilde{p} \) is regular at \( j \), it is clear that \( \tilde{p}(j', j'') > 0 \) only for \( j'' > j \).

Using Proposition 3.11 and the induction hypothesis we obtain

\[
\Pr \left[ T_{j',0}^t \right] = \sum_{j'' > j} \tilde{p}(j', j'') \Pr \left[ T_{j'',0}^{t} \right] \leq \sum_{j'' > j} \tilde{p}(j', j'') \Pr \left[ T_{j,0}^{t-1} \right] = \Pr \left[ T_{j,0}^{t-1} \right],
\]

since \( \sum_{j'' > j} \tilde{p}(j', j'') = 1. \)

□
3.2 The “Devil” Selection Rule For Schöning’s Algorithm

For all $k \geq 3$ and $n \geq 2k$, we will describe a $k$-CNF formula with $n$ variables such that the expectation of the number of runs of Schöning’s Algorithm on a particular formula is at least $\frac{1}{\text{poly}(n)} \left( \frac{k}{2(k-1)} \right)^n$.

Let $V$ be a set of boolean variables, such that $|V| = n$, and let $F_{n,k}^{\text{dev}}$ be the $k$-CNF formula over $V$ that contains all possible $k$-clauses satisfied by $1_V$. In other words, $F_{n,k}^{\text{dev}}$ is the set of all $k$-clauses of type at least 1. It is very easy to see that $1_V$ is the only satisfying assignment for $F_{n,k}^{\text{dev}}$. Recall that the type of a clause is simply the number of positive literals that appear in that clause.

We will now examine the execution of Schöning’s Algorithm on this formula. We will consider a modified version of the algorithm which behaves in the same way as the original, but at each step computes some useful values. Also, this algorithm is constructed to work only on the formula $F_{n,k}^{\text{dev}}$, and we assume that it possesses knowledge of the satisfying assignment.

The strategy of the “devil” will be to choose a clause of type 1 whenever possible, and any other clause otherwise. This is made more exact with the following statement.

**Remark 3.17** Given an assignment $\alpha \in \{0,1\}^V$, there exist unsatisfied clauses of type 1 if and only if $0 < d(\alpha, 1_V) \leq n - k + 1$.

Clearly, we can always find an unsatisfied clause of type 1 if $d(\alpha, 1_V) \leq n - k + 1$, since we can find $k - 1$ variables that are mapped to 1 in the assignment, and one which is mapped to 0, and out of those build an unsatisfied clause, which exists in $F_{n,k}^{\text{dev}}$.

We will now assume, for simplicity, that the variables in $V$ are totally ordered.

**Notation.** Given a clause $C$, let $\xi(C)$ be the smallest (with respect to ordering) variable that appears as a positive literal in $C$.

First of all, we must convince ourselves that Procedure Sch-Devil behaves as the usual Schöning’s Algorithm applied to $F_{n,k}^{\text{dev}}$. This is so because at each step, the next assignment is chosen using the same rules.

Furthermore, there is a link between the current assignment $\alpha$ and the variable $X$. Let $\alpha_i$ be the value of $\alpha$ in step $i$, and $X_i$ the value of $X$ at the same time step. In case the algorithm finishes with $i < t$, we extend $\alpha_j = 1_V$ and $X_j = 0$ for all $i < j \leq t$. We have the following lemma.

**Lemma 3.18** In algorithm Sch-Devil, for all $0 \leq i \leq t$ we have that $X_i \leq d(\alpha_i, 1_V)$. 

38
The “Devil” Selection Rule For Schöning’s Algorithm

Procedure Sch-Devil\( (n, k, t) \)
\[
i \leftarrow 0; \\
\alpha \in \text{rand}\{0, 1\}^V; \\
X = d(\alpha, 1_V); \\
\text{while } i < t \text{ and } \alpha \neq 1_V \text{ do} \\
i \leftarrow i + 1; \\
\text{if } d(\alpha, 1_V) \leq n - k + 1 \text{ then} \\
P_{\text{dev}}(n, k, \alpha) \\
P_{\text{dev}}(n, k, \alpha) \\
v \in \text{rand}\ \text{var}(C); \\
\alpha(v) \leftarrow 1 - \alpha(v); \\
\text{if } X \geq n - k \text{ then} \\
X \leftarrow X - 1 \\
\text{else} \\
\text{if } X > 0 \text{ then} \\
\text{if } v = \xi(C) \text{ then} \\
X \leftarrow X - 1; \\
\text{else} \\
X \leftarrow X + 1;
\]

Proof We use induction, where the base case is obvious. We have to check whether the condition is preserved at every step. Notice that \( d(\alpha_i, 1_V) \) can either increase or decrease by 1 at every step. We distinguish the following cases:

- If \( X_i \geq n - k \), since \( X \) will be always decremented, the condition will hold for \( i + 1 \) as well.

- If \( 0 < X_i < n - k \) and \( d(\alpha_i, 1_V) \geq X_i + 2 \), then the condition will continue to hold, because each term changes by \( \pm 1 \).

- Otherwise, given that \( 0 < X_i < n - k \), we have that \( d(\alpha_i, 1_V) \leq n - k + 1 \), so \( C \) is a clause of type 1. In this case, \( \xi(C) \) is the only positive literal, and flipping that variable is the only chance for the Hamming distance to decrease. This is, however, exactly the case when \( X \) decreases, so \( X \) and the Hamming distance will either both increase or both decrease. Because of this, the condition for \( i + 1 \) will hold.

- The case \( X_i = 0 \) is trivially clear.

The following statement is easy to check, by just examining the if-branches of the algorithm.
Proposition 3.19 If we view the $X_i$ as random variables, and set $t = \infty$, then \( \{X_i\}_{i \geq 0} \) is a Markov chain given by the generalized transition function (for all $j > 0$ and $j' \geq 0$):

\[
\tilde{p}^{\text{dev}}_{n,k}(j, j') = \begin{cases} 
1/k & \text{if } 0 < j < n - k \text{ and } j' = j - 1, \\
1 - 1/k & \text{if } 0 < j < n - k \text{ and } j' = j + 1, \\
1 & \text{if } j \geq n - k \text{ and } j' = j - 1, \\
1 & \text{if } j = 0 \text{ and } j' = 0, \\
0 & \text{otherwise.}
\end{cases}
\]

This coupling argument implies that the probability that the satisfying assignment is found in $t$ steps, given that the starting assignment is at Hamming distance $j$ from $1^V$ is at least the probability of the Markov chain to reach state 0 from state $j$ within $t$ steps, namely $\Pr \left[ T^t_{j \rightarrow 0}; \tilde{p}^{\text{dev}}_{n,k} \right]$. Note that we will consider now transcripts corresponding to different transition functions, and the reason for this non-standard notation is to make explicit which transition functions we use.

Since the Markov chain is clearly regular at all $j$ (see Definition 3.2), we are able to apply Lemma 3.15 and obtain the bound

\[
\Pr \left[ T^t_{j \rightarrow 0}; \tilde{p}^{\text{dev}}_{n,k} \right] \leq t \Pr \left[ \overline{T}^t_{j \rightarrow 0}; \tilde{p}^{\text{dev}}_{n,k} \right].
\]

Additionally, we consider the state-space homogenous Markov chain that we used in Section 1.6, and call $\tilde{p}'_k$ its generalized transition function. We notice that up to state $j < n - k$ these transitions coincide, and since we are considering transcripts bounded to the right, we obviously have

\[
\Pr \left[ \overline{T}^t_{j \rightarrow 0}; \tilde{p}'_k \right] = \Pr \left[ \overline{T}^t_{j \rightarrow 0}; \tilde{p}'_k \right].
\]

It is clear that $\Pr \left[ \overline{T}^t_{j \rightarrow 0}; \tilde{p}'_k \right]$ is upper bounded by the probability obtained in Lemma 1.6 (which would correspond to the case $t \rightarrow \infty$), which is exactly $(\lambda_{1/k}^{\text{sch}})^j$, where $\lambda_{1/k}^{\text{sch}} = \frac{1}{1 - 1/k}$.

\[
\Pr \left[ T^t_{j \rightarrow 0}; \tilde{p}^{\text{dev}}_{n,k} \right] \leq t \left( \lambda_{1/k}^{\text{sch}} \right)^j.
\]

Then, since $\Pr [X_0 = j] = \frac{1}{2^n} \binom{n}{j}$, the probability of reaching 0 in $3n$ steps is given by

\[
\Pr [X_{3n} = 0] = \frac{1}{2^n} \sum_{0 \leq j \leq n} \binom{n}{j} \Pr \left[ T^{3n}_{j \rightarrow 0}; \tilde{p}^{\text{dev}}_{n,k} \right].
\]
For all \( j \leq n/2 \), \( \tilde{p} \) is regular at \( j \), and we can apply Lemma 3.16, and obtain

\[
\Pr\left[T_{j;0;\tilde{p}^{\text{dev}}_{n,k}} \right] \geq \Pr\left[T_{j;0;\tilde{p}^{\text{dev}}_{n,k}} \right]
\]

\[
\Pr[X_{3n} = 0] \leq \frac{1}{2^n} \sum_{0 \leq j \leq n/2} 2 \cdot \binom{n}{j} \Pr\left[T_{j;0;\tilde{p}^{\text{dev}}_{n,k}} \right]
\]

\[
\leq \text{poly}(n) \frac{1}{2^n} \sum_{0 \leq j \leq n} \binom{n}{j} \left(\lambda_{1/k}^{\text{sch}}\right)^j
\]

\[
= \text{poly}(n) \left(1 + \lambda_{1/k}^{\text{sch}}\right)^n.
\]

This means that the expected number of calls of Schöning’s Algorithm on the formula \( F^{\text{dev}}_{n,k} \) is at least \( \frac{1}{\text{poly}(n)} \left(2^{(k-1)/k}\right)^n \), which is, up to polynomial factors, what we have obtained as a general lower bound in Section 1.6.

### 3.3 The “Random” Selection Rule For Schöning’s Algorithm

Just as in the previous case, we will describe a set of formulas \( F^{\text{rand}}_{n,k} \) for \( k \geq 3 \) and \( n \geq 2k \) such that the expected number of calls to procedure Sch is

\[
\Omega\left(\text{poly}(n) \left(\frac{2^{(k-1)/k}}{k^{1+2\epsilon}}\right)^n\right)
\]

for all \( \epsilon > 0 \).

To prove this, we fix \( 0 < \epsilon < 1 \) and \( n_0 = 2^k/\epsilon \), and consider only the case \( n > n_0 \).

Let \( V = \{v_1, \ldots, v_n\} \) be a set of \( n \) boolean variables. We define \( F^{\text{rand}}_{n,k} \) to be the \( k \)-CNF formula over \( V \) that contains all possible \( k \)-clauses of type 1, plus all \( k \)-clauses of type \( \geq 2 \) that contain just the variables \( v_1, \ldots, v_k \). There will be exactly \( k\binom{n}{k} \) clauses of type 1, and \( \sum_{j=2}^{k} \binom{k}{j} = 2^k - k - 1 \) clauses of type \( \geq 2 \).

**Proposition 3.20** For \( k \geq 3 \) and \( n \geq k \), the formula \( F^{\text{rand}}_{n,k} \) has only one satisfying assignment, namely \( 1_V \).

**Proof** Clearly the assignment \( 1_V \) is satisfying. Let \( \alpha \) be an assignment at Hamming distance \( j > 0 \) from \( 1_V \). Depending on \( j \), we distinguish the following cases:
3. Worst-Case Analysis

- If \( 0 < j \leq n - k \), then there are at least \( k - 1 \) variables mapped to 1 under \( \alpha \), and at least one variable mapped to 0. With these, we can build a clause of type 1 which is not satisfied by \( \alpha \), and this clause will be in \( F_{n,k}^{\text{rand}} \), since all clauses of type 1 are contained. Thus \( \alpha \) does not satisfy \( F_{n,k}^{\text{rand}} \).

