ETH zürich

ETH Library

Random walk algorithms for SAT and constraint satisfaction problems

Master Thesis

Author(s):
Schneider, Stefan

Publication date:
2010

Permanent link:
https://doi.org/10.3929/ethz-a-006154476

Rights / license:
In Copyright - Non-Commercial Use Permitted

This page was generated automatically upon download from the ETH Zurich Research Collection.
For more information, please consult the Terms of use.

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Random Walk Algorithms for SAT and Constraint Satisfaction Problems

Master Thesis

Stefan Schneider

August, 2010

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

Abstract

Schöning [25] presents a simple yet elegant randomized algorithm for (d,k)-CSP problems with a running time of $(d(k-1)/k)^n \operatorname{poly}(n)$. Here, d is the number of colours, k is the size of the constraints and n is the number of variables. In the context of the derandomization of Schöning's algorithm by Dantsin et al. [3], Scheder [24] gave an improvement by defining a graph structure on the set of d colours, the *search graph*. In this thesis we consider the same modification for the randomized version of Schöning's algorithm and show that it does not give a better running time. More precisely, we show that for any search graph satisfying the symmetry property of distance-regularity, the running time is $(d(k-1)/k)^n \operatorname{poly}(n)$, the same running time as the original algorithm.

Contents

Chapter 1

Introduction

1.1 Motivation and Outline

The constraint satisfaction problem (CSP) is an important problem in theoretical computer science. It is a generalisation of the problem of boolean satisfiability (SAT), which is one of the most famous examples of NP-completeness. Many problems in computer science can be reduced in a natural way to CSP or SAT in particular, for instance the eight queens puzzle, graph colouring and many scheduling problems. Due to both its theoretical and practical importance there has been a lot of research on faster CSP and SAT algorithms in the last few decades. While most of the research concentrates on the special case of SAT, many of the algorithms and their analysis can be nicely generalised for CSP.

A major improvement in the field of constraint satisfaction problems was made by Schöning [25] who proposed a simple randomized algorithm that finds a satisfying assignment for a 3-CNF formula in time $1.334^n \cdot \mathrm{poly}(n)$ and in time $(d(k-1)/k)^n \cdot \mathrm{poly}(n)$ for a general (d,k)-CNF. It chooses a random assignment at the beginning and then explores the set of assignments by randomly choosing a variable of an unsatisfied clause and assigning a new value to this variable, again randomly. If no satisfying assignment is found after a polynomial number of steps a new random assignment is resampled and the procedure is repeated.

A derandomization of Schöning's algorithm by Dantsin, Goerdt, Hirsch, Kannan, Kleinberg, Papadimitriou, Raghavan and Schöning [3] achieves a running time of $(dk/(k+1))^n \cdot \mathrm{poly}(n)$, inferior to the randomized version. The algorithm considers a deterministic set of initial assignments based on covering codes. Whenever Schöning's algorithm would choose a random variable and a random new value, it would instead consider all possible

choices in a search tree. Scheder [24] improved upon this algorithm and achieved a running time of $\left(d(k-1)/k \cdot k^d/(k^d-1)\right)^n \cdot \text{poly}(n)$ which comes closer to the running time of Schöning's algorithm, eliminating the gap almost completely for large d. Scheder's algorithm arranges the d possible values of a variable on a graph and instead of considering all new values for a variable it only considers new values that are adjacent to the current value in the graph. We refer to this graph as the *search graph*. Scheder shows in particular that the directed cycle as search graph is optimal.

In this thesis we consider Scheder's modification for the randomized version of Schöning's algorithm. Instead of randomly choosing a new value among all possible values, only consider values that are adjacent to the current value in the search graph. Choosing a complete graph gives us Schöning's algorithm, hence the analysis in this thesis contains Schöning's algorithm as a special case. We show that for any search graph satisfying the symmetry property of distance-regularity the running time of the modified algorithm is $(d(k-1)/k)^n \cdot \text{poly}(n)$, hence the same as Schöning's algorithm. Distance-Regularity is a weaker property than vertex-transitivity. Hence for a large number of search graphs, the running time is the same.

The main part of this thesis consists of a proof for an upper bound on the running time for general distance-regular graphs. We also give a proof that the obtained bound is tight. More specifically, we show that there is no selection rule such that a better upper bound can be reached, independent of the search graph. While the proof relies heavily on a tree branching process with some Markovian properties, we give an alternative proof for a special case using Markov chains in the appendix.

1.2 Notation and Terminology

This thesis will follow the notation used in the lecture notes of Prof. Emo Welzl at ETH Zürich for the course "Boolean Satisfiability - Combinatorics and Algorithms" [29].

Based on the universality of the conjunctive normal form (CNF) we will assume that all constraint satisfaction problems are given in CNF. In order to have a simplified representation of these formulas we introduce *set notation*.

In a *constraint satisfaction problem* (CSP) we are given a finite set of variables, each of which has a finite set of possible colours. Moreover, there is a set of constraints, each one a set of variable-colour pairs. If V denotes the set of variables and L_x denotes the list of possible colours of $x \in V$, then a

constraint or *clause* is a set

$$C = \{(x_1, c_1), (x_2, c_2), \ldots, (x_k, c_k)\}, k \in \mathbb{N}$$

with $x_j \in V$ pairwise distinct and $c_j \in L_{x_j}$ for $j \in \{1, \ldots, k\}$. A pair (x, c) is called a *literal* and a *formula* is a set of clauses. We use the notation $\mathrm{vbl}(C)$ for the set of variables in C and $\mathrm{vbl}(F) = V = \bigcup_{C \in F} \mathrm{vbl}(C)$ for the set of variables in F.

An *assignment* α assigns colours to variables from their respective lists of possible colours ($\alpha(x) \in L_x$ for all $x \in V$). An assignment *satisfies a clause* C if there is a literal $(x, c) \in C$ with $\alpha(x) \neq c$. We say α *violates* C if $\alpha(x) = c$ for all $(x, c) \in C$. We call such a clause a violated or unsatisfied clause. An assignment satisfies a formula F if it satisfies all clauses $C \in F$.

For a formula F and an assignment α, we denote the set of clauses in F that are violated by α as $\mathrm{vlt}(F, \alpha)$. By this notation an assignment α satisfies F if and only if $\mathrm{vlt}(F, \alpha) = \emptyset$.

The constraint satisfaction problem then is the decision problem if a given formula F has a satisfying assignment.

A CNF formula F represented in the logical notation

$$F = \bigwedge_{i=1}^{m} \left(\bigvee_{j=1}^{k_i} x_{i,j} \neq c_{i,j} \right)$$

with $x_{i,j} \in V$ and $c_{i,j} \in L_{x_{i,j}}$ is written in set notation as

$$F = \{C_1, \ldots, C_m\} \text{ where } C_i = \{(u_{i,1}, c_{i,1}), \ldots, (u_{i,k_i}, c_{i,k_i})\}$$

Throughout this thesis we will assume $|C| = k$ for all $C \in F$ and $L_x = \{1, \ldots, d\}$ for all $x \in \mathrm{vbl}(F)$. We call such a formula a (d, k)-CNF and the corresponding decision problem (d, k)-CSP. It can easily be seen that $(2, k)$-CSP and k-SAT are the same problem, hence CSP is a generalisation of SAT.

In various places we will use the notion of the Hamming distance between two assignments α and β. The Hamming distance $d_H(\alpha, \beta)$ is defined as the number of variables x, such that $\alpha(x) \neq \beta(x)$, i.e.

$$d_H(\alpha, \beta) = \{x \in V \mid \alpha(x) \neq \beta(x)\}$$

1.3 Markov Chains

In several parts of this thesis we will use the concept of Markov chains to analyse algorithms. We can think of a Markov chain as a variable that changes in discrete time steps. The new value of the variable only depends on its current value and is independent of older values. Hence a Markov chain can be described as memory-less.

Definition 1.1 *For some state space S, let $\{X_i\}_{i\geq 0}$ be a sequence of random variables with $X_i \in S$ for all i. The sequence $\{X_i\}_{i\geq 0}$ is a Markov chain if the following property, which we call the Markov property, holds for all $i \geq 0$ and all $x_1, \ldots, x_{i+1} \in S$:*

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

We will call $\Pr[X_{i+1} = x_{i+1} \mid X_i = x_i]$ the transition probabilities and the distribution of X_0 the starting distribution.

We are often dealing with Markov chains such that the transition probabilities are independent of i. We call Markov chains with that property time-homogeneous.

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

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

For time-homogeneous Markov chains we use the following short notation for transition probabilities.

Definition 1.3 *Let $\{X_i\}_{i\geq 0}$ be a time-homogeneous Markov chain. Then for $a, b \in S$ we define*

$$p_{\{X_i\}_{i\geq 0}}(a, b) \equiv \Pr[X_{i+1} = b \mid X_i = a]$$

In unambiguous context we will use $p(a, b)$ for the same.

1.4 Schöning's Algorithm

We now present the randomized algorithm introduced by Schöning [25]. Given a satisfiable (d, k)-CNF F, it finds a satisfying assignment in expected time $(d(k-1)/k)^n \operatorname{poly}(n)$, where $n = |\mathrm{vbl}(F)|$. We assume $d \geq 2$ and $k \geq 2$ with $d > 2$ or $k > 2$, such that the problem is NP-complete. The algorithm consists of repeating the function Sch(F,$3n$) (Algorithm 1) until a satisfying

Algorithm 1 Sch(F,t)

$\alpha \in_{\text{u.a.r.}} \{1,\ldots,d\}^{\text{vbl}(F)}$
if α satisfies F **then**
  return (true, α)
end if
for $i = 1$ **to** t **do**
  $C \in \text{vlt}(F,\alpha)$ (using any selection rule)
  $x \in_{\text{u.a.r}} \text{vbl}(C)$
  $a_{\text{new}} \in_{\text{u.a.r}} \{1,\ldots,d\} \backslash \alpha(x)$
  $\alpha(x) \leftarrow a_{\text{new}}$
  if α satisfies F **then**
    return (true, α)
  end if
end for
return (false, nil)

assignment is found. This algorithm can also be used to decide with an arbitrary small error probability if a formula F is satisfiable or not.

The function Sch(F,t) picks an assignment at random and repeats the following sequence at most t times: pick an arbitrary unsatisfied clause, take a random variable from that clause, flip the variable to a random new value. As soon as a satisfying assignment is found, the algorithm terminates successfully.