- If \( j > n - k \), then at least one of the variables \( v_1, \ldots, v_k \) is mapped to 0, and thus we can find a clause of type \( \geq 1 \) which contains exactly variables \( v_1, \ldots, v_k \) and which is not satisfied by \( \alpha \). This clause will also be contained in \( F_{n,k}^{\text{rand}} \), so \( \alpha \) will not be satisfying. \( \square \)

As we have already done for the “devil” selection rule, we will exhibit an algorithm, Sch-Random, that behaves the same way as Schöning’s, but also provides a coupling between a Markov chain and the Hamming distance between the current assignment and \( 1_V \). We also use the definition of \( \xi(C) \) from the last section.

Also, given an assignment \( \alpha \) we can easily compute the number the number \( N_{\alpha}^{(i)} \) of violated clauses of type \( i \) by counting. Let \( N_{\alpha} \) be the total number of violated clauses.

Again, we easily see that the extended algorithm Sch-Random() behaves in the same manner as the original, and we prove the following results, which provide coupling to a Markov chain.

**Lemma 3.21** In algorithm Sch-Random, for all \( 0 \leq i \leq t \) we have that \( X_i \leq d(\alpha_i, 1_V) \).

**Proof** Again, the base case is obvious.

- If \( X_i > n/2 \), then since \( X \) will be always decremented, the condition will hold for \( i + 1 \) as well.

- If \( 0 < X_i < n/2 \), and \( d(\alpha_i, 1_V) > n/2 + 2 \), then the condition will continue to hold, because both \( X_i \) and the Hamming distance change with \( \pm 1 \), and their initial difference is greater than 2.

- Otherwise, given that \( 0 < X_i < n/2 \), if \( C \) is a type 1 clause, then \( \xi(C) \) is the only positive literal in \( C \), and flipping that variable is the only chance for the Hamming distance to decrease. The same condition causes \( X \) to decrease, so \( X \) can only increase when the Hamming distance increases. Because of this, the condition for \( i + 1 \) will hold.

- If \( 0 < X_i < n/2 \) and \( C \) is not a type 1 clause, then \( X \) always decreases, so the condition for \( i + 1 \) holds as well.

- The case \( X_i = 0 \) is trivially clear. \( \square \)
**Procedure** Sch-Random(n, k, t, ε)

\[
i \leftarrow 0;
\]
\[
\alpha \in \text{rand}\ \{0,1\}^V;
\]
\[
X = d(\alpha, 1_V);
\]

**while** \( i < t \) and \( \alpha \neq 1_V \) **do**

\[
i \leftarrow i + 1;
\]
\[
C \in \text{rand vlt } \left( \text{rand}_{n,k}, \alpha \right);
\]
\[
v \in \text{rand var } (C);
\]
\[
\alpha(v) \leftarrow 1 - \alpha(v);
\]

Compute \( N_a^{(1)}, N_a \) and \( N_a^{(\geq 2)} = N_a - N_a^{(1)} \);

\[
\zeta \leftarrow \frac{1}{(1/k+\epsilon)N_a-N_a^{(2)}};
\]

**if** \( X > n/2 \) **then**

\[
X \leftarrow X - 1
\]

**else**

**if** \( X > 0 \) **then**

**if** \( d(\alpha, 1_V) > n/2 + 2 \) **then**

\[
\tau \in \text{rand } (0,1); \text{ if } \tau < 1/k + \epsilon \text{ then } X \leftarrow X - 1
\]

**else**

\[
X \leftarrow X + 1
\]

**else**

**if** \( C \in \text{vlt } \left( \text{rand}_{n,k}, \alpha \right) \) **then**

**if** \( v = \tilde{\zeta}(C) \) **then**

\[
X \leftarrow X - 1
\]

**else**

\[
\tau \in \text{rand } (1/2, 1);
\]

**if** \( \tau < \tilde{\zeta} \) **then**

\[
X \leftarrow X - 1
\]

**else**

\[
X \leftarrow X + 1
\]

**else**

\[
X \leftarrow X - 1
\]
Lemma 3.22 Considering the infinite version of Sch-Random, we have that \( \{X_i\}_{i \geq 0} \) is a Markov chain given by the generalized transition function

\[
P_{n,k}^{\text{rand},\varepsilon}(j, j') = \begin{cases} 
1/k + \varepsilon & \text{if } 0 < j \leq n/2 \text{ and } j' = j - 1, \\
1 - 1/k - \varepsilon & \text{if } 0 < j \leq n/2 \text{ and } j' = j + 1, \\
1 & \text{if } j > n/2 \text{ and } j' = j - 1, \\
1 & \text{if } j = 0 \text{ and } j' = 0, \\
0 & \text{otherwise}.
\end{cases}
\]

Proof Examining the if-branches, we see that the Markov condition is obvious for all cases except the one defined by \( 0 < X_i \leq n/2 \) and \( d(\alpha, 1_V) \leq n/2 + 2 \). For this last case we provide a proof.

Assuming that \( \zeta - 1/k \geq 0 \), using Bayes rule, for \( 0 < j \leq n/2 \) we have that

\[
\Pr[X_{i+1} = j - 1 \mid X_i = j] = \\
= \Pr[X_{i+1} = j - 1 \mid X_i = j, C \in \text{vlt}_1(F_{n,k}^{\text{dev}}, \alpha)] \Pr[C \in \text{vlt}_1(F_{n,k}^{\text{dev}}, \alpha)] + \\
\Pr[X_{i+1} = j - 1 \mid X_i = j, C \notin \text{vlt}_1(F_{n,k}^{\text{dev}}, \alpha)] \Pr[C \notin \text{vlt}_1(F_{n,k}^{\text{dev}}, \alpha)] \\
= \left( \frac{1}{k} + \zeta - \frac{1}{k} \right) \frac{N_{\alpha}^{(1)}}{N_{\alpha}} + 1 \cdot \frac{N_{\alpha}^{(\geq 2)}}{N_{\alpha}} \\
= \frac{\zeta N_{\alpha}^{(1)} + N_{\alpha}^{(\geq 2)}}{N_{\alpha}} \\
= \frac{(1/k + \varepsilon) N_{\alpha} - N_{\alpha}^{(\geq 2)} + N_{\alpha}^{(\geq 2)}}{N_{\alpha}} \\
= (1/k + \varepsilon).
\]

For this to hold, we need the fact that \( \zeta \geq 1/k \). From the definition of \( \zeta \), this is equivalent to

\[
\left( \frac{1}{k} + \varepsilon \right) N_{\alpha} - N_{\alpha}^{(\geq 2)} \geq \frac{1}{k} N_{\alpha}^{(1)},
\]

which, given the fact that \( N_{\alpha}^{(\geq 2)} = N_{\alpha} - N_{\alpha}^{(1)} \), reduces to

\[
\varepsilon N_{\alpha} \geq N_{\alpha}^{(\geq 2)} - \frac{1}{k} N_{\alpha}^{(\geq 2)}.
\]

We use now the fact that \( N_{\alpha}^{(\geq 2)} < 2^k \). This is so because only the first \( k \) variables occur in clauses of type \( \geq 2 \), so the number of such clauses is less
than $2^k$. We also use the fact that for $j < n/2 + 2$, we have that $N_{\alpha} > n$ (in fact it is usually much larger). This happens because $N_{\alpha} \leq N_{\alpha}^{(1)} = j(n-j)$, and for any reasonably large $n$, the worst-case binomial coefficient $(n/2 - 2)$ is larger than $n$.

Since $n > 2^k/\epsilon$ (initial assumption), we deduce that $\epsilon > 2^k/n$, so we have the following chain of inequalities:

$$\epsilon N_{\alpha} \geq \frac{2^k}{n} n = 2^k > N_{\alpha}^{(2)}.$$ 

This obviously is much stronger than Inequality 3.4, and thus completes the proof. □

Since the Markov chain associated to $p_{n,k}^{\text{rand, } \epsilon}$ is coupled to the run of Schöning’s Algorithm on $F_{n,k}^{\text{rand}}$, we have that the probability that a satisfying assignment is found after $t$ steps is upper bounded by

$$\Pr[X_t = 0] = \Pr[T_{f,0}^t; p_{n,k}^{\text{rand, } \epsilon}].$$

As in the previous section, we have that

$$\Pr[T_{f,0}^t; p_{n,k}^{\text{rand, } \epsilon}] \leq t \Pr[T_{f,0}^t; p_{n,k}^{\text{rand, } \epsilon}]$$

$$\leq t \left( \frac{\lambda_{\text{sch}}}{1/k + \epsilon} \right)^j$$

$$= t \left( \frac{1/k + \epsilon}{1 - \frac{1}{k} - \epsilon} \right)^j$$

$$= t \left( \frac{1 + k\epsilon}{k - 1 - k\epsilon} \right)^j$$

$$\leq t \left( \frac{(1 + k\epsilon)^2}{k - 1} \right)^j,$$

where we have used that $\frac{1}{x-k\epsilon} = \frac{1}{x} + \frac{k\epsilon}{x(x-k\epsilon)} \leq \frac{1+k\epsilon}{x}$. Using the same reasoning as in the previous section, we obtain
\begin{align*}
\Pr \left[ X_{3n}^{(F_{n,k}^{\text{rand}})} = 0 \right] & \leq \text{poly}(n) \frac{1}{2^n} \sum_{0 \leq j \leq n/2} \binom{n}{j} \left( \frac{(1+k\varepsilon)^2}{k-1} \right)^j \\
& \leq \text{poly}(n) \frac{1}{2^n} \sum_{0 \leq j \leq n} \binom{n}{j} \left( \frac{(1+k\varepsilon)^2}{k-1} \right)^j \\
& = \text{poly}(n) \left( 1 + \frac{(1+k\varepsilon)^2}{(k-1)} \right)^n \\
& = \text{poly}(n) \left( \frac{k-1 + (1+k\varepsilon)^2}{2(k-1)} \right)^n \\
& \leq \text{poly}(n) \left( \frac{k(1+2\varepsilon)}{2(k-1)} \right)^n.
\end{align*}

Then the expected number of invocations of Schöning’s Algorithm on the formula \( F_{n,k}^{\text{rand}} \) is at least \( \frac{1}{\text{poly}(n)} \left( \frac{2(k-1)}{n(1+2\varepsilon)} \right)^n \).

### 3.4 The “Devil” Selection Rule For the Modified Algorithm

We will consider now the modified algorithm (see Section 2.1) with parameter \( \rho \in (0,1) \). We use the same formula \( F_{n,k}^{\text{dev}} \) as for Schöning’s algorithm and prove that for all \( k \geq 3 \) and \( n \geq 8k \), the expectation of the number of runs of the modified Schöning’s algorithm using a worst-case (“devil”) selection rule on \( F_{n,k}^{\text{dev}} \) is at least \( \frac{1}{\text{poly}(n)} \left( \frac{2(k-1)}{n(1+2\varepsilon)} \right)^n \). Each run should be invoked by Sch-Variant\( (F_{n,k}^{\text{dev}}, t, \rho) \), where \( t = \beta n \) for some \( \beta \) (this will prove irrelevant, since we allow for any polynomial factors anyway). We start by changing the auxiliary algorithm to achieve coupling for the Modified Algorithm, obtaining Procedure Mod-Devil.

As before, the coupling argument is provided by the following two statements.

**Lemma 3.23** In algorithm Mod-Devil, for all \( 0 \leq i \leq t \) we have that \( X_i \leq d(\alpha, 1_V) \).

**Proof** We observe that in general \( d(\alpha, 1_V) \) can decrease by at most \( k \) and increase by at most \( k-1 \) at every step. If a clause of type 1 is picked, then it can decrease by at most 1 at every step.
The “Devil” Selection Rule For the Modified Algorithm

**Procedure** Mod-Devil($n, k, t, \rho$)

1. $i \leftarrow 0$;
2. $\alpha \in_{\text{rand}} \{0, 1\}^V$;
3. $X = d(\alpha, 1_v)$;
4. **while** $i < t \text{ and } \alpha \neq 1_v$ **do**
5. 1. $i \leftarrow i + 1$;
6. 2. **if** $d(\alpha, 1_v) \leq n - k + 1$ **then**
7. 3. $C \in \text{vlt}_1 \left( F_{n,k}^{\text{dev}}, \alpha \right)$
8. 4. **else**
9. 5. $C \in \text{vlt} \left( F_{n,k}^{\text{dev}}, \alpha \right)$
10. 6. $DX = 0$;
11. 7. **for all** $v \in \text{var}(C)$ **do**
12. 8. 1. $r \in_{\text{rand}} (0, 1)$;
13. 9. 2. **if** $r > \rho$ **then**
14. 10. 3. $\alpha(v_h) = 1 - \alpha(v_h)$;
15. 11. 4. **if** $v = \tilde{\zeta}(C)$ **then**
16. 12. 5. $DX \leftarrow DX - 1$
17. 13. 6. **else**
18. 14. 7. $DX \leftarrow DX + 1$
19. 8. **if** $X \geq n - 3k$ **then**
20. 9. 1. $X \leftarrow X - k$
21. 10. **else**
22. 11. 2. **if** $X > 0$ **then**
23. 12. 3. $X \leftarrow X + DX$

- If $X_i \geq n - 3k$, since $X$ will be always decremented by $k$, the condition will hold for $i + 1$ as well.

- If $0 < X_i < n - 3k$, and $d(\alpha_i, 1_v) > n - k$, then the condition will continue to hold, because in the worst case $X$ increases by $k - 1$, and the Hamming distance decreases by $k$.

- Otherwise, given that $0 < X_i < n - 3k$ and $d(\alpha_i, 1_v) \leq n - k$, there exist violated clauses of type 1, so $C$ is a clause of type 1. In this case, $\tilde{\zeta}(C)$ is the only positive literal, and we observe that at the end of the for loop, $DX$ is exactly the variation in the Hamming distance.

- The case $X_i = 0$ is trivially clear. □

**Remark 3.24** If we view the $X_i$ as random variables, we can view \{X_i\}_{i \geq 0} as a
Markov chain given by the generalized transition function given below.

\[
\tilde{p}_{n,k}^{\text{dev2}}(j, j') = \begin{cases} 
(k-1)p^{2+\delta}(1-p)^{k-2-\delta} + (k-1)p^{\delta}(1-p)^{k-\delta} & \text{if } 0 < j < n - 3k \text{ and } j' = j + \delta, -1 \leq \delta \leq k - 1, \\
1 & \text{if } j \geq n - 3k \text{ and } j' = j - k, \\
1 & \text{if } j = 0 \text{ and } j' = 0, \\
0 & \text{otherwise.}
\end{cases}
\]

The rest of the argument is almost the same as the one in Section 3.2, and we just outline the differences:

- For all \( j < n - 4k \), the Markov chain corresponding to \( \tilde{p}_{n,k}^{\text{dev2}} \) is regular at \( j \). This is so, since the only possible transitions that go more than one step towards 0 can be found at \( j \geq 3k \), and there the transitions go \( k \) steps towards 0. Since \( n > 8k \), we also have that the Markov chain is regular for all \( j \leq n/2 \), and this is why we required \( n > 8k \) in the first place.

- The Markov chain as defined above has the same transitions for \( j < n/2 \) as the Markov chain considered in the analysis of the modified algorithm (see Section 2.5). We obtain

\[
\Pr\left[ T_{j,0}^{i} \tilde{p}_{n,k}^{\text{dev}} \right] = \left( \lambda_{\varrho,k} \right)^{i}.
\]

- The rest of the analysis is the same, with the conclusion that

\[
\Pr\left[ X_{i}^{(\tilde{p}_{n,k}^{\text{dev}})} = 0 \right] \leq \text{poly}(n) \left( \frac{1 + \lambda_{\varrho,k}}{2} \right)^{n},
\]

since \( t = \text{poly}(n) \).

### 3.5 The “Random” Selection Rule For the Modified Algorithm

Again, the main goal is to use a family of formulas such that the modified algorithm using a selection rule that picks a random unsatisfied clause, needs in expectation at least \( \frac{1}{\text{poly}(n)} \left( \frac{1 + \lambda_{\varrho,k}}{2} \right)^{n} \), where \( \varepsilon > 0 \) can be arbitrarily chosen. We will consider the formulas \( f_{n,k}^{\text{rand}} \), which we have considered before in Section 3.3. As in the previous section, each run should be invoked by
Sch-Variant($F_{\text{dev}}^{n,k}, t, \rho$), where $t = \beta n$ for some $\beta$ (which of course depends on $\rho$ and $k$, but we consider it fixed for our purposes here). In this case, however, $\beta$ will play a role in the analysis. Furthermore, we only allow large $n$, satisfying the following condition (this is fine, since we are analyzing the asymptotic behaviour, anyway):

$$n > \max \left( \frac{2e\beta k}{\varepsilon} \left( \lambda_{\rho,k}^{\text{var}} \right)^{-2k/\varepsilon}, 2k + 8 \right)$$

(3.5)

The main problem in this case is the fact that at every step there is a small probability that a clause of type $\geq 2$ is chosen. If we were to use the same method as in the previous sections, we would have to allow transitions that go more than one state towards 0, which would mean that our results using transcripts are unapplicable, since we would not be able to guarantee that the Markov chain is regular at any $j$.

To overcome this, we start our Markov chain not at the state corresponding to the Hamming distance between $\alpha$ and $1_V$, but in a state shifted with $\lceil \varepsilon n \rceil$ towards 0 (or 0 if the Hamming distance is already too close and we would obtain a negative state). Furthermore, we introduce variables $B_i$ which are 1 if a clause of type $\geq 2$ has been picked.

Let $B = \sum_{i=1}^{t} B_i$, where for $i > i_{\text{max}}$, $B_i$ is defined as a Bernoulli variable that is 1 with probability $1/n$, and which is independent from any other variables we use. We have the following result:

**Lemma 3.25** If $B < \frac{\varepsilon n}{2k}$, then for all $0 \leq i \leq t$ we have that

$$X_i \leq d(\alpha_i, 1_V) - \lceil \varepsilon n \rceil + k \sum_{i'=1}^{i} B_{i'}.$$

**Proof** In case $X_0 \leq \lceil \varepsilon n \rceil$, then $X_i = 0$ for all $0 \leq i \leq t$, and the claim holds. For the case $X_0 > \lceil \varepsilon n \rceil$, we use induction. The base case is clearly true, from the way $X$ is initialized. For the rest, it is sufficient to prove the following:

$$X_i - X_{i-1} \leq d(\alpha_i, 1_V) - d(\alpha_{i-1}, 1_V) + 2kB_i.$$  

(3.6)

We distinguish the following cases:

- In case $B_i = 1$, we know $X_i - X_{i-1} \leq k$ and $d(\alpha_i, 1_V) - d(\alpha_{i-1}, 1_V) \geq -k$, and clearly Inequality 3.6 holds.
- If $B_i = 0$ and $X_i \geq 3n/4$, since $X$ will be always decremented by $k$, the condition will hold for $i + 1$ as well.
3. Worst-Case Analysis

Procedure Mod-Random\((n, k, t, \varepsilon, \rho)\)

\[
i \leftarrow 0;
\alpha \in \text{rand} \{0,1\}^V;
X = \max (d(\alpha, 1_V, -) \left[\varepsilon n\right], 0);
\]

\[
\text{while } i < t \text{ and } \alpha \neq 1_V \text{ do}
\]

\[
i \leftarrow i + 1;
C \in \text{rand} vlt \left(F_{n,k}^\alpha\right);
DX = 0;
\text{for all } v \in \text{var} (C) \text{ do}
\]

\[
r \in \text{rand} (0,1);
\text{if } r > \rho \text{ then}
\]

\[
\alpha(v_h) = 1 - \alpha(v_h);
\]

\[
\text{if } v = \xi(C) \text{ then}
\]

\[
DX \leftarrow DX - 1
\]

\[
\text{else}
\]

\[
DX \leftarrow DX + 1
\]

\[
\text{if } X \geq 3n/4 \text{ then}
\]

\[
X \leftarrow X - k
\]

\[
\text{else}
\]

\[
\text{if } X > 0 \text{ then}
\]

\[
X \leftarrow X + DX
\]

\[
\text{if } d(\alpha, 1_V) \geq 7n/8 \text{ or } \alpha = 1_V \text{ then}
\]

\[
r \in (0,1);
\]

\[
\text{if } r < 1/n \text{ then}
\]

\[
B_i = 1
\]

\[
\text{else}
\]

\[
B_i = 0
\]

\[
\text{else}
\]

\[
\text{Compute } v = \frac{N_{\geq 2}}{N_n};
\]

\[
\text{if } C \notin vlt \left(F_{n,k}^\alpha\right) \text{ then}
\]

\[
B_i = 1
\]

\[
\text{else}
\]

\[
r \in (v,1);
\]

\[
\text{if } r < 1/n \text{ then}
\]

\[
B_i = 1
\]

\[
\text{else}
\]

\[
B_i = 0
\]
• If $B_i = 0$ and $0 < X_i < 3n/4$, and $d(\alpha_i, 1) \geq 7n/8 > 3n/4 + 2k$, then the condition will continue to hold, because in the worst case $X$ increases by $k - 1$, and the Hamming distance decreases by $k$.

• If $B_i = 0$ and $0 < X_i < 3n/4$ and $d(\alpha_i, 1) < 7n/8$, then since $B_i = 0$, a clause of type 1 has been chosen. In this case, $\tilde{\xi}(C)$ is the only positive literal, and we observe that at the end of the for loop, $DX$ is exactly the variation in the Hamming distance.

• The case $X_i = 0$ is trivially clear.

\textbf{Corollary 3.26} If $B < \frac{\eta}{2k}$, then for all $0 \leq i \leq t$ we have that $X_i \leq d(\alpha_i, 1)$.

\textbf{Remark 3.27} If we view the $X_i$ as random variables, we can view $\{X_i\}_{i \geq 0}$ as a Markov chain given by the generalized transition function given below.

$$p_{n,k}^{\text{rand}_2(j, j')} = \begin{cases} (k-1)^2 (1 - \rho)^{k - 2 - \delta} + (k-1)^\delta (1 - \rho) \rho^\delta & \text{if } 0 < j < 3n/4 \text{ and } j' = j + \delta, -1 \leq \delta \leq k - 1, \\ 1 & \text{if } j \geq 3n/4 \text{ and } j' = j - k, \\ 1 & \text{if } j = 0 \text{ and } j' = 0, \\ 0 & \text{otherwise.} \end{cases}$$

\textbf{Lemma 3.28} For $1 \leq i \leq t$, the random variables $B_i$ are independent, randomly distributed and with the same Bernoulli distribution $(1/k, 1 - 1/k)$.

\textbf{Proof} For the fact that they are independent, it is sufficient to observe that the random source for each $B_i$ is the clause $C$ and the local variable $r$, both of which are resampled in each round.