Note that while both the variable to be flipped and its new assignment is chosen randomly, the unsatisfied clause used is chosen arbitrarily. In order to fully describe the algorithm we need to specify how the unsatisfied clause is chosen. We call such a rule a *selection rule*.

The algorithm Sch(F,$3n$) needs to be repeated a large number of times in order to obtain a reasonable probability of success. For the analysis of how many times the algorithm has to be repeated we mostly concentrate on the success probability of a single run of Sch(F,$3n$). Then the expected number of repetitions needed can be obtained easily. We state here the result that will be proven at then end of this chapter.

Theorem 1.4 *In expectation, for a satisfiable (d,k)-CNF F, a satisfying assignment is found after repeating Sch(F,$3n$) at most $3\left(d\frac{k-1}{k}\right)^n$ times. The probability that it does not find a satisfying assignment after $3T\left(d\frac{k-1}{k}\right)^n$ repetitions is at most e^{-T}.*

1.5 Recent Results for SAT and CSP

Schöning's algorithm is very similar to a randomized algorithm by Papadimitriou [18], which solves 2-SAT in expected quadratic time. Hofmeister et al. [13] and Rolf [21] improved upon Schöning's algorithm for the special case of 3-SAT by improving the initial assignment.

Apart from local search based algorithms like Schöning's algorithm and Papadimitriou's algorithm, the other major approaches for SAT are DPLL-based algorithms and PPSZ. The DPLL algorithm by Davis, Putnam, Longemann and Loveland [4, 5] is mostly interesting for implementations of SAT solvers where the focus is on empirical results rather than strict upper bounds on the running time. DPLL-based implementations include Posit [7], SATO [30] and GRASP [17]. There are also local search based implementations such as GSAT [26] and WalkSAT [27]. The PPSZ algorithm by Paturi, Pudlák, Saks and Zane [20] is an improvement of the PPZ algorithm [19] and still the best algorithm for k-SAT with $k > 4$. Both DPLL and PPSZ do not easily generalise to CSP, but some work into this direction has been done [6, 16]. The currently best algorithm for 3-SAT is due to Rolf [22] based on Iwama and Tamaki [14], who combined Schöning's algorithm with PPSZ.

Schöning's algorithm has been derandomized by Dantsin et al. [3]. An improvement for 3-SAT is due to Scheder [23]. As mentioned earlier, Scheder [24] also gave an improvement for CSP by considering search graphs.

1.6 Analysis of Schöning's Algorithm

Since the algorithm consists of independent runs of Sch(F,$3n$), we will first compute the probability of success for a single run. Also, we require F to be satisfiable, as the procedure returns false with probability 1 otherwise.

In order to determine the probability of success, given that F is satisfiable we use a *coupling argument*. Given two randomized algorithms or random processes (e.g. Markov chains) we imagine them running in parallel and both get the same random numbers. The random decisions of the two coupled processes then depend on each other, but behave exactly like the original process when observed individually. In this section we will couple Schöning's algorithm to a Markov chain, which then simplifies the analysis.

Let α^* be a particular satisfying assignment of F. We now consider the algorithm Sch-Coupling (algorithm 2) which keeps an additional value X but is identical to Sch otherwise. The algorithm Sch-Coupling is given

knowledge of the satisfying assignment α^*. The main idea behind the auxiliary algorithm Sch-Coupling is that the value X is always an upper bound for the Hamming distance $d_H(\alpha, \alpha^*)$. Given a clause $C \in F$, we further consider the following definition for a particular variable $x \in \mathrm{vbl}(C)$, such that if C is unsatisfied by an assignment α, then $\alpha(x) \neq \alpha^*(x)$.

Definition 1.5 *Assuming some ordering on vbl(F), let $\xi : F \to vbl(F)$ map a clause $C \in F$ to the smallest variable $x \in vbl(C)$ with $(x, a) \in C$, such that $a \neq \alpha^*(x)$.*

Algorithm 2 Sch-Coupling(F,t,α^*)

$\alpha \in_{\text{u.a.r.}} \{1, \ldots, d\}^{\mathrm{vbl}(F)}$
$X \leftarrow d_H(\alpha, \alpha^*)$
if α satisfies F **then**
  return (true, α)
end if
for $i = 1$ **to** t **do**
  $C \in \mathrm{vlt}(F, \alpha)$ (using any selection rule)
  $x \in_{\text{u.a.r}} \mathrm{vbl}(C)$
  $a_{\text{new}} \in_{\text{u.a.r}} \{1, \ldots, d\} \backslash \alpha(x)$
  $\alpha(x) \leftarrow a_{\text{new}}$
  if $x = \xi(C)$ **then**
    if $a_{\text{new}} = \alpha^*(x)$ **then**
      $X \leftarrow X - 1$
    else
      $X \leftarrow X$
    end if
  else
    $X \leftarrow X + 1$
  end if
  if α satisfies F **then**
    return (true, α)
  end if
end for
return (false, nil)

It can easily be seen that the Sch-Coupling behaves exactly like Sch, as we only keep track of an additional variable, which has no influence on the control flow. In some sense, we can also understand the relationship between the two algorithms as a coupling, such that both algorithms pick the same variable and the same new value in every iteration.

Let X_i and α_i be the values of X and α at iteration i. Furthermore, let $i_{\max}$ be the iteration where the algorithm returns. Then we have the following

lemma.

Lemma 1.6 *For all possible runs of Sch-Coupling, and for all* $0 \leq i \leq i_{max}$ *we have* $d_H(\alpha_i, \alpha^*) \leq X_i$.

Proof We use induction. For $i = 0$ the claim holds since $X_0 = d_H(\alpha_0, \alpha^*)$ by definition. For $0 < i \leq i_{\max}$ we assume that the claim holds for $i-1$. Clearly α_{i-1} is not satisfying and an unsatisfied clause C is picked.

If the variable x is picked such that $x = \xi(C)$ then by the definition of ξ we have $\alpha_{i-1}(x) \neq \alpha^*(x)$. If for the new value a_{new} we have $a_{\text{new}} = \alpha^*(x)$, then we have $d_H(\alpha_i, \alpha^*) = d_H(\alpha_{i-1}, \alpha^*) - 1$ and $X_i = X_{i-1} - 1$, hence the condition $d_H(\alpha_i, \alpha^*) \leq X_i$ is maintained. If we have $a_{\text{new}} \neq \alpha^*(x)$ then $d_H(\alpha_i, \alpha^*) = d_H(\alpha_{i-1}, \alpha^*)$ and $X_i = X_{i-1}$, hence the condition is maintained as well. If $x \neq \xi(C)$ then $X_i = X_{i-1} + 1$. As the Hamming distance can not increase by more than 1 in a single iteration, the claim follows. □

A simple corollary then is that the probability that $X_i = 0$ is a lower bound on the probability that $\alpha_i = \alpha^*$.

Note that, other than the Hamming distance, X is not bounded by n. We now consider the following Markov chain.

Definition 1.7 *Let* $\{\mathcal{W}_i^{(n,d)}\}_{i \geq 0}$ *be a time-homogeneous Markov chain with state space* $S = \mathbb{N}$, *a starting distribution of the form*

$$\Pr[\mathcal{W}_0^{(n,d)} = x_0] = \binom{n}{x_0} \left(\frac{d-1}{d}\right)^{x_0} \left(\frac{1}{d}\right)^{n-x_0} \tag{1.1}$$

for any $x_0 \in S$ *and transition probabilities of the form*

$$p(x, x+\delta) = \begin{cases} \frac{1}{k(d-1)} & \text{if } \delta = -1 \\ \frac{d-2}{k(d-1)} & \text{if } \delta = 0 \\ \frac{k-1}{k} & \text{if } \delta = 1 \\ 0 & \text{otherwise} \end{cases} \tag{1.2}$$

if $x > 0$ *and*

$$p(x, x+\delta) = \begin{cases} 1 & \text{if } \delta = 0 \\ 0 & \text{otherwise} \end{cases} \tag{1.3}$$

if $x = 0$.

We make a coupling argument to link the Markov chain $\{\mathcal{W}_i^{(n,d)}\}_{i \geq 0}$ to the algorithm Sch-Coupling. We choose $\mathcal{W}_0^{(n,d)}$ as X_0 in the algorithm. Every iteration of Sch-Coupling corresponds to one step of the Markov chain. The Markov chain always changes by the same amount as X. If the Markov chain reaches state 0, the algorithm must already have terminated as X is

an upper bound to the Hamming distance. Note that a Markov chain can not "terminate" like an algorithm. However, the Markov chain $\{\mathcal{W}_i^{(n,d)}\}_{i\geq 0}$ simply stays in state 0 if it ever reaches that state. It can easily be seen that both the probability distribution of the initial state and the transition probabilities of the Markov chain correspond exactly to the probabilities as described by the coupling argument.

The Markov chain $\{\mathcal{W}_i^{(n,d)}\}_{i\geq 0}$ is closely related (i.e. can be coupled to) a chain that does not allow $\delta = 0$ such that the state changes in every step and the transition probabilities are normalised accordingly. We define the following Markov chain for any $0 < q < \frac{1}{2}$.

Definition 1.8 *Let $\{\mathcal{W}'^{(n,d,q)}_i\}_{i\geq 0}$ be a time-homogeneous Markov chain with state space $S = \mathbb{N}$, a starting distribution of the form*

$$\Pr[\mathcal{W}'^{(n,d,q)}_0 = x_0] = \binom{n}{x_0}\left(\frac{d-1}{d}\right)^{x_0}\left(\frac{1}{d}\right)^{n-x_0} \tag{1.4}$$

for any $x_0 \in S$ and transition probabilities of the form

$$p(x, x+\delta) = \begin{cases} q & \text{if } \delta = -1 \\ 1-q & \text{if } \delta = 1 \\ 0 & \text{otherwise} \end{cases} \tag{1.5}$$

if $x > 0$ and

$$p(x, x+\delta) = \begin{cases} 1 & \text{if } \delta = 0 \\ 0 & \text{otherwise} \end{cases} \tag{1.6}$$

if $x = 0$.

If the context is unambiguous, we use $\{\mathcal{W}_i\}_{i\geq 0}$ and $\{\mathcal{W}'_i\}_{i\geq 0}$ for the two Markov chains.

Let Y be the random variable that represents the number of steps that the Markov chain $\{\mathcal{W}_i\}_{i\geq 0}$ needs to reach 0, i.e. Y is the smallest integer i such that $\mathcal{W}_i = 0$ and let Y' be the corresponding value for the Markov chain $\{\mathcal{W}'_i\}_{i\geq 0}$. In case Y (or Y') does not exist we set $Y = \infty$ ($Y' = \infty$ respectively). We further define X_j as the number of steps needed to reach 0, given that we start in state $\mathcal{W}_0 = j$ and X'_j as the number of steps needed to reach 0, given that we start in state $\mathcal{W}'_0 = j$. We are interested in the probability $\Pr[Y \leq 3n]$ i.e. the probability that we reach the state 0 in at most $3n$ steps.

Using the idea of normalising the transition probabilities of $\{\mathcal{W}_i\}_{i\geq 0}$ we get $q = \frac{1}{k(d-1)-(d-2)}$ and the following lemma

Lemma 1.9 *For $q = \frac{1}{k(d-1)-(d-2)}$ we have*

$$\Pr[Y < \infty] = \Pr[Y' < \infty]$$

and

$$E[Y \mid Y < \infty] = E[Y' \mid Y' < \infty] \frac{k(d-1)}{k(d-1)-(d-2)}$$

Proof Consider the following coupling between the two Markov chains $\{\mathcal{W}_i\}_{i\geq 0}$ and $\{\mathcal{W}'_i\}_{i\geq 0}$. Both Markov chains start in the same starting state. Whenever the Markov chain $\{\mathcal{W}_i\}_{i\geq 0}$ to the left or to the right, the Markov chain $\{\mathcal{W}'_i\}_{i\geq 0}$ does the same. If however the first Markov chain does a step with $\delta = 0$, i.e. stays in the same state, then the second state does not make a step. It can easily be seen that using this coupling the probability for $\{\mathcal{W}'_i\}_{i\geq 0}$ to make a step to the left is $q = \frac{1}{k(d-1)-(d-2)}$. Both have the same probability of ever reaching 0, as they are always in the same state. As the expected number of "non-steps" before changing the state is $\frac{k(d-1)}{k(d-1)-(d-2)}$, the expected number of steps differs by this factor. □

We analyse $\{\mathcal{W}'_i\}_{i\geq 0}$ using the approach from [29], which we reproduce here for completeness.

Lemma 1.10 *For all $j \geq 0$ and $0 < q < \frac{1}{2}$ we have*

$$\Pr[X'_j < \infty] = \left(\frac{q}{1-q}\right)^j$$

and

$$E[X'_j \mid X'_j < \infty] = \frac{j}{1-2q}$$

Proof We prove the first equation first. The lemma clearly holds for $j = 0$, hence we assume $j > 0$. At each step $\mathcal{W}'$ decreases with probability q and increases with probability $1-q$. We consider all paths that start in j and reach 0, such that we make exactly i steps that increase $\mathcal{W}'$ (and thus $i+j$ steps that decrease $\mathcal{W}'$, and reach 0 the first time with the last step. According to the ballot theorem (see [12]) the number of such paths is $\binom{2i+j}{i}\frac{j}{2i+j}$. We weigh each path with its probability of occurrence $\mathcal{P}_q(i,j) = (1-q)^i q^{i+j}$ and obtain the probability that the random walk eventually reaches 0 given that it starts at state j.

$$\begin{aligned}
\Pr[X'_j < \infty] &= \sum_{i\geq 0} \binom{2i+j}{i} \frac{j}{2i+j} \mathcal{P}_q(i,j) \\
&= q^j \sum_{i\geq 0} \binom{2i+j}{i} \frac{j}{2i+j} (q(1-q))^i \\
&= q^j (\mathcal{B}_2(q(1-q)))^j \qquad (1.7)
\end{aligned}$$

where $\mathcal{B}_2(z)$ is the *generalised binomial series* given by

$$\mathcal{B}_2(z) = \sum_{i \geq 0} \binom{2i+1}{i} \frac{z^i}{2i+1} = \frac{1 - \sqrt{1-4z}}{2z} \tag{1.8}$$

and according to [11] we have that for all $r \geq 1$

$$(\mathcal{B}_2(z))^r = \sum_{i \geq 0} \binom{2i+r}{i} \frac{rz^i}{2i+r} \tag{1.9}$$

Replacing the closed from of $\mathcal{B}_2(z)$ in equation 1.7 we obtain

$$\Pr[X'_j < \infty] = q^j \left(\frac{1 - \sqrt{1-4q(1-q)}}{2q(1-q)} \right)^j = \left(\frac{q}{1-q} \right)^j \tag{1.10}$$

Proceeding in a similar manner we get for the expectation

$$\begin{aligned} E[X'_j \mid X'_j < \infty] &= \frac{1}{\Pr[X_j < \infty]} \sum_{i \geq 0} (2i+j) \binom{2i+j}{i} \frac{j}{2i+j} \mathcal{P}_q(i,j) \\ &= \frac{jq^j}{\Pr[X_j < \infty]} \sum_{i \geq 0} \binom{2i+j}{i} (q(1-q))^i \\ &= \frac{jq^j}{q^j (\mathcal{B}_2(q(1-q)))^j} \frac{\mathcal{B}_2(q(1-q)))^j}{\sqrt{1-4q(1-q)}} \\ &= \frac{j}{1-2q} \end{aligned} \tag{1.11}$$

using the fact (see [11]) that for all $r > 0$

$$\frac{(\mathcal{B}_2(z))^r}{\sqrt{1-4z}} = \sum_{i \geq 0} \binom{2i+r}{i} z^i \tag{1.12}$$

□

We can use the conditional probability and expected number of steps to derive the following lemmas.

Lemma 1.11 *For* $0 < q < \frac{1}{2}$,

$$\Pr[Y' < \infty] = \left(\frac{d-1}{d} \frac{q}{1-q} + \frac{1}{d} \right)^n$$

Proof

$$\begin{aligned}\Pr[Y' < \infty] &= \sum_{j=0}^{n} \binom{n}{j} \left(\frac{d-1}{d}\right)^j \left(\frac{1}{d}\right)^{n-j} \Pr[X_j < \infty] \\ &= \sum_{j=0}^{n} \binom{n}{j} \left(\frac{d-1}{d}\right)^j \left(\frac{1}{d}\right)^{n-j} \left(\frac{q}{1-q}\right)^j \\ &= \left(\frac{d-1}{d}\frac{q}{1-q} + \frac{1}{d}\right)^n\end{aligned}$$

using the Binomial Theorem

$$\sum_{i=0}^{n} \binom{n}{i} x^i y^{n-i} = (x+y)^n \tag{1.13}$$

□

We are again interested in the expected number of steps if we reach the state 0 at all.

Lemma 1.12 *For* $0 < q < \frac{1}{2}$,

$$E[Y' \mid Y' < \infty] = \frac{n(d-1)q}{(q(d-2)+1)(1-2q)}$$

Proof

$$
\begin{aligned}
&E[Y' \mid Y' < \infty] \\
&\quad = \sum_{i=0}^{\infty} i \Pr[Y' = i \mid Y' < \infty] \\
&\quad = \frac{1}{\Pr[Y' < \infty]} \sum_{i=0}^{\infty} i \Pr[Y' = i] \\
&\quad = \frac{1}{\Pr[Y' < \infty]} \sum_{i=0}^{\infty} i \left(\sum_{j=0}^{n} \binom{n}{j} \left(\frac{d-1}{d}\right)^{j} \left(\frac{1}{d}\right)^{n-j} \Pr[X'_j = i] \right) \\
&\quad = \frac{1}{\Pr[Y' < \infty]} \sum_{j=0}^{n} \binom{n}{j} \left(\frac{d-1}{d}\right)^{j} \left(\frac{1}{d}\right)^{n-j} \left(\sum_{i=0}^{\infty} i \Pr[X'_j = i] \right) \\
&\quad = \frac{1}{\Pr[Y' < \infty]} \sum_{j=0}^{n} \binom{n}{j} \left(\frac{d-1}{d}\right)^{j} \left(\frac{1}{d}\right)^{n-j} E[X'_j \mid X'_j < \infty] \Pr[X'_j < \infty] \\
&\quad = \left(\frac{d-1}{d}\frac{q}{1-q} + \frac{1}{d}\right)^{-n} \sum_{j=0}^{n} \binom{n}{j} \left(\frac{d-1}{d}\right)^{j} \left(\frac{1}{d}\right)^{n-j} \frac{j}{1-2q} \left(\frac{q}{1-q}\right)^{j} \\
&\quad = \left(\frac{d-1}{d}\frac{q}{1-q} + \frac{1}{d}\right)^{-n} \frac{n}{1-2q} \sum_{j=1}^{n} \binom{n-1}{j-1} \left(\frac{(d-1)q}{d(1-q)}\right)^{j} \left(\frac{1}{d}\right)^{n-j} \\
&\quad = \left(\frac{d-1}{d}\frac{q}{1-q} + \frac{1}{d}\right)^{-n} \frac{n}{1-2q} \frac{(d-1)q}{d(1-q)} \sum_{j=1}^{n} \binom{n-1}{j-1} \left(\frac{(d-1)q}{d(1-q)}\right)^{j-1} \left(\frac{1}{d}\right)^{n-j} \\
&\quad = \left(\frac{d-1}{d}\frac{q}{1-q} + \frac{1}{d}\right)^{-n} \frac{n}{1-2q} \frac{(d-1)q}{d(1-q)} \left(\frac{(d-1)q}{d(1-q)} + \frac{1}{d}\right)^{n-1} \\
&\quad = \frac{d(1-q)}{q(d-2)+1} \frac{n}{1-2q} \frac{(d-1)q}{d(1-q)} \\
&\quad = \frac{n(d-1)q}{(q(d-2)+1)(1-2q)}
\end{aligned}
$$

□

Using lemma 1.9 we then get

Lemma 1.13

$$\Pr[Y < \infty] = \left(\frac{1}{d}\frac{k}{k-1}\right)^{n}$$

Proof We set $q = \frac{1}{k(d-1)-(d-2)}$ and get

$$\begin{aligned}
\Pr[Y < \infty] &= \Pr[Y' < \infty] \\
&= \left(\frac{d-1}{d}\frac{q}{1-q} + \frac{1}{d}\right)^n \\
&= \left(\frac{d-1}{d}\frac{1}{q^{-1}-1} + \frac{1}{d}\right)^n \\
&= \left(\frac{d-1}{d}\frac{1}{k(d-1)-(d-2)-1} + \frac{1}{d}\right)^n \\
&= \left(\frac{d-1}{d}\frac{1}{(k-1)(d-1)} + \frac{1}{d}\right)^n \\
&= \left(\frac{1}{d}\frac{k}{k-1}\right)^n
\end{aligned}$$