To check that the distribution is indeed the one specified, we only need to verify the case $0 < d(\alpha, 1) < 7n/8$, as the rest are trivial. Using Bayes’ rule, and the fact that $v = \Pr[C \in \text{vlt}_1 \left( F_{n,k}^{\text{dev}}, \alpha \right)]$, we obtain

$$\Pr[B_i = 1] = \Pr[C \in \text{vlt}_1 \left( F_{n,k}^{\text{dev}}, \alpha \right)] \Pr[B_i = 1 \mid C \in \text{vlt}_1 \left( F_{n,k}^{\text{dev}}, \alpha \right)] +$$

$$+ \Pr[C \notin \text{vlt}_1 \left( F_{n,k}^{\text{dev}}, \alpha \right)] \Pr[B_i = 1 \mid C \notin \text{vlt}_1 \left( F_{n,k}^{\text{dev}}, \alpha \right)]$$

$$= v \frac{1/n - v}{1 - v} + v = \frac{1}{n},$$

where we have used the fact that $\Pr[B_i = 1 \mid C \in \text{vlt}_1 \left( F_{n,k}^{\text{dev}}, \alpha \right)] = \frac{1}{1 - v}$.

But clearly this is only true if $v \leq 1/n$. The rest of the argument will be concerned with the proof of this inequality.
Setting \( j = d(a, 1, \nu) \), and the properties of \( F_{n,k}^{\text{rand}} \) developed in Section 3.3 we have the following sequence of inequalities:

\[
\nu = \frac{N_k^{(2)}}{N_n} \leq \frac{N_k^{(2)}}{N_n^{(1)}} \leq \frac{2^k}{j(k-1)} \leq \frac{2^k}{(n-j)^2} \leq \frac{2^k}{(n-7n/8)^2} \leq \frac{2^k}{(n/8)^2} \leq \frac{2^k}{n^2/256} = \frac{2^{k+8}}{n^2}.
\]

Using the restriction given by Inequality 3.5, namely that \( n > 2^{k+8} \), we deduce that \( \nu \leq 1/n \).

**Lemma 3.29** We have that \( \Pr[2kB > \epsilon n] < (\lambda_{\text{var}}^n) \).

**Proof** The expectation of \( B \) is \( \beta/n = \beta \). We prepare the expression in order to apply Chernoff bounds:

\[
\Pr[2kB > \epsilon n] = \Pr\left[B > \frac{\epsilon n}{2kB} \beta\right].
\]

We use Theorem A.2 from the Appendix, in the form of Inequality A.3, with \( \delta + 1 = \frac{\epsilon n}{2kB} \), obtaining:

\[
\Pr[2kB > \epsilon n] < e^{-\beta} \left(\frac{e}{\epsilon n}ight)^{2k} \frac{\epsilon n}{2kB} ^{-2k}\epsilon n.
\]

Using again relation 3.5, we have that

\[
n > \frac{2\beta k}{\epsilon} \left(\lambda_{\text{var}}^n \rho_{k} \right)^{-2k/\epsilon}.
\]

From the last two equations, the claim follows.

Let \( \Pr[\text{succ}] \) be the probability of success for the algorithm. We state now the final result of this section.

**Theorem 3.30** The probability of success of Sch-Variant\( F_{n,k}^{\text{rand}}, \beta n, \rho \) is given by

\[
\Pr[\text{succ}] \leq \text{poly}(n) \left(1 + \lambda_{\text{var}}^n \rho_{k} \right)^{(1-\epsilon)n} / 2,
\]

for any \( \epsilon > 0 \).
The “Random” Selection Rule For the Modified Algorithm

Proof Using Bayes’ rule, we have
\[ \Pr[\text{succ}] = \Pr[2kB > \epsilon n] \cdot \Pr[\text{succ} \mid 2kB > \epsilon n] + \Pr[2kB \leq \epsilon n] \cdot \Pr[\text{succ} \mid 2kB \leq \epsilon n]. \tag{3.7} \]
\[ + \Pr[2kB \leq \epsilon n] \cdot \Pr[\text{succ} \mid 2kB \leq \epsilon n]. \tag{3.8} \]

We use now the (somewhat gross) approximations \( \Pr[2kB \leq \epsilon n] \leq 1 \) and \( \Pr[\text{succ} \mid 2kB > \epsilon n] \leq 1 \). Furthermore, note that due to Corollary 3.26, we also have \( \Pr[\text{succ} \mid 2kB \leq \epsilon n] \leq \Pr[X_{\beta n} = 0] \). Then Equation 3.7 implies

\[ \Pr[\text{succ}] \leq \Pr[2kB > \epsilon n] + \Pr[X_{\beta n} = 0]. \tag{3.9} \]

We will now compute \( \Pr[X_{\beta n} = 0] \), in a fashion similar to what we have done in the previous sections:
\[ \Pr[X_{\beta n} = 0] = \sum_{0 \leq j \leq n} \Pr[d(\alpha_0, 1_V) = j] \Pr[X_{\beta n} = 0 \mid d(\alpha_0, 1_V) = j]. \tag{3.10} \]

For \( j > \lceil \epsilon n \rceil \), we have
\[ \Pr[X_{\beta n} = 0 \mid d(\alpha_0, 1_V) = j] = \Pr[X_{\beta n} = 0 \mid X_0 = j - \lceil \epsilon n \rceil]. \]

Furthermore, for all \( j < 3n/4 - k \), Remark 3.27 guarantees that the generalized transition function is regular at \( j \). Then we can apply the same technique as in the previous sections to obtain
\[ \Pr[X_{\beta n} = 0 \mid X_0 = j - \lceil \epsilon n \rceil] \leq \beta n \left( \lambda_{\varrho,k} \right)^{j - \lceil \epsilon n \rceil}. \tag{3.11} \]

For \( j \leq \lceil \epsilon n \rceil \), we have that \( \Pr[X_{\beta n} = 0 \mid d(\alpha_0, 1_V) = j] = 1 \), since the starting state of \( X \) is 0. To have a more unitary treatment for the cases, we can see that Relation 3.11 is valid for this case as well.

Substituting this in Equation 3.10, and also applying the fact that the terms appearing there with \( j > n/2 \) are at most as large as the terms corresponding to \( n - j \), we finally obtain
\[ \Pr[X_{\beta n} = 0] \leq \sum_{0 \leq j \leq n/2} 2 \cdot \binom{n}{j} \text{poly}(n) \left( \lambda_{\varrho,k} \right)^{j - \lceil \epsilon n \rceil} \]
\[ \leq \text{poly}(n) \left( \lambda_{\varrho,k} \right)^{\lceil \epsilon n \rceil} \sum_{0 \leq j \leq n} \binom{n}{j} \left( \lambda_{\varrho,k} \right)^j \]
\[ = \text{poly}(n) \left( \lambda_{\varrho,k} \right)^{\lceil \epsilon n \rceil} \left( 1 + \frac{\lambda_{\varrho,k}}{2} \right)^n \]
\[ \leq \text{poly}(n) \left( 1 + \frac{\lambda_{\varrho,k}}{2} \right)^{(1 - \epsilon)n}. \]
3. Worst-Case Analysis

We use now Relation 3.9 and Lemma 3.29, concluding:

\[ \Pr[\text{succ}] = \text{poly}(n) \left( \frac{1 + \lambda \var_{\rho_k}}{\rho_k} \right)^{(1-c)n} . \]

\( \square \)
Chapter 4

Selection Rules of Type “Angel”

In this chapter we will discuss the existence of special selection rules, namely those that actively try to “help” Schöning’s Algorithm reach a satisfying assignment fast (we will have two alternative definitions of what “fast” means in this case, but they typically involve running the algorithm in polynomial time). It is intuitive that such selection rules contain some information about the satisfying assignments, in order to “guide” the run of the algorithm in the right direction.

4.1 More About Selection Rules

In the previous chapter we have analyzed two particular selection rules, which we called “devil” and “random”. We will now try to generalize the concept, and introduce three broad types of deterministic selection rules, namely those with memory, with clock and without memory. The simplest type is the selection rule without memory. Every time the selection rule is invoked by the algorithm, a violated clause is produced depending solely on the current assignment and the $k$-CNF formula. A selection rule with clock keeps track of the current iteration of the algorithm, and produces the violated clause depending on both this clock and the current assignment. The most general type of selection rule is the one with memory, which takes into account the full history of assignments that have been encountered so far by the algorithm.

Furthermore, to be able to analyze asymptotically the performance of such a selection rule, we have to consider formulas with an increasing number of variables. In this context we define a class of formulas as follows.

**Definition 4.1** A class of $k$-CNF formulas $\mathcal{F}$ is a set of formulas such that for any $n \in \mathbb{N}$ there is a formula $F \in \mathcal{F}$ such that $|\text{var}(F)| > n$. 

55
We are now ready to formally define the types of deterministic selection rules.

**Definition 4.2** A selection rule without memory for a class $\mathcal{F}$ of $k$-CNF formulas is a set of maps from assignments to clauses $\sigma^{(w)}_F : \{0,1\}^{\text{var}(F)} \to F \cup \{\emptyset\}$ indexed by $F \in \mathcal{F}$, with the condition that $\sigma^{(w)}_F(\alpha) \in \text{vlt}(F,\alpha)$ if $\alpha$ is not satisfying, and $\emptyset$ otherwise.

**Definition 4.3** A selection rule with clock for a class $\mathcal{F}$ of $k$-CNF formulas is a set of maps $\sigma^{(c)}_F : \{0,1\}^{\text{var}(F)} \times \mathbb{N} \to F \cup \{\emptyset\}$ indexed by $F \in \mathcal{F}$, with the condition that $\sigma^{(c)}_F(\alpha,i) \in \text{vlt}(F,\alpha)$ if $\alpha$ is not satisfying, and $\emptyset$ otherwise.

**Definition 4.4** A selection rule with memory for a class $\mathcal{F}$ of $k$-CNF formulas is a set of maps $\sigma^{(m)}_F : \mathcal{A}_{\text{var}(F)} \to F \cup \{\emptyset\}$ indexed by $F \in \mathcal{F}$, where $\mathcal{A}_{\text{var}(F)}$ is the set of nonempty strings of assignments to the variables that appear in $F$, subject to the following conditions:

- for each string of exactly one assignment $\alpha$, $\sigma^{(m)}_F(\alpha) \in \text{vlt}(F,\alpha)$ if $\alpha$ is not satisfying, and $\emptyset$ otherwise;
- if the string $\alpha_1 \ldots \alpha_i \in \mathcal{A}_{\text{var}(F)}$, with $i > 1$ and all $\alpha_j$ not satisfying, is such that for all $1 \leq j < i$ there exists $v \in \text{var}(\sigma^{(m)}_F(\alpha_1 \ldots \alpha_j))$ such that $\alpha_{j+1}$ is the same as $\alpha_j$, except the value of $v$ is flipped, then $\sigma^{(m)}_F(\alpha_1 \ldots \alpha_i) \in \text{vlt}(F,\alpha_i)$.

Given a selection rule with memory, if an assignment string $s \in \mathcal{A}_{\text{var}(F)}$ fulfills the last condition above, we call it a regular assignment string w.r.t that rule.

Naturally, the “random” selection rule from the last chapter does not fit into this formalism, since here we only consider deterministic selection rules. We can, however, consider choosing randomly a selection rule with clock, and it is easy to see that the behaviour will be exactly like using the “random” rule.

**Remark 4.5** A selection rule without memory can be transformed in a straightforward manner into a selection rule with clock, which will have the same behaviour, and likewise, a selection rule with clock can be transformed into one with memory.