□

Lemma 1.14

$$E[Y \mid Y < \infty] = \frac{1}{k - \frac{d}{d-1}} \cdot n \leq 2n$$

Proof We set $q = \frac{1}{k(d-1)-(d-2)}$, use lemma 1.9 and get

$$\begin{aligned}
E[Y \mid Y < \infty] &= E[Y' \mid Y' < \infty]\frac{k(d-1)}{k(d-1)-(d-2)} \\
&= E[Y' \mid Y' < \infty]k(d-1)q \\
&= \frac{n(d-1)q}{(q(d-2)+1)(1-2q)} \cdot k(d-1)q \\
&= \frac{(d-1)^2q^2k}{(q(d-2)+1)(1-2q)} \cdot n \\
&= \frac{(d-1)^2k}{q^{-2}(q(d-2)+1)(1-2q)} \cdot n \\
&= \frac{(d-1)^2k}{((d-2)+q^{-1})(q^{-1}-2)} \cdot n \\
&= \frac{(d-1)^2k}{k(d-1)(k(d-1)-d)} \cdot n \\
&= \frac{d-1}{k(d-1)-d} \cdot n \\
&= \frac{1}{k - \frac{d}{d-1}} \cdot n
\end{aligned}$$

Since we assume $d \geq 2$ and $k \geq 2$ with $d > 2$ or $k > 2$ (i.e. an NP-complete case of (d,k)-CSP) the maximum is at $E[Y \mid Y < \infty] = 2n$ for $d = 3$ and $k = 2$. □

Finally, we can employ Markov's inequality to bound the probability that we reach 0 within $3n$ steps.

Lemma 1.15

$$\Pr[Y \leq 3n] \geq \frac{1}{3}\left(\frac{1}{d}\frac{k}{k-1}\right)^n$$

Proof By Markov's inequality we have

$$\Pr[Y > \lambda E[Y \mid Y < \infty] \mid Y < \infty] < \frac{1}{\lambda}$$

or

$$\Pr[Y \leq \lambda E[Y \mid Y < \infty] \mid Y < \infty] \geq 1 - \frac{1}{\lambda}$$

for $\lambda \geq 1$.

Hence

$$\begin{aligned}
\Pr[Y \leq 3n] &= \Pr[Y \leq 3n \mid Y < \infty] \Pr[Y < \infty] \\
&= \Pr[Y \leq \frac{3}{2}2n \mid Y < \infty] \Pr[Y < \infty] \\
&\geq \Pr[Y \leq \frac{3}{2}E[Y \mid Y < \infty] \mid Y < \infty] \Pr[Y < \infty] \\
&\geq \frac{1}{3}\left(\frac{1}{d}\frac{k}{k-1}\right)^n
\end{aligned}$$

□

By our coupling this probability is also a lower bound for the success probability of the function Sch(F,$3n$). Hence we have to repeat the function Sch(F,$3n$) at most $3\,(d(k-1)/k)^n$ times to find a satisfying assignment. Furthermore the probability that we do not find a satisfying assignment within $3T\,(d(k-1)/k)^n$ repetitions is at most e^{-T}. Note that a single run of Sch(F,$3n$) needs polynomial time only.

1.7 Distance-Regular Graphs

The notion of distance-regular graphs will be used in various places in this thesis. In particular, the main result will rely on this symmetry property for graphs. Distance-regularity is a strictly weaker condition than vertex-transitivity and a strictly stronger condition than regularity. The definition extends to both directed and undirected graphs. For our notation we assume a directed graph, as for any undirected graph we can simply replace every edge with two directed edges. However, for some of the

proofs the undirected case and the directed case (with a girth of at least 3) are treated separately. We begin with some notation for graphs in general.

Let $G = (V, E)$ be a simple directed graph, where V is the set of vertices and $E \in V \times V$ the set of edges. During the whole thesis we assume the graphs to be loop-free, i.e. for all $v \in V$ we have $(v, v) \notin E$. For some $v \in V$, $\Gamma_G(v)$ ($\Gamma(v)$ in unambiguous context) denotes the set of all out-neighbours of v, i.e. $\{u \in V \mid (v, u) \in E\}$ and $d_G(u, v)$ denotes the distance (the length of the shortest path) from u to v. Furthermore $\Gamma_i(v)$ is the set of vertices with distance i from v. We typically use l for the diameter, i.e. the maximum distance among all pairs of vertices, and g for the girth, i.e. the length of the shortest cycle. For regular graphs we further use $\deg(G)$ for the (out-)degree of G. Note that for regular graphs the out-degree and the in-degree are necessarily the same. We now define distance-regularity.

Definition 1.16 *A graph $G = (V, E)$ is called* distance-regular *if it is strongly connected and for any i, j with $0 \leq i, j \leq l$ the number*

$$|\Gamma_i(u) \cap \Gamma_1(v)| =: p_{i,j}$$

is the same for all $u, v \in V$ with $d_G(u, v) = j$

For undirected graphs we can only have $p_{i,j} > 0$ if $i \in \{j-1, j, j+1\}$. A common notation then is $b_j = p_{j+1,j}$, $c_j = p_{j-1,j}$ and $a_j = p_{j,j}$. The array

$$\{b_0, b_1, \ldots, b_{l-1}; c_1, c_2, \ldots, c_l\} \tag{1.14}$$

is then called the *intersection array*. The values a_j are omitted in the intersection array as distance-regularity implies regularity as shown later. We will mostly use the more general notation for directed graphs.

We establish some properties for distance-regular graphs first. This section mainly follows [2].

For undirected graphs, we have for any $u, v \in V$, $d_G(u, v) = d_G(v, u)$. For directed graphs with $g \geq 3$ we have a similar property called *stability*.

Definition 1.17 *A graph $G = (V, E)$ with $g \geq 3$ is called* stable *if for all $0 < s < g$*

$$d_G(u, v) = s \iff d_G(v, u) = g - s$$

Lemma 1.18 *If $G = (V, E)$ is distance-regular, then G is stable.*

Proof Let $u, v \in V$ with $d_G(u, v) = s$. We have $d_G(v, u) \geq g - s$ as otherwise there would be a cycle containing u and v of length less than g, which contradicts the definition of G.

Let

$$u = u_0 \to u_1 \to \cdots \to u_{g-1} \to u_g = u$$

be a cycle in G of length g. By the definition of g there is such a cycle. Furthermore, there are no shortcuts. For any $t < g - 1$ we have

$$u_{t+1} \in \Gamma_{t+1}(u) \cap \Gamma_1(u_t)$$

hence

$$p_{t+1,t} = |\Gamma_{t+1}(u) \cap \Gamma_1(u_t)| > 0 \tag{1.15}$$

Similarly we can conclude

$$p_{0,g-1} > 0 \tag{1.16}$$

Now let $u, v \in V$ be any pair of vertices with $d_G(u, v) = s$, where $0 < s < g$. By 1.15 and 1.16 there is a vertex $w \in V$ such that $(v, w) \in E$ and $w \in \Gamma_{s+1}(u)$. By repeating this argument there is a path from v to u of length $g - s$. Hence $d_G(v, u) \leq g - s$ which concludes the proof. □

This lemma is also true for undirected graphs, i.e. graphs with $g = 2$, however it is a trivial statement then.

Lemma 1.19 *A distance-regular graph $G = (V, E)$ is regular.*

Proof Let $u \in V$. We show that both the in-degree as well as the out-degree of u are independent of the choice of u. Since there are no duplicate edges $|\Gamma_1(u)|$ corresponds to the out-degree of u. Since

$$|\Gamma_1(u)| = |\Gamma_1(u) \cap \Gamma_1(u)| = p_{1,0}$$

this number is independent of G. By lemma 1.18 $|\Gamma_1(g-1)| = p_{0,g-1}$ corresponds to the in-degree, which is independent of u as well. □

Above lemmas are true for both undirected and directed graphs. The next lemma however is only true for directed graphs with $g \geq 3$.

Lemma 1.20 *If $G = (V, E)$ is distance-regular and $g \geq 3$ then either*

$$l = g$$

or

$$l = g - 1$$

Proof Assume $l > g$. Then there are vertices $u, v \in V$ such that $d_G(u, v) = g + 1$. Let

$$u = u_0 \to u_1 \to \cdots \to u_{g-1} \to u_g \to u_{g+1} = v$$

be a shortest path from u to v. Then we have $d_G(u, u_{g-1}) = g - 1$ and $d_G(u_{g-1}, v) = 2$. By stability, $d_G(u_{g-1}, u) = 1$ and $d_G(v, u_{g-1}) = g - 2$. Hence

$$d_G(v, u) \leq d_G(v, u_{g-1}) + d_G(u_{g-1}, u) = g - 1$$

contradicting stability. Assuming $l < g - 1$, then any circuit of length g would have a shortcut, which contradicts the definition of g. □

The following definition is often used.

Definition 1.21 *A distance-regular graph $G = (V, E)$ with $g \geq 3$ is called* long *if $l = g$ and* short *if $l = g - 1$.*

All results in this thesis apply to both short and long graphs.

For both directed and undirected graphs, the value $d_G(v, u)$ is determined by $d_G(u, v)$. Assuming that the graph is clear from the context we use the following notation.

Definition 1.22 *Let $* : \{1, \ldots, l\} \to \{1, \ldots, l\}$ be the following permutation. For an undirected graph, $s^* = s$ for any $0 \leq s \leq l$. For a directed graph with $g \geq 3$, $s^* = g - s$ if $0 < s < g$, also $0^* = 0$ and $g^* = g$.*

A direct consequence of this definition as well as lemmas 1.18 and 1.20 then is

Lemma 1.23 *Let $G = (V, E)$ be a distance-regular graph. Then any $u, v \in V$ with $d_G(u, v) = s$ we have $d_G(v, u) = s^*$.*

We show that for distance-regular graphs and any s, $|\Gamma_s(u)|$ is independent of u. The first proof works for any distance-regular graphs, while there is a shorter proof for undirected graphs in particular at the end of the section.

We use the definition of *distance matrices*.

Definition 1.24 *Let $G = (V, E)$ be any graph with $|V| = n$. For $0 \leq s \leq l$ the* distance matrices *A_s are $n \times n$ matrices with rows and columns indexed by the vertices such that*

$$(A_s)_{u,v} = \begin{cases} 1 & \text{if } d_G(u, v) = s \\ 0 & \text{otherwise} \end{cases}$$

Hence A_1 is the adjacency matrix of G.

This leads us to the following lemma.

Lemma 1.25 *For $0 \leq s \leq l - 1$,*

$$A_1 A_s = \sum_{i=0}^{s+1} p_{s^*, i^*} \cdot A_i$$

with $p_{s^*,(s+1)^*} > 0$.

Proof

$$\begin{aligned}(A_1A_s)_{u,v} &= |\{w \in V \mid d_G(u,w) = 1 \text{ and } d_G(w,v) = s\}| \\ &= |\{w \in V \mid d_G(v,w) = s^* \text{ and } d_G(u,w) = 1\}| \\ &= p_{s^*,i^*} \text{ if } d_G(u,v) = i \\ &= (\sum_i p_{s^*,i^*} A_i)_{u,v}\end{aligned}$$

For $i > s+1$ we have $p_{s^*,i^*} = 0$, hence the sum stops at $s+1$. Also if $i = s+1$, i.e. $d_G(u,v) = i$, there is a vertex w on the closest path from u to v with $d_G(u,w) = s$ and $d_G(w,v) = 1$. Hence $p_{s^*,(s+1)^*} > 0$. □

A direct consequence is that A_s is a polynomial in A_1.

Lemma 1.26 *For any $0 \leq s \leq l$, A_s is a polynomial in A_1 of degree s.*

Proof Clearly $A_0 = I = (A_1)^0$ where I denotes the identity matrix and $A_1 = (A_1)^1$. Assuming for $0 \leq s < l$ the lemma is true for any $s' \leq s$. By lemma 1.25 we get

$$A_{s+1} = \frac{1}{p_{s^*,(s+1)^*}} \left(A_1A_s - \sum_{i=0}^{s} p_{s^*,i^*} \cdot A_i \right)$$

which is clearly a polynom in A_1 of degree $s+1$. □

We therefore arrive at the lemma that the number of vertices with distance s from u is independent of u.

Lemma 1.27 *Let $G = (V,E)$ be a distance-regular graph. For any $u, v \in V$ and any $s \leq l$*

$$|\Gamma_s(u)| = |\Gamma_s(v)|$$

Proof We have

$$|\Gamma_s(u)| = \sum_{w \in V} (A_s)_{u,w}$$

and

$$\sum_{w \in V} (A_1)_{u,w} = \sum_{w \in V} (A_1)_{v,w}$$

follows directly from regularity. It can easily be seen that

$$\sum_{w \in V} (A_1^{s'})_{u,w} = \sum_{w \in V} (A_1^{s'})_{v,w}$$

for any s', hence since A_s is a linear combination of matrices with this property the lemma follows directly. □

Above proof works for both directed and undirected graphs. However there is a shorter and simpler proof for undirected graphs.

Lemma 1.28 *Let $G = (V, E)$ be an undirected distance-regular graph. For any $u, v \in V$ and any $s \leq l$*

$$|\Gamma_s(u)| = |\Gamma_s(v)|$$

Proof Let $u, v \in V$ arbitrarily. Clearly

$$|\Gamma_0(u)| = |\Gamma_0(v)| = 1$$

Let $s \leq l$ and assume the lemma is true for $s - 1$. By a double counting argument the number of edges between the sets $|\Gamma_s(u)|$ and $|\Gamma_{s-1}(u)|$ is

$$b_{s-1} \cdot |\Gamma_{s-1}(u)| = c_s \cdot |\Gamma_s(u)|$$

Both b_{s-1} and c_s are independent of u and both are greater than 0 by connectivity. Furthermore $|\Gamma_{s-1}(u)|$ is independent of u by the induction hypothesis. Hence $|\Gamma_s(u)|$ is also independent of u. □

Since $|\Gamma_s(u)|$ is independent of u we will use the notion $|\Gamma_s|$.

For a very detailed discussion on undirected distance-regular graphs see [1]. Directed distance-regular graphs are discussed in various papers such as [2] and [15].

Chapter 2

Schöning's Algorithm using Search Graphs

2.1 The Algorithm

In this chapter we consider the (d,k)-CSP problem. Schöning's algorithm solves such a problem in $(d(k-1)/k)^n \cdot \text{poly}(n)$ many steps as discussed in the introductory chapter. It does so by repeatedly calling a procedure that runs in polynomial time and has a success probability of $\left(\frac{1}{d}k/(k-1)\right)^n \Theta(1)$.

We now consider a modification of Schöning's algorithm. We only modify the polynomial time procedure, while the repetition idea remains the same. Hence during the whole thesis we will mostly consider the success probability of a single run of our procedure. The consequences for the expected number of repetitions then is implied. In addition to the parameters of Schöning's algorithm the modified version is also given a (directed) graph $G = (V, E)$ on d vertices with a mapping from the colour list of the formula to the vertices of that graph. We call the graph G the *search graph*. For simplicity we will always assume that both the colour list and the set of vertices is $\{1, \ldots, d\}$ and we are using the trivial mapping. Throughout this chapter we will assume that the graph G is distance-regular. When Schöning's algorithm chooses to flip a variable, it assigns randomly a new value to that variable. Instead, our modified algorithm Sch-Searchgraph (Algorithm 3) limits the possible values a variable can be flipped to. Our modified algorithm chooses the new value randomly among the neighbours in the search graph. If we use the complete graph K_d as our search graph we get the original algorithm by Schöning.

Algorithm 3 Sch-Searchgraph (F,t,G)

$\alpha \in_{\text{u.a.r.}} \{1,\dots,d\}^{\text{vbl}(F)}$
if α satisfies F **then**
  return (true, α)
end if
for $i = 1$ **to** t **do**
  $C \in \text{vlt}(F,\alpha)$ (using any selection rule)
  $x \in_{\text{u.a.r}} \text{vbl}(C)$
  $a_{\text{new}} \in_{\text{u.a.r}} \Gamma_G(\alpha(x))$
  $\alpha(x) \leftarrow a_{\text{new}}$
  if α satisfies F **then**
    return (true, α)
  end if
end for
return (false, nil)

2.2 Selection Rules

In this section we formalise the concept of a selection rule. As we usually assume an arbitrary selection rule we have to define precisely how powerful such a rule can be.

On an intuitive level, the selection rule is given knowledge about any former and current states of the algorithm. It then chooses a clause $C \in F$ which is not satisfied by the current assignment. We assume unbounded computing power and access to random numbers for the selection rule. This part is based on a similar discussion in Andrei Giurgiu's Master thesis [9], where he addresses the problem with regard to universally good selection rules, so called "angels". We will consider angels again in the worst-case analysis.

In order to argue about asymptotic behaviour of selection rules we need to define selection rules that work for formulas with an increasing number of variables. We define a class of formulas as follows.

Definition 2.1 *A class of (d,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 $|vbl(F)| \geq n$.*

Further we define the set $\mathcal{A}_{\text{vbl}(F)}$ representing the set of possible sequences of assignments on the formula F.

Definition 2.2 *For some variable set $vbl(F)$ let $\mathcal{A}_{vbl(F)}$ be the set of all nonempty strings $\alpha_1 \dots \alpha_i$ with $i \geq 1$ such that α_j is an assignment on $vbl(F)$ for $1 \leq j \leq i$. Furthermore, for any $1 \leq j < i$, we have that α_j and α_{j+1} only differ in one position.*

We are now ready to define a selection rule.

Definition 2.3 *A* selection rule *for a class $\mathcal{F}$ of (d,k)-CNF formulas is a set of maps $\sigma_F : \mathcal{A}_{vbl(F)} \times \{0,1\}^* \to F \cup \{\emptyset\}$ indexed by $F \in \mathcal{F}$. The functions are subject to the following conditions:*

- *If there is a j such that α_j satisfies F, then σ_F returns $\emptyset$.*
- *Else σ_F returns $C \in vlt(F, \alpha_i)$.*

The second input is assumed to be a random string. The notion $\sigma_F(\mathfrak{a}, \mathfrak{r})$ will be used where $\mathfrak{a}$ is the history of the assignment α during the algorithm and $\mathfrak{r}$ is a random bit string.

With respect to this formal notion of selection rules we can rewrite our algorithm Sch-Searchgraph as in Algorithm 4.

Algorithm 4 Sch-Searchgraph (F,t,G,σ_F)

$\alpha \in_{\text{u.a.r.}} \{1,\dots,d\}^{\text{vbl}(F)}$
if α satisfies F **then**
  return (true, α)
end if
for $i = 1$ **to** t **do**
  $C \leftarrow \sigma_F(\mathfrak{a}, \mathfrak{r})$
  $x \in_{\text{u.a.r}} \text{vbl}(C)$
  $a_{\text{new}} \in_{\text{u.a.r}} \Gamma_G(\alpha(x))$
  $\alpha(x) \leftarrow a_{\text{new}}$
  if α satisfies F **then**
    return (true, α)
  end if
end for
return (false, nil)

2.3 The Main Theorem

We are interested in the success probability of Sch-Searchgraph(F,t,G,σ_F) (i.e. the probability that the procedure returns true) for different G and a reasonably small t, i.e. a t, such that a single run of Sch-Searchgraph runs in polynomial time. We obviously require that F is satisfiable in this analysis, as the success probability is 0 otherwise. Like with the classical Schöning's algorithm, we can then construct a randomized algorithm that decides satisfiability by repeating Sch-Searchgraph several times. The main theorem is the following.

Theorem 2.4 *Given a satisfiable (d,k)-CNF formula F with $d \geq 2$ and $k \geq 2$ and $d > 2$ or $k > 2$, t large enough, and a distance-regular graph G on d vertices and any selection rule the success probability of Sch-Searchgraph is at least*

$$\left(\frac{1}{d} \cdot \frac{k}{k-1}\right)^n \cdot \Theta(1)$$

In other words, the success probability is not worse than for the classical Schöning's algorithm.