**Lemma 4.6** Independent of the choice for the selection rule, for any formula $F$, $\text{Sch}(F, \infty)$ will terminate with probability 1. Moreover, the expectation of the number of iterations is finite (to be exact, it is always at most $n \cdot 3^n$, where $n = |\text{vbl}(F)|$).
Proof Let $Y$ be the number of iterations. The key to the proof lies in the following observation, valid for all $t \leq 0$:

$$\Pr[Y > t + n] \leq \left(1 - \left(\frac{1}{3}\right)^n\right)\Pr[Y > t]. \quad (4.1)$$

To see why this holds, it is enough if we observe that at any point in time and with any selection rule, with probability at least $1/3$, the algorithm will approach a satisfying assignment. With probability at least $\left(\frac{1}{3}\right)^n$, it will always make the “right” move in the next $n$ steps, and thereby reach a satisfying assignment. This amounts to stating that $\Pr[Y \geq t + n | Y > t] \geq \left(\frac{1}{3}\right)^n$, which proves Inequality 4.1.

We are now ready to compute the expectation of $Y$:

$$\mathbb{E}[Y] = \sum_{t=0}^{\infty} \Pr[Y \geq t]$$

$$= \sum_{t=0}^{n-1} \sum_{j=0}^{\infty} \Pr[Y \geq i + j \cdot n]$$

$$\leq \sum_{t=0}^{n-1} \sum_{j=0}^{\infty} \left(1 - \left(\frac{1}{3}\right)^n\right)^j \Pr[Y > i]$$

$$= \sum_{t=0}^{n-1} \Pr[Y > i] \cdot \sum_{j=0}^{\infty} \left(1 - \left(\frac{1}{3}\right)^n\right)^j$$

$$\leq n \cdot \frac{1}{1 - \left(1 - \left(\frac{1}{3}\right)^n\right)} = n \cdot 3^n. \quad \square$$

### 4.2 Types of Angels

We now want to have a criterion to discriminate the selection rules which are extremely beneficent for the runtime of the algorithm. To be in accordance to our nomenclature for selection rules that we have established so far, we will call those selection rules “angels”. We will have two types of angels, depending on how we limit the runtime of the algorithm.

**Definition 4.7** A selection rule for a class $\mathcal{F}$ of $k$-CNF formulas is a type 1 angel if there exist polynomials $f(n)$ and $p(n)$ such that for all formulas $F \in \mathcal{F}$ with $n = \|\text{var}(F)\|$, one run of $\text{Sch}(F, f(n))$ finds a satisfying assignment with probability at least $1/p(n)$.

Clearly using Lemma 1.9, in case we are using a type 1 angel, after polynomially many rounds of $\text{Sch}(F, f(n))$, a satisfying assignment will be found
with probability that vanishes exponentially. We now introduce a second type of angel.

**Definition 4.8** A selection rule for a class $F$ of $k$-CNF formulas is a type 2 angel if there exists a polynomial $f(n)$ such that for all formulas $F \in F$ with $n = |\text{var}(F)|$, one run of $\text{Sch}(F, \infty)$ finds a satisfying assignment in less than $f(n)$ steps in expectation.

**Definition 4.9** A universal angel (of either type) is a selection rule for the class of all $k$-CNF formulas which is an angel (of the corresponding type).

We do not know so far whether such a universal angel exists at all.

### 4.3 Relations Between Types of Angels

We will now show some results that relate the different types of angels. First, we will show that the notion of angel of type 2 is at least as “strong” as an angel of type 1.

**Proposition 4.10** Given a class $F$ of $k$-CNF formulas, an angel of type 2 is also an angel of type 1.

**Proof** Suppose $f(n)$ is the polynomial that upper-bounds the expectation of the number of steps needed for $\text{Sch}()$ to achieve success (cf. Definition 4.8). Then for all formulas $F \in F$ with $n = |\text{var}(F)|$, applying Markov’s inequality yields, for any $\kappa > 1$, that the probability that one run of $\text{Sch}(F, \kappa f(n))$ is successful is at least $1 - \frac{1}{\kappa^2}$. Thus we can use the polynomials $f'(n) = \kappa f(n)$ and $p(n) = 1 - \frac{1}{\kappa}$ in Definition 4.7, and conclude that that our initial type 2 angel is an angel of type 1 as well. □

**Proposition 4.11** Given a class $F$ of $k$-CNF formulas, if there exists an angel of type 2 with memory for $F$, then there exists also an angel of type 2 with clock for the same class of formulas.

**Proof** Let $\sigma^{(m)}$ be the angel of type 2 with memory, and let $Y$ be the random variable representing the number of iterations of $\text{Sch}()$ performed until a satisfying assignment has been found. Clearly $\Pr[Y = \infty] = 0$, since otherwise $E[Y]$ would not be finite. We will now describe a selection rule of type 2 with clock $\sigma^{(c)}$, and prove that the expected number of iterations performed by $\text{Sch}()$ with $\sigma^{(c)}$ is at most $E[Y]$. Let $Y'$ be the random variable that counts those iterations.

To describe $\sigma^{(c)}$, we need to specify the value it takes given the value $i$ of the clock and the current assignment $a$. We look through the all regular assignment strings $s$ of length $i$ ending in $a$, and pick the one which minimizes the
expectation of the number of steps needed given that so far the assignments in \( s \) have been produced. We call this assignment \( s_{a,i}^{\text{min}} \), and set \( \sigma_F^{(c)}(\alpha, i) \) to the violated clause indicated by \( \sigma_F^{(m)}(s_{a,i}^{\text{min}}) \).

In case no such \( s \) exists, we set \( \sigma_F^{(c)}(\alpha, i) \) to \( \emptyset \) if \( \alpha \) is satisfying, or to some arbitrary violated clause if not. To state this formally, let \( S_{a,i} \) be the set all regular assignment strings. Then we define \( s_{a,i}^{\text{min}} \) for the case where \( S_{a,i} \neq \emptyset \):

\[
s_{a,i}^{\text{min}} = \arg\min_{s \in S_{a,i}} \mathbb{E}[Y | s \text{ encodes the first assignments encountered}].
\]

If there are multiple choices for \( s_{a,i}^{\text{min}} \), any one will do. We now define the selection rule with clock:

\[
\sigma_F^{(c)}(\alpha, i) = \begin{cases} 
\sigma_F^{(m)}(s_{a,i}^{\text{min}}) & \text{if } S_{a,i} \neq \emptyset, \\
\emptyset & \text{if } \alpha \text{ is not satisfying for } F, \\
\text{any clause from } \text{vlt}(F, \alpha) & \text{otherwise.}
\end{cases}
\]

(4.2)

It can be easily seen that the last branch from the case distinction will never be reached by an actual run of \( \text{Sch}() \) with the selection rule \( \sigma^{(c)} \). This happens because all nonsatisfying reachable assignments in step \( i \) are also the end of some regular assignment string of length \( i \).

Our final goal is to prove \( \mathbb{E}[Y'] \leq \mathbb{E}[Y] \). To achieve this, we define some auxiliary selection rules with memory \( \sigma_F^{(i)} \), for \( i \geq 0 \), in the following manner:

\[
\sigma_F^{(i)}(s) = \begin{cases} 
\sigma_F^{(c)}(s_{|s|}, |s|) & \text{if } |s| \leq i, \\
\sigma_F^{(m)}(s_{a,i}^{\text{min}} \circ \text{suf}(s, |s| - i - 1)) & \text{if } |s| > i,
\end{cases}
\]

where \( s_{|s|} \) denotes the last element of \( s \), “\( \circ \)” denotes string concatenation, and the function \( \text{suf}(s, i) \) returns the suffix of \( s \) consisting of \( i \) elements. Intuitively, we think of \( \sigma_F^{(i)} \) as a selection rule that plays like \( \sigma_F^{(c)} \) for the first \( i \) rounds, and afterwards plays like \( \sigma_F^{(m)} \), where the first \( i + 1 \) assignments from the current history are replaced with the string \( s_{a,i+1}^{\text{min}} \circ \text{suf}(s, |s| - i - 1) \). We also define \( Y^{(i)} \) as the random variables counting the number of iterations of \( \text{Sch}() \) when running with \( \sigma_F^{(i)} \).

It is easy to see that \( \sigma_F^{(m)} = \sigma_F^{(0)} \) and thus \( Y = Y^{(0)} \). We will show now that \( \mathbb{E}[Y^{(i)}] \geq \mathbb{E}[Y^{(i+1)}] \). Since \( \sigma_F^{(i)} \) and \( \sigma_F^{(i+1)} \) behave identically up to iteration \( i \),
inclusively, we have
\[
\Pr\left[Y^{(i)} \leq i\right] = \Pr\left[Y^{(i+1)} \leq i\right], \quad \text{and}
\]
\[
\mathbb{E}\left[Y^{(i)} \mid Y^{(i)} \leq i\right] = \mathbb{E}\left[Y^{(i+1)} \mid Y^{(i+1)} \leq i\right].
\]

Then using Bayes’ rule for expectations, it is sufficient to convince ourselves that
\[
\mathbb{E}\left[Y^{(i)} \mid Y^{(i)} \geq i + 1\right] \geq \mathbb{E}\left[Y^{(i+1)} \mid Y^{(i+1)} \geq i + 1\right]. \quad (4.3)
\]

Let \( \alpha^{(i)}_j \) be the current assignment after \( j \) iterations, given that selection rule \( \sigma^{(i)}_j \) is used. Also, let \( s^{(i)}_j \) be the string formed by the current assignments from the first \( j \) iterations. We have
\[
\Pr[\alpha^{(i)}_{i+1} = \beta \mid \alpha^{(i)}_i = \alpha] = \Pr[\alpha^{(i+1)}_{i+1} = \beta \mid \alpha^{(i+1)}_i = \alpha]. \quad (4.4)
\]

We finally obtain (using the above equation and Relation 4.2)
\[
\mathbb{E}\left[Y^{(i)} \mid Y^{(i)} > i, \alpha^{(i)}_i = \alpha\right] =
\]
\[
\sum_{\beta \in \{0,1\}^{\text{var}(F)}} \Pr[\alpha^{(i)}_{i+1} = \beta \mid \alpha^{(i)}_i = \alpha] \mathbb{E}\left[Y^{(i)} \mid Y^{(i)} > i, \alpha^{(i)}_i = \alpha, \alpha^{(i+1)}_i = \beta\right]
\]
\[
\leq \sum_{\beta \in \{0,1\}^{\text{var}(F)}} \Pr[\alpha^{(i+1)}_{i+1} = \beta \mid \alpha^{(i+1)}_i = \alpha] \mathbb{E}\left[Y \mid Y > i, s^{(i)}_{i+1} = s^{\min}_{\alpha,i} \circ \beta\right]
\]
\[
= \mathbb{E}\left[Y^{(i+1)} \mid Y^{(i+1)} > i, \alpha^{(i+1)}_i = \alpha\right].
\]

By applying the Bayes’ rule for expectations, we obtain Inequality 4.3. So far we have showed that
\[
\mathbb{E}[Y] \geq \mathbb{E}[Y^{(1)}] \geq \mathbb{E}[Y^{(2)}] \geq \ldots \geq \mathbb{E}[Y^{(i)}] \geq \ldots
\]

We will now prove that \( \mathbb{E}[Y'] \) is less than all those terms, which will complete the proof. By the definition of expectation, \( \mathbb{E}[Y'] = \sum_{j \geq 1} j \Pr[Y' = j] \). We consider the partial sums
\[
f_i = \sum_{1 \leq j \leq i} j \Pr[Y' = j].
\]
Since $\Pr[Y' = j] = \Pr[Y(i) = j]$, for $j < i$, we have that $f_i \leq \mathbb{E} \left[ Y(i) \right]$. Since the $f_i$ are increasing, while $\mathbb{E} \left[ Y(i) \right]$ are decreasing, we conclude that $\mathbb{E} \left[ Y' \right]$, which is the limit of $f_i$, is less than all $\mathbb{E} \left[ Y(i) \right]$, and in particular less than $\mathbb{E} \left[ Y \right]$. □

We will now prove a similar result using type 1 angels.