We require $d > 2$ or $k > 2$, as we are not interested in $(2,2)$-CSP, i.e. 2-SAT, as this problem can be solved in polynomial time anyway. In the classical Schöning's algorithm we set $t = 3n$, here our constant is different, but we still set $t = O(n)$, such that Sch-Searchgraph runs in the same asymptotic time. Our search graph G has to be distance-regular, but can be either directed or undirected. Basically, distance-regularity is the symmetry property that ensures that the success probability does not depend on the labelling of the graph.

Apart from the last section, the whole chapter consists of a proof for the main theorem. This section here only gives an outline. In the first section we modify Sch-Searchgraph, such that it keeps track of some additional variables. This process then is easier to analyse.

In the second section we analyse the auxiliary variables introduced in the first section, defining *forced shifting events* that can occur whenever a clause selected by the selection rule has more than one literal different from the satisfying assignment. We show that forced shifting events can only increase the success probability of the algorithm. Hence a selection rule that minimises the success probability never chooses a clause with more than one literal different from the satisfying assignment. Furthermore we simplify the process such that the selection rule is always able to choose such a clause. This process is then independent of the input formula and considers only one relevant satisfying assignment.

In the classical algorithm the coupling removed all degrees of freedom that came from the selection rule, as any variable either corresponds to the satisfying assignment or not. In our case however, variables that do not correspond to the satisfying assignment might be far away or close in the search graph, in particular if the graph is sparse. As the coupling does not address this problem, the third section shows that even though there is still some degree of freedom (which makes it no Markov chain for instance), the success probability is not affected by it. Furthermore the success probability is given depending on some values λ_s that satisfy some system of equations. The proof idea behind this section is that at any point in time the variables different from the satisfying assignment can be regarded independently of

each other as some kind of tree branching process. The order in which we build this tree does however not affect the success probability. In this section we first analyse the process with $t = \infty$ for simplicity and then show that this is not necessary afterwards. This is very similar to the classical analysis, where we first analyse the hitting probability of a Markov chain and only later bound the expected hitting time.

In the fourth section we then analyse all solutions of the system of equations the values for λ_s satisfy. We show that for any possible solution of the system, and independent of the graph G, plugging in those values to the success probability gives us the main theorem. In this section we also deal with the problem that the system of equations might have other solutions. We show that only the solutions corresponding to the main theorem are possible values for the λ_s.

In the last section we show that the analysis is indeed tight. We give a formula, such that the success probability is within a polynomial factor of the lower bound given in the main theorem, independent of the search graph or the selection rule. The formula is chosen in a way, such that the selection rule has to choose a worst-case clause most of the time. This then allows us to reuse the tools we already established in the previous sections to show that the analysis of the coupled process does not only provide a lower bound for the success probability of Sch-Searchgraph, but also an upper bound (within a polynomial margin) of the success probability with the given formula.

An interesting special case for the graph G is the hypercube. If $d = 2^q$ we can choose the q-dimensional hypercube as the search graph. Note that hypercubes are distance-regular. Using this search graph is identical to replacing every variable with q new binary variables and running the binary version of Schöning's algorithm on our new formula. We already know that one run of Schöning's algorithm on this problem has a success probability of at least $(\frac{1}{d}\frac{qk}{qk-1})^{qn} < (\frac{1}{4}\frac{k}{k-1})^n$. As a direct consequence of above main theorem this analysis is not tight, hence the corresponding binary formula can be solved faster than a general qk-CNF. The two-dimensional hypercube also allows for an analysis using Markov chains that is closer to the original analysis. This alternative analysis can be found in the appendix.

2.4 The Coupling Argument

In this section we couple the algorithm to a process that is easier to analyse. We use the same definitions for an assignment α^* and the function ξ as in the introductory chapter. The definitions are restated here.

Let α^* be a particular assignment that satisfies F. For some $x \in \text{vbl}(F)$, we call $\alpha^*(x)$ the origin of x. Furthermore $\xi(C)$ is defined as

Definition 2.5 *Assuming some ordering on vbl(F), let $\xi : F \to vbl(F)$ map a clause $C \in F$ to the smallest variable $x \in vbl(C)$ with $(x, a) \in C$, such that $a \neq \alpha^*(x)$.*

Note that there is always a variable x not set to $\alpha^*(x)$, as otherwise α^* would not satisfy that clause.

We define an auxiliary algorithm that follows the same structure as the Sch-Searchgraph, but it keeps track of additional values. In particular, it keeps an *auxiliary assignment* β. We first need some definitions for that. We introduce the *auxiliary variable set*.

Definition 2.6 *Let*

$$W = \{w_1, w_2, \ldots\}$$

be an infinite set with $w_i \notin vbl(F)$. Then

$$S = vbl(F) \cup W$$

is called the auxiliary variable set. *The variables in S are called* auxiliary variables.

While α^* is a particular satisfying assignment on $\text{vbl}(F)$, the assignment β^* will play the corresponding role for the auxiliary variable set.

Definition 2.7 *Let*

$$\begin{aligned} \beta^*(x) &= \alpha^*(x) \qquad \text{for all } x \in vbl(F) \\ \beta^*(w) &= 1 \qquad \text{for all } w \in W \end{aligned}$$

be an assignment on the auxiliary variable set.

In the analysis later, β^* will be the target assignment and the success of the algorithm will be measured based on if it reaches β^*. The main idea behind defining such an assignment on an infinite set of variables is to simplify the analysis by ensuring that the distance to the satisfying assignment is not bounded by n, i.e. there are no border cases to consider. This corresponds to defining an infinite Markov chain in the classic analysis of Schöning's algorithm.

At the beginning of the algorithm we set β to the initial assignment α for all variables $x \in \text{vbl}(F)$ and all variables $w \in W$ we set to 1, which corresponds to β^* of these values. For the whole algorithm, β can be regarded as an approximation to α. In every time step, the selection rule σ_F chooses a clause

C. A randomly chosen variable x in $\mathrm{vbl}(C)$ is then flipped in the assignment α. In the assignment β we do something similar. Let $s = d_G(\alpha^*(x), \alpha(x))$ be the distance of x to the origin at the beginning of the iteration. We consider three cases

- If the flipped variable x corresponds to $\xi(C)$, we flip a variable $x' \in S$ with $d_G(\beta^*(x'), \beta(x')) = s$
- If x is not $\xi(C)$ and has distance 0, we flip some variable in S with distance 0
- If x is not $\xi(C)$ and has not distance 0, we both flip a variable with distance 0 and one with distance s

We will use the notation

$$S_s(\beta) = \{x \in S \mid d_G(\beta^*(x), \beta(x)) = s\}$$

for the the set of variables in S with distance s to the origin.

Whenever we flip a variable in α we flip one or two variables in β. These variables change their values in a coupled way. We want to ensure that if those values have the same distance to the origin before, they also have to same distance to the origin afterwards. A random number r between 1 and the (out-)degree $\deg(G)$ is chosen and all of those variables flip to the rth neighbour of the current value in G, where the rth neighbour is defined according to the following ordering.

Definition 2.8 *Let $G = (V, E)$ be a distance-regular graph. With respect to some origin $u \in V$ and some vertex $v \in V$, we define a total order $\leq_{u,G}$ on the set $\Gamma(v)$ such that for $x, y \in \Gamma_G(v)$ we have $x \leq y$ if $d_G(u, x) < d_G(u, y)$. To get a total order on the neighbours of v we choose some arbitrary ordering among vertices with equal distance to the origin. We further define the notion $\Gamma^{(r)}_{G,u}(v)$ for the rth neighbour of v with respect to the total order $\leq_{u,G}$.*

In our case the origin will always be $\alpha^*(x)$ or $\beta^*(x)$, while the graph will be the search graph, so we will leave out the subscript and write $\Gamma^{(r)}(\alpha(x))$ instead of $\Gamma^{(r)}_{\alpha^*(x),G}(\alpha(x))$. Note that we require distance-regularity, as otherwise two variables that had the same distance to the origin before might have a different distance to the origin after we flip both to the rth neighbour.

The algorithm Sch-Searchgraph-Coupling (Algorithm 5) now gives the coupling between α and β.

As Sch-Searchgraph-Coupling is identical to Sch-Searchgraph, except that we are keeping track of the auxiliary assignment, it is evident that the two algorithms perform the same number of iterations when given the same

Algorithm 5 Sch-Searchgraph-Coupling (F,t,G,σ_F,α^*)

$\alpha \in_{\text{u.a.r.}} \{1,\dots,d\}^{\text{vbl}(F)}$
$\beta(x) \leftarrow \alpha(x)$ for all $x \in \text{vbl}(F)$
$\beta(w) \leftarrow 1$ for all $w \in W$
if α satisfies F **then**
  return (true, α)
end if
for $i = 1$ **to** t **do**
  $C \leftarrow \sigma_F(\mathfrak{a},\mathfrak{r})$
  $x \in_{\text{u.a.r}} \text{vbl}(C)$
  $r \in_{\text{u.a.r}} \{1,\dots,\deg(G)\}$
  $s \leftarrow d_G(\alpha^*(x),\alpha(x))$
  $\alpha(x) \leftarrow \Gamma^{(r)}(\alpha(x))$
  if $x = \xi(C)$ **then**
    $x' \in S_s(\beta)$
    $\beta(x') \leftarrow \Gamma^{(r)}(\beta(x'))$
  else
    $x'' \in S_0(\beta)$
    $\beta(x'') \leftarrow \Gamma^{(r)}(\beta(x''))$
    if $s \neq 0$ **then**
      $x^* \in S_s(\beta)$
      $\beta(x^*) \leftarrow \Gamma^{(r)}(\beta(x^*))$
    end if
  end if
  if α satisfies F **then**
    return (true, α)
  end if
end for
return (false, nil)

input formula, selection rule, search graph and random bits. What remains to show is a relation between α and β.

Let α_i and β_i be the values for α and β at iteration i and let $i_{\max}$ be the iteration where the algorithm returns.

Lemma 2.9 *For all possible runs of Sch-Searchgraph-Coupling, for all* $0 \leq i \leq i_{max}$ *and for all* $s \in \mathbb{N}$ *we have*

$$|\{x \in vbl(F) \mid d_G(\alpha^*(x),\alpha_i(x)) = s\}| \leq |S_s(\beta_i)|$$

Proof The proof is an induction over time. The property is true for the

initial assignment as we have $|S_0(\beta_0)| = \infty$ and for all $s \geq 1$ we have

$$|\{x \in \text{vbl}(F) \mid d_G(\alpha^*(x), \alpha_0(x)) = s\}| = |S_s(\beta_0)|$$

by the definition of the initial assignment β on S. In every iteration i, whenever we flip a variable $x \in \text{vbl}(F)$ with $d_G(\alpha^*(x), \alpha_i(x)) = s$, we also flip a variable $x' \in S_s(\beta_i)$. Since we flip both to the rth neighbour, the lemma holds. □

This immediately gives us the following.

Lemma 2.10 *$\beta = \beta^*$ implies $\alpha = \alpha^*$*

In order to get a lower bound on the success probability we can now analyse the probability that we reach the state $\beta = \beta^*$.

2.5 More About Selection Rules

We have established in the previous section that in the algorithm Sch-Searchgraph-Coupling β reaches the origin slower than α. We will now analyse how the assignment β evolves decoupled from α. In order to do that we consider *strong selection rules*. The selection rule chooses any auxiliary variable $\xi' \in S$ with $\beta(\xi') \neq \beta^*(\xi')$ and a flag f determining one of two possible actions, an *expansion event* or a *forced shifting event*. In an expansion event we flip the returned variable with probability $\frac{1}{k}$ and flip some variable with distance 0 else. In a forced shifting event we flip the returned variable with probability 1.

In this section we first formalise these selection rules and give the algorithm the selection rules work on. We then show that any selection rule on Sch-Searchgraph can indeed be represented by a strong selection rule. As the last step of the section we determine which selection rule gives the lowest success probability.

Let us first define $\mathcal{B}_S$ similar to how we defined $\mathcal{A}_{\text{vbl}(F)}$ in definition 2.2.

Definition 2.11 *For some auxiliary variable set S let $\mathcal{B}_S$ be the set of all nonempty strings $\beta_1 \dots \beta_i$ with $i \geq 1$ such that β_j is an assignment on S for $1 \leq j \leq i$. Furthermore, for any $1 \leq j < i$, we have that β_j and β_{j+1} only differ in one position.*

We now define a strong selection rule.

Definition 2.12 *A* strong selection rule *is a set of maps $\sigma^* : \mathcal{B}_S \times \{0,1\}^* \to (S \times \{0,1\}) \cup \{\emptyset\}$. The functions are subject to the following conditions:*

- *If there is an j such that $\beta_j = \beta^*$, then σ^* returns $\emptyset$.*

- *Else* σ^* *returns* (ξ', f) *with* $\xi' \in \{x \in S \mid \beta_i(x) \neq \beta^*(x)\}$ *and* $f \in \{0, 1\}$.

We will again use the notion $\sigma^*(\mathfrak{b}, \mathfrak{r})$ where $\mathfrak{b}$ is the history of the assignment β during the algorithm and $\mathfrak{r}$ is a random bit string.

Using this definition we can decouple the algorithm from α and get Sch-Searchgraph-Decoupled (Algorithm 6). As the decoupled algorithm does not depend on F anymore (but still on $\mathrm{vbl}(F)$ and k) the input format is changed accordingly. Furthermore β^* rather than α^* is passed to the algorithm.

Algorithm 6 Sch-Searchgraph-Decoupled (vbl(F),k,t,G,σ^*,β^*)

```
β(x) ← α(x)        for all x ∈ vbl(F)
β(w) ← 1           for all w ∈ W
if β = β* then
  return (true, β)
end if
for i = 1 to t do
  (ξ', f) ← σ*(𝔟, 𝔯)
  r ∈u.a.r [0, 1)
  if r < 1/k or f = 1 then
    β(ξ') ∈u.a.r Γ_G(β(ξ'))
  else
    x'' ∈ S_0(β)
    β(x'') ∈u.a.r Γ_G(β(x''))
  end if
  if β = β* then
    return (true, β)
  end if
end for
return (false, nil)
```

We first show that the new definition for a selection rule is indeed stronger than the original definition.

Lemma 2.13 *For every selection rule* σ_F *there is a strong selection rule* σ^* *such that the success probability of the call Sch-Searchgraph(F,t,G,*σ_F*) is not lower than of the call Sch-Searchgraph-Decoupled(vbl(F),k,t,G,*σ^**,*β^**).*

Proof Using a coupling argument we construct a selection rule σ^* such that the assignment β in Sch-Searchgraph-Decoupled behaves like in Sch-Searchgraph-Coupling with the corresponding selection rule σ_F. For every iteration of Sch-Searchgraph, we do one or two iterations of Sch-Searchgraph-Decoupled. We pick the same initial assignment β_0 and

then observe that in every iteration of of Sch-Searchgraph-Coupling, there are exactly three variables that might be flipped.

- $x' \in S_s(\beta)$, if $x = \xi(C)$
- $x'' \in S_0(\beta)$, if $x \neq \xi(C)$
- $x^* \in S_s(\beta)$, if $x \neq \xi(C)$ and $s \neq 0$

Note that s denotes the distance of x to the origin. The algorithm does not specify how exactly those three variables are chosen among their respective sets. However, it does not influence the success probability and we simply assume here that it is some well-defined rule such that we can pick the same variables.

The selection rule σ^* now does the following. For every iteration of Sch-Searchgraph-Coupling, pick $\xi' = x'$ and $f = 0$ (i.e. not forcing a shifting event), where x' is the corresponding variable in the run of Sch-Searchgraph-Coupling, that might be flipped. If $x = \xi(C)$, we flip both x' in the run of Sch-Searchgraph-Coupling and ξ' in the run of Sch-Searchgraph-Decoupled. Otherwise we flip x'' in both algorithms. We also couple those flips such that the new value assigned is the same. It can easily be seen that the probabilities are the same in both algorithms. If $x \neq \xi(C)$ and x has distance $s \neq 0$, then the selection rule σ^* selects $\xi' = x^*$ and forces the shifting event in the next iteration of Sch-Searchgraph-Decoupled. Again, the variable x^* is therefore flipped in both algorithms. Using this coupling argument, both algorithms have the same assignment β in every iteration. Note that both the fact that Sch-Searchgraph-Coupling might terminate before $\beta = \beta^*$ and the fact that one iteration of Sch-Searchgraph-Coupling might correspond to two iterations of Sch-Searchgraph-Decoupled can only decrease the success probability of Sch-Searchgraph-Decoupled. □

We also consider a *weak selection rule* that works similar to a strong selection rule but is not allowed to force any shifting events.

Definition 2.14 *A* weak selection rule *is a set of maps* $\sigma' : \mathcal{B}_S \times \{0,1\}^* \to S \cup \{\emptyset\}$. *The functions are subject to the following conditions:*

- *If there is an* j *such that* $\beta_j = \beta^*$, *then* σ' *returns* $\emptyset$.
- *Else* σ' *returns* $\xi' \in \{x \in S \mid \beta_i(x) \neq \beta^*(x)\}$.

When using weak selection rules, the algorithm Sch-Searchgraph-Decoupled then requires a few minor changes in syntax as in Algorithm 7.

In order to get a lower bound on the success probability of Sch-Searchgraph we are interested in the infimum of the success probability for a fixed $n = |\mathrm{vbl}(F)|$, k, t, G and β^* and all possible strong selection rules σ^*. The variable

Algorithm 7 Sch-Searchgraph-Decoupled (vbl(F),k,t,G,σ',β^*)

$\beta(x) \leftarrow \alpha(x)$ for all $x \in \mathrm{vbl}(F)$
$\beta(w) \leftarrow 1$ for all $w \in W$
if $\beta = \beta^*$ **then**
  return (true, β)
end if
for $i = 1$ **to** t **do**
  $\xi' \leftarrow \sigma'(\mathfrak{b}, \mathfrak{r})$
  $r \in_{\text{u.a.r}} [0, 1)$
  if $r < \frac{1}{k}$ **then**
    $\beta(\xi') \in_{\text{u.a.r}} \Gamma_G(\beta(\xi'))$
  else
    $x'' \in S_0(\beta)$
    $\beta(x'') \in_{\text{u.a.r}} \Gamma_G(\beta(x''))$
  end if
  if $\beta = \beta^*$ **then**
    return (true, β)
  end if
end for
return (false, nil)

set $\mathrm{vbl}(F)$ clearly does not affect the success probability other than by the size n. Let n, k, t, G and β^* be fixed and let f_{σ^*} denote the success probability of Sch-Searchgraph-Decoupled(vbl(F),k,t,G,σ^*,β^*). We now show that for any strong selection rule σ^* there is a weak selection rule σ' such that the success probability is not higher.

Lemma 2.15 *For any fixed n, k, t, G, β^* and every strong selection rule σ^* there is a weak selection rule σ' with*

$$f_{\sigma'} \leq f_{\sigma^*}$$

Proof The selection rule σ' imitates the selection rule σ^* as long as there is no forced shifting event. If σ^* returns $(\xi', 1)$, i.e. a forced shifting event on some variable ξ', the weak selection rule σ' returns ξ' until the value of ξ' changes. During this process several variables might be flipped. Afterwards σ' imitates σ^* again with the history $\mathfrak{b}$ as if the changed value of $\beta(\xi')$ was achieved with a forced shifting event directly. Note that if the run with selection rule σ' is in iteration i, it might imitate a run with σ^* at iteration $i' < i$. Given that ξ' changes its value the probabilities of its new value are exactly the same as if the shift was forced. Furthermore σ' behaves exactly the same as σ^* if no shift is forced. With probability exactly f_{σ^*} we reach a state where all variables touched by imitating σ^* are set to β^*, while only

variables that were flipped while repeatedly trying to flip ξ' might not be correct yet. Hence the success probability using σ' cannot be bigger than the success probability using σ^*. □

We can therefore limit the discussion to weak selection rules only.

2.6 A Tree Branching Process

We now want to argue that the success probability of Sch-Searchgraph-Decoupled is independent of the weak selection rule σ' and describe the success probability in terms of n, d, k and the graph G. For this, we define a *labelled tree branching process* and couple a run of Sch-Searchgraph-Decoupled to this process.

Definition 2.16 *Given a set L, called the* label space, *a* labelled tree branching process *is given by a* set of roots V_0, *for every root $v \in V_0$ a* label *or* distance $\delta(v) \in L$ *and a* branching rule *specifying the probability distribution for the number and labels of the children of a node. We further require that the branching rule only depends on the label of the parent.*

The requirement that in every branching step the labels of the newly appended nodes only depend on the parent can be considered a Markovian property, as any node therefore only depends on its parent rather than the whole tree. Also, different subtrees are therefore independent of each other. Note that we do not specify an order in which the tree is branched as this would not influence the properties we will discuss in this section because of this Markovian property. Our definition for a labelled tree branching process can be considered a labelled version of a Galton-Watson process [8] that considers trees with the number of children of a node randomly chosen and independent of each other.