**Proposition 4.12** Given a class $\mathcal{F}$ of $k$-CNF formulas, if there exists an angel of type 1 with memory for $\mathcal{F}$, then there exists also an angel of type 1 with clock for the same class of formulas.

**Proof** The outline of the proof is similar to the proof of the previous lemma, so we just point out the differences. Let $\sigma_F^{(m)}$ be an angel of type 1 with memory for $\mathcal{F}$, which means that there is a polynomial $f(n)$ such that Sch() running on a formula $F \in \mathcal{F}$ with $n$ variables finds a satisfying assignment within $f(n)$ steps with probability at least $1/p(n)$, where $p(n)$ is a fixed polynomial. We will exhibit a selection rule with clock $\sigma_F^{(c)}$ which is an angel of type 1 with the same parameters $f(n)$ and $p(n)$.

We define $s_{\alpha,i}^{\text{max}}$, this time only for $i \leq f(n)$, as the regular assignment string of length $i$ that maximizes the probability that a satisfying assignment is found within $f(n)$ steps:

$$s_{\alpha,i}^{\text{max}} = \arg \max_{s \in S_{\alpha,i}} \Pr[Y \leq f(n) \mid s \text{ encodes the first assignments encountered}],$$

and then define $\sigma_F^{(c)}(\alpha, i)$ as

$$\sigma_F^{(c)}(\alpha, i) = \begin{cases} 
\sigma_F^{(m)} \left( s_{\alpha,i}^{\text{max}} \right) & \text{if } i \leq f(n) \text{ and } S_{\alpha,i} \neq \emptyset, \\
\emptyset & \text{if } \alpha \text{ is not satisfying for } F, \\
\text{any clause from } \text{vlt}(F, \alpha) & \text{otherwise.}
\end{cases}$$

(4.5)

We then define $\sigma_F^{(i)}$ as in the proof of the previous lemma, but only for $0 \leq i \leq f(n)$. Then we notice that $\Pr[Y \leq f(n)] = \Pr[Y(0) \leq f(n)]$ and $\Pr[Y(f(n)) \leq f(n)] = \Pr[Y' \leq f(n)]$. The rest of the argument will be concerned with proving that $\Pr[Y(i) \leq f(n)] \leq \Pr[Y(i+1) \leq f(n)]$. Since

$$\Pr[Y(i) \leq f(n)] = \Pr[Y(i) \leq i] + \Pr[Y(i) \leq f(n) \mid Y(i) > i],$$

we will...
and the fact that $\Pr[Y(i) \leq i] = \Pr[Y(i+1) \leq i]$, then it suffices to prove the following:

$$\Pr[Y(i) \leq f(n) \mid Y(i) > i] \leq \Pr[Y(i+1) \leq f(n) \mid Y(i+1) > i].$$  \hspace{1cm} (4.6)

We finally obtain (using Equation 4.4, which holds in this case as well, and Relation 4.5)

$$\Pr[Y(i) \leq f(n) \mid Y(i) > i, a_i^{(i)} = \alpha] =$$

$$\sum_{\beta \in \{0, 1\}^{\var(F)}} \Pr[a_{i+1}^{(i)} = \beta \mid a_i^{(i)} = \alpha] \cdot \Pr[Y(i) \leq f(n) \mid Y(i) > i, a_i^{(i)} = \alpha, a_{i+1}^{(i)} = \beta]$$

$$= \sum_{\beta \in \{0, 1\}^{\var(F)}} \Pr[a_{i+1}^{(i)} = \beta \mid a_i^{(i)} = \alpha] \Pr[Y \leq f(n) \mid Y > i, s_{i+1}^{(i)} = s_{\min}^{\alpha, i} \circ \beta]$$

$$\geq \sum_{\beta \in \{0, 1\}^{\var(F)}} \Pr[a_{i+1}^{(i+1)} = \beta \mid a_i^{(i+1)} = \alpha] \Pr[Y \leq f(n) \mid Y > i, s_{i+1}^{(i)} = s_{\min}^{\beta, i+1}]$$

$$= \Pr[Y(i+1) \leq f(n) \mid Y(i+1) > i, a_i^{(i+1)} = \alpha],$$

which together with an application of Bayes' rule proves Inequality 4.6.

\[\square\]

**Proposition 4.13** Given a class $\mathcal{F}$ of $k$-CNF formulas, if there exists an angel of type 2 with clock for $\mathcal{F}$, then there exists also an angel of type 2 without memory for the same class of formulas.

**Proof** Let $\sigma^{(m)}$ be the angel of type 2 with clock. We fix a formula $F \in \mathcal{F}$, and define a selection rule $\sigma^{(w)}$ without memory. Let $Y$ be the random variable that counts the number of iterations needed to find a satisfying assignment using $\sigma^{(w)}$.

We first examine the quantities $h_{\alpha, i} = E[Y \mid Y > i, a_i = \alpha] - i$, for all assignments $\alpha$ that are not satisfying. Let $h_{\alpha} = \lim \inf_{i \leq 0} h(\alpha, i)$. Then since there are only finitely many clauses, there must a be a set $D_{\alpha} \subset \mathbb{Z}_+$ such that

- for all $i \in \mathbb{Z}_+$, there is $i' \in D$ such that $h(\alpha, i') \leq h(\alpha, i)$; we set $\mu(\alpha, i) = i'$;
- there exists a clause $C_{\alpha}$ such that $\sigma_F^{(c)}(\alpha, i) = C$ for all $i \in D_{\alpha}$.

Furthermore, given a finite set $T \subset \mathbb{Z}_+$, we define

$$\mu(\alpha, T) = \mu(\alpha, \arg \min_{i \in T} h(\alpha, i)).$$
and then clearly \( \mu(\alpha, T) \in D_\alpha \) and has the property that
\[
h(\alpha, \mu(\alpha, T)) \leq h(\alpha, i), \text{ for all } i \in T.
\]

We then define \( \sigma_F^{(w)}(\alpha) = C_\alpha \), and \( Y \) and \( Y' \) the random variables that count the number of iterations using \( \sigma_F^{(c)} \) and \( \sigma_F^{(w)} \), respectively. Just as in the previous two lemmas, we will define some helpful auxiliary selection rules. These selection rules will be with memory, but their structure is quite simple, so we will describe them in words first. The selection rule \( \sigma_F^{(i)} \) behaves just like \( \sigma_F^{(w)} \) in the first \( i \) steps. At step \( i \), it “shifts” its clock by some amount, depending on the current assignment at that step, and afterwards behaves like \( \sigma_F^{(c)} \) with the shifted clock. It is clear that this can be modelled as a selection rule with memory, but in order to simplify the notation, we will use the “clock” notation with an additional parameter, which specifies the assignment encountered at step \( i \) (the “turnpoint”). In case step \( i \) has not been reached yet, this parameter will be undefined. Here is the definition:

\[
\sigma_F^{(i)}(\alpha, j, \alpha_{\text{turn}}) = \begin{cases} 
\sigma_F^{(w)}(\alpha) & \text{if } j \leq i, \\
\sigma_F^{(c)}(\alpha, j - i + \delta(\alpha_{\text{turn}}, i)) & \text{if } j > i,
\end{cases}
\]

where \( \delta(\alpha_{\text{turn}}, i) \) determines the corresponding clock shift, and is defined recursively in the following way:

\[
\delta(\alpha_{\text{turn}}, i) = \mu(\alpha, \{ \delta(\beta, i - 1) + 1 : \beta \text{ is not satisfying} \})
\]

Clearly \( \delta(\alpha_{\text{turn}}, i) \) fulfills the following properties:

- \( \delta(\alpha_{\text{turn}}, i) \in D_{\alpha_{\text{turn}}}; \)
- for all non-satisfying assignments \( \beta \),

\[
h(\alpha_{\text{turn}}, \delta(\alpha_{\text{turn}}, i)) \leq h(\alpha_{\text{turn}}, \delta(\beta, i - 1) + 1).
\]

Let \( Y^{(i)} \) be the random variable that counts the number of iterations performed using \( \sigma_F^{(i)} \), and \( \alpha^{(i)} \) be the current assignment. We have the following identity, for \( j \leq i \leq 0 \):

\[
E \left[ Y^{(i)} \mid Y^{(i)} > j, \alpha^{(i)} = \alpha \right] = h(\alpha, \delta(\alpha, i) + j - i) + i.
\]

(4.8)
As in the previous proofs, we will show that $\mathbb{E}\left[Y^{(i)} \mid Y^{(i)} > i, a_i^{(i)} = \alpha \right] \geq \mathbb{E}\left[Y^{(i+1)} \mid Y^{(i+1)} > i, a_i^{(i+1)} = \alpha \right]$. This is achieved by the following sequence of relations (where we apply Identity 4.8):

\[
\begin{align*}
\mathbb{E}\left[Y^{(i)} \mid Y^{(i)} > i, a_i^{(i)} = \alpha \right] &= \\
&= \sum_{\beta \in \{0,1\}^{\text{var}(F)}} \Pr\left[a_{i+1}^{(i)} = \beta \mid a_i^{(i)} = \alpha \right] \mathbb{E}\left[Y^{(i)} \mid Y^{(i)} > i, a_i^{(i)} = \alpha, a_{i+1}^{(i)} = \beta \right] \\
&= \sum_{\beta \in \{0,1\}^{\text{var}(F)}} \Pr\left[a_{i+1}^{(i)} = \beta \mid a_i^{(i)} = \alpha \right] h(\beta, \delta(\alpha, i) + 1) + i + 1 \\
&\geq \sum_{\beta \in \{0,1\}^{\text{var}(F)}} \Pr\left[a_{i+1}^{(i+1)} = \beta \mid a_i^{(i+1)} = \alpha \right] h(\beta, \delta(\beta, i + 1)) + i + 1 \\
&\geq \sum_{\beta \in \{0,1\}^{\text{var}(F)}} \Pr\left[a_{i+1}^{(i+1)} = \beta \mid a_i^{(i+1)} = \alpha \right] \mathbb{E}\left[Y^{(i+1)} \mid Y^{(i+1)} > i, a_i^{(i+1)} = \beta \right] \\
&= \mathbb{E}\left[Y^{(i+1)} \mid Y^{(i+1)} > i, a_i^{(i+1)} = \alpha \right].
\end{align*}
\]

The rest of this proof is analogous to that of Proposition 4.11. □

4.4 Angels For Some Restricted Classes of Formulas

The purpose of this section is to find angels for some formulas for which we have reasons to think that they are “hard” for Schöning’s algorithm. We use two classes of formulas: the first consists of the formulas used as a worst-case example for Schöning’s algorithm using a random selection rule; the second consists of formulas used in a previous diploma thesis as hard examples. We exhibit linear angels of type 2 for both classes.

The Formulas Used For the Worst-case Scenario

We will now find an angel of type 2 without memory for the set of all formulas $F_{n,k}^{\text{rand}}$, first considered in Section 3.3, where $k$ will be fixed from the start.

This will also imply having an angel for the formulas $F_{n,k}^{\text{dev}}$, since each formula $F_{n,k}^{\text{rand}}$ contains a subset of the clauses of $F_{n,k}^{\text{dev}}$, and they have the same satisfying assignment.

To remind the reader, the $k$-CNF formula $F_{n,k}^{\text{rand}}$ consists of all possible type 1 clauses (the type is the number of positive literal), and all possible clauses
of type at least 2 containing the only the first $k$ variables (we assume that the variables are ordered), which we will now call *pivoting variables*. Also, the clauses that contain just these variables will be *pivoting clauses*.

The angel will act in the following way:

- If in the current assignment at least one of the pivoting variables is 0, then exactly one pivoting clause is violated, and the angel chooses that clause.
- Else, assuming that there is a 0 in the current assignment (otherwise we are done), the angel will pick the type 1 clause consisting of the first $k - 1$ pivoting variables as negative literals, and as the positive literal, a variable which is mapped to 0.

The basic idea is that once a non-pivoting variable is set to 1, it will never change again. Also, the algorithm keeps picking a pivoting clause until all the pivoting variables are set to 1, which we will call a *pivoting loop*. When the pivot loop ends, either all variables are 1, or the algorithm will try next to flip one non-pivoting variable from 0 to 1. This may succeed (with probability $1/k$), or not. If it does, there are less non-pivoting variables mapped to 0. If it does not, then the algorithm will go into a pivoting loop, and then try again.

The key lies in the observation that the number $U$ of non-pivoting variables that are 0 can only decrease. Clearly the durations of the pivoting loops are independent of one another, and we will show that the expected duration is upper-bounded by a constant dependent on $k$ (for $k = 3$, this constant is 10). From this, we can deduce a bound on the expected time needed to decrease $U$ by 1. This gives a bound for the overall runtime, since $U$ can decrease at most $n - k$ times.

We will first analyze a pivoting loop. Let $j$ be the number of pivoting variables that are currently set to 0. Then at each time step, $j$ can increase or decrease by 1, and the pivoting loop exits when $j = 0$. The transition probabilities are determined by the chance to flip either a 1 or a 0 in the chosen violated clause. Clearly, the clause is of type $j$, and thus the probability to choose a 0 is $j/k$, which will also be the probability that $j$ will decrease by 1. Also, if $j = k$, it is clear that $j$ can only decrease. Let $X^{(k)}$ be the random variable that counts the number of 0’s among the pivoting variables, and let $Y^{(k)}$ denote the duration of the pivoting loop.

We want to obtain an upper bound for the expectation of $Y^{(k)}$. This amounts to using Lemma 4.6 on the subformula of $F^{\text{rand}}_{n,k}$ consisting of the clauses containing the first $k$ variables ($2^k - 1$ clauses in total). The pivoting loop
ends when a satisfying assignment is found for the subformula. We thus obtain

$$E[Y^{(k)}] \leq k \cdot 3^k,$$

which is already enough for what we need.

We can however compute the exact amount value for $E[Y^{(k)}]$ by looking at the proof of the following proposition. Also we get the following bound for $E[Y^{(k)}]$, which is slightly better than what we had before.

**Proposition 4.14** We have that $E[Y^{(k)}] \leq 2e^k - 1$.

**Proof** Proceeding in a similar manner as for the previous proof, we first observe that $E[Y^{(k)} | X^{(k)}_0 = j] = E[Y^{(k)} | X^{(k)}_1 = j] + 1$ by the Markov property. Then we have for $0 < j \leq k$,

$$g^{(k)}_j = E[Y^{(k)} | X^{(k)}_0 = j] =$$

$$= \Pr[X^{(k)}_1 = j - 1 | X^{(k)}_0 = j] \cdot E[Y^{(k)} | X^{(k)}_1 = j - 1] +$$

$$+ \Pr[X^{(k)}_1 = j + 1 | X^{(k)}_0 = j] \cdot E[Y^{(k)} | X^{(k)}_1 = j + 1]$$

$$= \frac{j}{k} g^{(k)}_{j-1} + \left(1 - \frac{j}{k}\right) g^{(k)}_{j+1} + 1. \quad (4.9)$$

Clearly we have $g^{(k)}_0 = 0$, and $g^{(k)}_k = g^{(k)}_{k-1} + 1$. From this and the equation above, we can determine the value of $g^{(k)}_k$. To make this task simple, let $a^{(k)}_j = g^{(k)}_j - g^{(k)}_j$. Then we will have $a^{(k)}_0 = 0, a^{(k)}_{k-1} = 1$, and, derived from Equation 4.9,

$$a^{(k)}_{j-1} = \frac{k}{j} (a^{(k)}_j + 1) - \left(\frac{k}{j} - 1\right) a^{(k)}_{j+1}. \quad (4.10)$$

This recurrence provides the means to compute $g^{(k)}_k$, and the exact values for some small $k$ are provided in the table below.

<table>
<thead>
<tr>
<th>$k$</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>$g^{(k)}_k$</td>
<td>10.00</td>
<td>21.33</td>
<td>42.67</td>
<td>83.20</td>
<td>161.07</td>
<td>312.08</td>
<td>607.09</td>
<td>1186.54</td>
</tr>
<tr>
<td>$2e^{k-1}$</td>
<td>14.8</td>
<td>40.2</td>
<td>109.2</td>
<td>296.8</td>
<td>806.9</td>
<td>2193.3</td>
<td>5961.9</td>
<td>16206.2</td>
</tr>
</tbody>
</table>
We want now to prove that $g^{(k)}_k \leq 2e^{k-1}$. For $j = k - 1$, Relation 4.10 is $a^{(k)}_{k-2} = \frac{2k}{k+1}$, and for $j < k - 1$, we approximate it with $a^{(k)}_j \leq \frac{k}{j} \cdot a^{(k)}_j$. Then we obtain the bound

$$g^{(k)}_k = a^{(k)}_0 \leq 2 \frac{k^{k-1}}{(k-1)!} \leq 2e^{k-1}.$$ 

We now turn back to the problem of bounding the expected time needed to decrease by 1 the number of non-pivoting 0’s. Let $N^{(k)}$ and $Z^{(k)}$ be the random variables representing the number of loops and the number of individual steps, respectively, needed until a decrease in $U$ is obtained. Clearly $N^{(k)}$ has a geometric distribution, with $p = 1/k$. Also, if we consider a fixed value of $N^{(k)}$, we have $E[Z^{(k)} | N^{(k)} = \nu] \leq \nu g^{(k)}_k$. We obtain

$$E[Z^{(k)}] = \sum_{\nu \geq 1} \Pr[N^{(k)} = \nu] \cdot E[Z^{(k)} | N^{(k)} = \nu] \leq g^{(k)}_k \sum_{\nu \geq 1} \Pr[N^{(k)} = \nu] \nu \leq g^{(k)}_k E[N^{(k)}] = k g^{(k)}_k.$$ (4.11)

Since the number of possible decreases of $U$ is at most $n - k$, and for each decrease we need to wait in expectation at most $kg^{(k)}_k$ steps. Then clearly the overall number of steps needed for the algorithm find a satisfying assignment is in expectation at most $(n - k)kg^{(k)}_k$, which is a polynomial in $n$. Thus, the selection rule which we have described is an angel of type 2.

**The Circular-XOR Formula**

We now consider a second type of formulas for which we will devise an angel of type 2. These formulas have been defined in Matthias Kaufmann’s diploma thesis[6], where they were conjectured to be a hard example for Schöning’s algorithm. Up to isomorphism, they have the following form:

$$F^{\text{circ}}_n = \bigwedge_{i=1}^n (x_i \oplus x_{i+1} \oplus x_{i+2}),$$

where the set of variables is $\{x_1, \ldots, x_n\}$, indices are taken modulo $n$, and “$\oplus$” represents the logical operation “XOR”. In this section, we will prove that there exists a linear angel of type 2 (one that finds a satisfying assignment in linear time in expectation).
Proposition 4.15 The formula $F_n^{\circ}$ has exactly one satisfying assignment if and only if $n$ is not divisible by 3.

Proof Clearly the all-one assignment is always satisfying. If $n \equiv 0 \pmod{3}$, then the assignment defined by $a[x_{3j}] = 1$, $a[x_{3j+1}] = a[x_{3j+2}] = 0$ is satisfying. Otherwise, let $a$ be a satisfying assignment such that for some $x_i$, $a[x_i] = 0$. Since $a[x_i] \oplus a[x_{i+1}] \oplus a[x_{i+2}] = 1$, we must have $a[x_{i+1}] \oplus a[x_{i+2}] = 1$, and $a[x_{i+3}] = 0$. We can now deduce that $a[x_{i+3j}] = 0$, for all $j \in \mathbb{Z}$. If $n \neq 0 \pmod{3}$, then the assignment $a$ is all-zero, which is clearly not satisfying, contradiction! \square

In the rest of this section, we will assume that $n$ is not divisible by three, so there will be only a single satisfying assignment. We transform the formula logical formula above easily into the 3-CNF formula below:

$$F_n^{\circ} = \bigcup_{i=1}^{n} \{ \{x_i, x_{i+1}, x_{i+2}\}, \{\overline{x}_i, \overline{x}_{i+1}, \overline{x}_{i+2}\}, \{x_i, x_{i+1}, x_{i+2}\}, \{\overline{x}_i, x_{i+1}, x_{i+2}\} \}.$$

In other words, $F_n^{\circ}$ contains all clauses of type 1 and 3 containing variables with adjacent indices (which wrap around modulo $n$).

We will now construct a type 2 angel with memory. The core idea that we use to construct the angel is the following: given the current assignment, we look at it as at a circular string of bits, and consider the longest continuous substring of 1’s. The goal is to “guide” Schöning’s algorithm in such a way that this sequence gets larger and larger. Suppose the length of this sequence is $L$, and that $3 \leq L \leq n - 3$. For the other cases we will have special provisions. We consider a window of size 6 such that it contains exactly the 3 rightmost 1’s in the sequence. The window will then necessarily have the form “1110∗∗”. We then repeat the following for at most 20 steps (this will be called window processing): depending on the current window string, we choose one particular violated clause, or we stop. if there is no violated clause in the window (we consider only clauses that lie completely within the window), then we stop. The exact rules that we use to determine the violated clause will be given below. At the end of this process, the longest sequence of 1’s will be changed, but the expected change will always be positive, or, equivalently, in expectation, the longest sequence $L$ will increase.

After processing one window, we want to have a current assignment that does not violate any clauses of type 3. Therefore, the angel should pick any clauses of type 3 (corresponding to substrings of the form “000”) that may have been created by the window process. There will clearly be at most 8 picks needed, since solving one clause of type 3 does not create new violations of other clauses of the same type.
The window processing is repeated as long as possible (i.e., as long as \(3 \leq L \leq n\)). If \(L < 3\), we use a different tactic. Clearly there must be a violated clause of type 1, since we have assured that no clauses of type 3 are violated. We pick that clause, and with 1/3 probability, we will obtain a sequence of 3 ones in the current assignment. Otherwise, we clean again any clauses of type 3 that might have appeared and try again.

If \(L = n - 2\), then there are clearly only two violated clauses, both of type 1. We pick one of them, and with 1/3 probability we will obtain \(L = n - 1\). If \(L = n - 1\), then there is only one violated clause, of type 1, and picking it will lead with probability 1/3 to satisfaction. In conclusion, every time \(L \geq n - 2\), there is chance of at least 1/9 to jump to satisfiability in the next 2 steps.

We will now describe the rules needed for window processing. Given a particular window string, the rules are the following:

- if the sequence 000 appears anywhere, we consider the leftmost occurrence;
- if not, then if sequence 101 appears anywhere, we consider the leftmost occurrence;
- if not, if the string starts with at least 4 ones, the processing stops;
- if not, if the sequence 011 appears anywhere, we consider the rightmost occurrence;
- if not, if the sequence 110 appears anywhere, we consider the rightmost occurrence;
- if not, clearly there are no violated clauses within the window, and processing stops.

With the aid of a program, we compute the probability that at the end of the window process, the number of 1’s at the left of the window changes with \(\delta\), where \(\delta\) can be between \(-3\) and 3. The program maintains a distribution over the window strings, and changes it at each step. The results are given in Table 4.2, for all possible starting window strings.

We briefly describe the operation of the program. It takes the starting window string \(s\) as input, and creates the initial distribution \(w^{(0)}\) by assigning probability 1 to the starting string and 0 to the rest. We consider that if the window process stops before the 20-th turn, it keeps its final state for the rest of the turns. At each step, the probability distribution is updated in the
4. Selection Rules of Type “Angel”

Table 4.1: All possible window strings. If they force the processing to stop, a box is drawn, otherwise the selected violated clause is underlined.

\[
w^{(i+1)}(s) = \sum_{s' \text{ successor of } s} \frac{1}{3}w^{(i)}(s') \quad \text{if } s \text{ is not stopping;}
\]

\[
w^{(i+1)}(s) = \sum_{s' \text{ successor of } s} \frac{1}{3}w^{(i)}(s') + w^{(i)}(s) \quad \text{if } s \text{ is stopping.}
\]

Clearly \(w^{(i)}\) represents probability distribution of the current window string after \(i\) turns. After 20 turns, we add the probabilities of strings corresponding to the same \(\delta\), and output the results.

<table>
<thead>
<tr>
<th>start</th>
<th>(\delta = -3)</th>
<th>(\delta = -2)</th>
<th>(\delta = -1)</th>
<th>(\delta = 0)</th>
<th>(\delta = 1)</th>
<th>(\delta = 2)</th>
<th>(\delta = 3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>111001</td>
<td>0.3105</td>
<td>0.1091</td>
<td>0.0665</td>
<td>0.0077</td>
<td>0.2396</td>
<td>0.0281</td>
<td>0.2986</td>
</tr>
<tr>
<td>111010</td>
<td>0.2419</td>
<td>0.0513</td>
<td>0.0019</td>
<td>0.0027</td>
<td>0.1873</td>
<td>0.4710</td>
<td>0.0440</td>
</tr>
<tr>
<td>111011</td>
<td>0.2154</td>
<td>0.0825</td>
<td>0.0087</td>
<td>0.0009</td>
<td>0.1175</td>
<td>0.0485</td>
<td>0.5265</td>
</tr>
</tbody>
</table>

Table 4.2: The probabilities corresponding to each change \(\delta\), and the expectation of \(\delta\), corresponding to the starting window string.

70
We now return to the problem of bounding the expectation of the number of window processes required. Also, we regard the case where \( L < 3 \) as a special type of window process, where only values \( \delta = 1 \) and \( \delta = 0 \) are possible, the first one coming with probability 1/3, exactly when a zero is flipped into a one. Then the expectation of \( \delta \) in this case is 1/3. Notice that so far, the least expectation of \( \delta \) that we have encountered was given by the process starting with 111001, in which case \( \mathbb{E}[\delta] = 0.035 \).

**Notation.** We set \( \mu = 0.035 \).

Since the processing of a window (including the elimination of clauses of type 3) is done in constant time, instead of considering individual time steps in the analysis, we will only look at the times when each window process starts. Let \( D_t \) be the random variable representing the value \( \delta \) resulted from the \( t \)-th window process. If \( L_t \) is the longest sequence of ones in the current assignment after the \( t \)-th window process, then we see that if \( L < n - 2 \) every time, then also \( \sum_{i=1}^{t} D_i \leq L_t - L_0 \). This happens because all \( D_i \) always underestimate the change in the length of the longest subsequence of ones.

Intuitively, since \( D_t \) will increase in expectation whenever \( L < n - 2 \), we expect that the number of turns needed until we reach \( L \geq n - 2 \) is linear. Let \( Y \) be the random variable that counts how many turns (read: window processes) are needed until \( L \geq n - 2 \). We want to find a bound for \( \mathbb{E}[Y] \), and we use the following definition of expectation:

\[
\mathbb{E}[Y] = \sum_{t=0}^{\infty} t \cdot \mathbb{P}[Y = t] = \sum_{t=0}^{\infty} \mathbb{P}[Y \geq t]. \tag{4.12}
\]

The idea is to show that for \( t \) greater than some threshold, the terms \( \mathbb{P}[Y \geq t] \) contribute negligibly to the expectation above. We set the threshold to \( 2n/\mu \), and we grossly approximate the terms with \( t < 2n/\mu \) by \( \mathbb{P}[Y \geq t] \leq 1 \), obtaining

\[
\mathbb{E}[Y] \leq \frac{2n}{\mu} + \sum_{t=2n/\mu}^{\infty} \mathbb{P}[Y \geq t]. \tag{4.13}
\]

We notice that the event \( Y > t \), for the values of \( t \) considered above, implies automatically \( L_t < n - 2 \), which in turn implies \( \sum_{i=1}^{t} D_i < n - 2 \). Now we will use Azuma’s Inequality (see Appendix A.2) to estimate the probability that this event happens. We cannot yet apply the theorem, since \( D_i \) does not fulfill the condition of zero expectation conditioned on previous values. We construct another set of random variables \( \{C_i\} \), such that it does. We simply define them as

\[
C_i = D_i - \mathbb{E}[D_i | D_{i-1}, \ldots, D_1].
\]
This corresponds to shifting $D_i$ with the expectation of $\delta$ at each step, removing the bias. It can be easily showed that

$$E[C_i | C_{i-1}, \ldots, C_1] = E[C_i | D_{i-1}, \ldots, D_1] = 0.$$  

Also, clearly $E[D_i | D_{i-1}, \ldots, D_1] \geq \mu$, and thus we have $D_i \geq C_i + \mu$. We obtain (remember that $t \geq 2n/\mu$)

$$\Pr[Y > t] \leq \Pr \left[ \sum_{i=1}^{t} D_i < n - 2 \right] \leq \Pr \left[ \sum_{i=1}^{t} (C_i + \mu) < n - 2 \right]$$

$$\leq \Pr \left[ \sum_{i=1}^{t} C_i < n - t\mu \right] \leq \Pr \left[ \sum_{i=1}^{t} C_i < -\frac{t\mu}{2} \right] \leq e^{-\frac{t^2\mu^2}{8t\varepsilon^2}} = e^{-t^2/8\varepsilon},$$

where we have applied Azuma’s Inequality, knowing that $|C_i| < c$ (for example $c = 4$ suffices).

Clearly the sum $\sum_{i=2n/\mu}^{\infty} \Pr[Y \geq t]$ converges and actually decreases with $n$. We have so far obtained $E[Y] \leq \frac{2n}{\mu} + O(1)$. However, this is not the expected runtime of the algorithm, for two reasons:

- Since we actually counted window processes, not iterations, we need to multiply our result with a constant factor (at most, say, 30, in this case).

- Also, because $Y$ measures the time needed to arrive at an assignment with $L \leq n - 2$, and not necessarily the satisfying assignment. However, from that point, we always get a chance of at least 1/9 to jump directly to the satisfying assignment. Following the same reasoning as in the previous subsection (see Relation 4.11), we arrive at the conclusion that we need to multiply the expectation by an additional factor of at most 9.

Putting everything together, we obtain an upper bound for the expected number of iterations in Schöning’s algorithm given by the polynomial

$$f(n) = 2n \cdot \frac{1}{0.035^2} \cdot 30 \cdot 9 + O(1) \approx 17000n + O(1),$$

but which is, nevertheless, linear.
Chapter 5

Conclusions and Open Problems

In the introductory chapter, we have introduced Schöning’s algorithm and more exact version of the analysis due to Welzl [15]. We have also detailed the coupling arguments that are needed in order to reduce the operation of the algorithm to a random walk on the integers.

In the second chapter we have applied an adapted version of the analysis of Schöning’s algorithm to a modified algorithm that very closely resembles the one used for derandomizing the Lovász Local Lemma [7]. To achieve this, we have introduced a combinatorial formalism that allows us to generalize the results from the original analysis. We showed that when applied to general $k$-CNF formulas, the modified algorithm performs worse than Schöning’s algorithm. In particular, in the case of 3-CNF formulas, it has a runtime of $O(1.618^n)$, compared to $O(1.334^n)$, which corresponds to the original algorithm. However, it is not known yet how this algorithm behaves on certain classes of formulas, in particular for those which “almost” fulfill the conditions of Lovász Local Lemma, so future research might be concentrated in this direction.

The third chapter shows that the analyses we have presented in the first two chapters are tight. To achieve this, we have shown that for a very pessimistic selection rule (the devil) and for a random rule there exist formulas that make both Schöning’s algorithm and the modified version run as slow as the previously determined upper bound, up to polynomial factors.

In the fourth chapter, we consider the possibility of a benevolent selection rule (the angel). We propose two ways in which to define such rules (of a stronger type, type 2, and a weaker type, type 1), and differentiate them according to whether they use previous knowledge about the assignments encountered so far. We also prove various relations between these types of angel, in particular we show that for angels of type 2 it does not matter
whether they use such previous knowledge or not. We think that there is a strong possibility that there exists a universal angel (cf. Section 4.2). One way to determine this may be to exhibit angels for more classes of formulas and then attempt a generalization. Alternatively, one may try to show that there exist no angels for a particular class of formulas, which would imply the non-existence of a universal angel.
Appendix A

Probability Theory

A.1 Chernoff Bounds

**Theorem A.1** Let $0 < p < 1/2$, and $X_1, \ldots, X_n$ be $n$ independent Bernoulli random variables, such that $\Pr[X_i = 1] = p$ for $1 \leq i \leq n$. Then, setting $Y = \sum_{i=1}^{n} X_i$, we have

$$\Pr[Y > n/2] < e^{-2n(p - \frac{1}{2})^2}. \quad (A.1)$$

A second form approximates the probability that the sum of the Bernoulli variables deviates with some factor from the expectation.

**Theorem A.2** Let $0 < p < 1/2$, $\delta > 0$, and $X_1, \ldots, X_n$ be $n$ independent Bernoulli random variables, such that $\Pr[X_i = 1] = p$ for $1 \leq i \leq n$. Then, setting $Y = \sum_{i=1}^{n} X_i$, we have

$$\Pr[Y > (\delta + 1)E[Y]] < \left(\frac{e^{\delta}}{(1 + \delta)^{1+\delta}}\right)^{E[Y]}. \quad (A.2)$$

In the proof of Lemma 3.29, we use the form

$$\Pr[Y > (\delta + 1)E[Y]] < e^{-E[Y]} \left(\frac{e^{\delta}}{1 + \delta}\right)^{(1+\delta)E[Y]}. \quad (A.3)$$
A.2 Azuma’s Inequality

Theorem A.3 (Azuma) If \( \{X_i\}_{i\geq 0} \) are real-valued random variables satisfying the following conditions,

\[
\begin{align*}
E[|X_i|] &< \infty, \\
E[X_{i+1} | X_i, \ldots, X_1] & = X_i, \text{ and} \\
|X_i - X_{i-1}| &< c_i, \text{ for all } i \leq 1,
\end{align*}
\]

then for all positive integers \( j \) and reals \( t > 0 \), we have

\[
Pr[X_j - X_0 \geq t] \leq \exp\left(-\frac{t^2}{2\sum_{i=1}^{j} c_i^2}\right).
\]

Setting \( C_i = X_i - X_{i-1} \), and \( c_i = c \), we obtain the following variant.

Corollary A.4 If we consider random variables \( \{C_i\}_{i\geq 1} \) such that for all \( i \geq 0 \),

\[
\begin{align*}
E[C_i | C_{i-1}, \ldots, C_1] & = 0 \text{ and} \\
|C_i| &< c,
\end{align*}
\]

then for all positive integers \( i \) and reals \( t > 0 \), we have

\[
Pr\left[\sum_{i=1}^{j} C_i \geq t\right] \leq e^{-\frac{t^2}{2ic}}.
\]

By using \(-C_i\) instead of \( C_i \), we obtain

\[
Pr\left[\sum_{i=1}^{j} C_i \leq -t\right] \leq e^{-\frac{t^2}{2ic}}.
\]
Bibliography