In the following definitions and lemma we will use the assumption that G is distance-regular (directed or undirected) with diameter l and degree $\deg(G)$. The notion $p_{s',s}$ refers to the corresponding values in the definition of distance-regularity (definition 1.16), i.e. any vertex with distance s to the origin has exactly $p_{s',s}$ out-neighbours of distance s' to the origin. For the whole section we assume that we run Sch-Searchgraph-Decoupled with a weak selection rule.

We now define a particular instance of labelled tree branching process that models a run of Sch-Searchgraph-Decoupled.

Definition 2.17 *Let $L = \{0, \dots, l\}$ and T is the labelled tree branching process such that $|V_0| = n$ and for every $v \in V_0$ the distance is chosen randomly with the*

probability distribution

$$\Pr[\delta(v) = s] = \frac{|\Gamma_s|}{d}$$

for any $0 \leq s \leq l$ *independent of each other.*

Further any node v *with* $s = \delta(v) \neq 0$ *has*

- *One child with distance* 1 *and one child with distance* s *with probability* $\frac{k-1}{k}$
- *One child with distance* s' *with probability* $\frac{1}{k} \cdot \frac{p_{s',s}}{deg(G)}$ *for any* $0 \leq s' \leq l$

and any node v *with* $\delta(v) = 0$ *has zero children (i.e. is a leaf).*

Hence the tree branching process starts with exactly n roots with a random label. With a probability of $\frac{k-1}{k}$ a node v with distance not equal to 0 has 2 children, one of distance 1 and one of distance $\delta(v)$. With a probability of $\frac{1}{k}$ the node v only has a single child. Given that it only has a single child, the probability that this child has a distance of s' is given by $\frac{p_{s',\delta(v)}}{\deg(G)}$. Note that for a vertex u in the graph G with distance $\delta(v)$ to the origin, this probability corresponds to the probability that a random neighbour of u has distance s' to the origin. Also, a node of distance 0 never has any children.

Let V be the set of all nodes of T. For some node $v \in V$, let Y_v be the number of nodes in the subtree rooted at v, including v. Further, let Y'_v be the number of non-leaf nodes rooted at v. The number of nodes in the whole forest and the number of non-leaf nodes are denoted by Y and Y' respectively. We then have the following relation between the algorithm Sch-Searchgraph-Decoupled and the labelled tree branching process T

Lemma 2.18 *The probability that Sch-Searchgraph-Decoupled(vbl(F),k,t,G,σ',β^*) terminates successfully is exactly* $\Pr[Y' \leq t]$.

Proof We couple a run of Sch-Searchgraph-Decoupled to T using a function $\tau : V \to S$ that maps every node of the tree to an auxiliary variable. We have a notion of time here, as the algorithm gives a particular order in which the variables are flipped. For the tree branching process, every iteration of Sch-Searchgraph-Decoupled corresponds to adding the children of a single node. Consider the initial assignment β_0. For every variable $x \in \text{vbl}(F)$ there is a node $v \in V_0$ with $\tau(v) = x$ and $\delta(v) = d_G(\beta^*(x), \beta(x))$, as there are exactly n such nodes and its label $\delta(v)$ is distributed exactly as the distance of a randomly chosen value to the variable's origin. In every iteration of Sch-Searchgraph-Decoupled, the selection rule σ' selects a variable $\xi' \in S$ which has a distance to the origin of at least 1. In the unfinished tree at this point, there is exactly one leaf node v with $\tau(v) = \xi'$ and $\delta(v)$ being the distance of ξ' to the origin. If the algorithm Sch-Searchgraph-Decoupled flips the variable ξ', we attach a single node v' to v with $\tau(v') = \xi'$ and

$\delta(v)$ corresponding to the new distance of ξ' to the origin. If we flip some other variable $x'' \neq \xi'$ we instead attach two nodes to v, a variable v' with $\tau(v') = \xi'$ and $\delta(v') = \delta(v)$, reflecting that the value of ξ' did not change, and one node v' with $\tau(v') = x''$ and $\delta(v') = 1$, reflecting that x'' was flipped from distance 0 to distance 1.

It can easily be seen that the probabilities correspond exactly to the probabilities specified by the definition of T. Furthermore, at any point there is a variable $x \in S$ with $\beta(x) \neq \beta^*(x)$ if and only if the tree has a single leaf v with $\tau(v) = x$ and $\delta(v) = d_G(\beta^*(x), \beta(x))$. For any variable $x \in S$ with $\beta(x) \neq \beta^*(x)$ there is no leaf in the tree with $\tau(v) = x$ and $\delta(v) \neq 0$. Hence the algorithm Sch-Searchgraph-Decoupled terminates successfully if and only if the corresponding tree T has no leaves left with a distance other than 0. Since Y' grows by exactly 1 in every iteration, the lemma follows. □

We are therefore interested in the probability $\Pr[Y' \leq t]$. Since

$$\Pr[Y' \leq t] \geq \Pr[Y \leq t] \tag{2.1}$$

we will concentrate on $\Pr[Y \leq t]$. We first determine $\Pr[Y < \infty]$, which corresponds to the success probability of Sch-Searchgraph-Decoupled for $t = \infty$. Later we prove that for some t linear in n the success probability is within a constant bound of the infinite case. This proof concept corresponds to the the idea in the original analysis to first consider the hitting probability of a Markov chain (allowing infinite time), and then bounding the number of allowed steps afterwards.

Let $v \in V_0$ be some root. It can easily be seen that the probability that the tree rooted at v is finite can only depend on $\delta(v)$. Hence using the following definition for λ_s the values λ_s are well-defined.

Definition 2.19 *Let $v \in V$ be any node with $\delta(v) = s$. Then we define*

$$\lambda_s := \Pr[Y_v < \infty]$$

Furthermore the probabilities that separate subtrees are finite are independent of each other. Hence

Lemma 2.20

$$\Pr[Y < \infty] = \prod_{v \in V_0} \Pr[Y_v < \infty]$$

Given the definition of the values λ_s and the probability distribution for the label of a root we can give the probability that the tree branching process T gives a finite tree.

Lemma 2.21

$$\Pr[Y < \infty] = \left(\sum_{s=0}^{l} \frac{|\Gamma_s| \cdot \lambda_s}{d} \right)^n$$

Proof By lemma 2.20 we have

$$\begin{aligned}
\Pr[Y < \infty] &= \prod_{v \in V_0} \Pr[Y_v < \infty] \\
&= \prod_{v \in V_0} \sum_{s=0}^{l} \Pr[\delta(v) = s] \cdot \Pr[Y_v < \infty \mid \delta(v) = s] \\
&= \prod_{v \in V_0} \sum_{s=0}^{l} \Pr[\delta(v) = s] \cdot \lambda_s \\
&= \prod_{v \in V_0} \sum_{s=0}^{l} \frac{|\Gamma_s|}{d} \cdot \lambda_s \\
&= \left(\sum_{s=0}^{l} \frac{|\Gamma_s| \cdot \lambda_s}{d} \right)^n
\end{aligned}$$

□

We now determine a system of equations for $\lambda_0, \ldots, \lambda_l$. We already have $\lambda_0 = 1$ as a node v with $\delta(v) = 0$ never has any children. For all the other values λ_s we know the probabilities of its children and their distances. This gives us the following equation.

Lemma 2.22 *For all* $1 \leq s \leq l$ *we have*

$$\lambda_s = \frac{k-1}{k} \cdot \lambda_s \lambda_1 + \frac{1}{k} \left(\sum_{s'=0}^{l} \frac{p_{s',s}}{deg(G)} \cdot \lambda_{s'} \right)$$

Proof Let v be a node with $\delta(v) = s$. By definition 2.19 we have

$$\lambda_s = \Pr[Y_v < \infty]$$

We use the law of total probability with the children of v. The vertex v either has two children (with distances s and 1) or a single child with some distance s'. We write $c(v) = \{1, s\}$ if v has two children and $c(v) = \{s'\}$ if it

has a single child with distance s'.

$$\begin{aligned}\lambda_s &= \Pr[c(v) = \{1,s\}] \cdot \Pr[Y_v < \infty \mid c(v) = \{1,s\}] \\ &\quad + \sum_{s'=0}^{l} \Pr\left[c(v) = \{s'\}\right] \cdot \Pr\left[Y_v < \infty \mid c(v) = \{s'\}\right] \\ &= \frac{k-1}{k} \cdot \Pr[Y_v < \infty \mid c(v) = \{1,s\}] \\ &\quad + \frac{1}{k}\left(\sum_{s'=0}^{l} \frac{p_{s',s}}{\deg(G)} \cdot \Pr\left[Y_v < \infty \mid c(v) = \{s'\}\right]\right) \\ &= \frac{k-1}{k} \cdot \lambda_s\lambda_1 + \frac{1}{k}\left(\sum_{s'=0}^{l} \frac{p_{s',s}}{\deg(G)} \cdot \lambda_{s'}\right) \qquad \square\end{aligned}$$

This gives us a system of equations the values λ_s satisfy. Before analysing the solutions to this system of equations however we first show that it is not necessary to set $t = \infty$. In the classical Schöning's algorithm it is sufficient to set $t = 3n$ and still have a success probability of $\left(\frac{1}{d}k/(k-1)\right)^n \cdot \Theta(1)$. In this analysis we do not get the same constant 3, but for we show that for some $t = O(n)$ we still have a success probability of $(\sum_{s=0}^{l} |\Gamma_s| \cdot \lambda_s/d)^n \cdot \Theta(1)$. We will analyse each of the n roots separately and show that for a root $v \in V_0$ we have $E[Y_v \mid Y_v < \infty] \leq c$ for some constant c. Hence $E[Y \mid Y < \infty] \leq cn$ and using Markov's inequality we can then conclude $\Pr[Y \leq 2 \cdot cn \mid Y < \infty] \geq \frac{1}{2}$. Therefore

$$\Pr[Y \leq 2 \cdot cn] \geq \Pr[Y < \infty] \cdot \frac{1}{2} \tag{2.2}$$

In order to derive an upper bound on $E[Y_v \mid Y_v < \infty]$ we explicitly assume that d and k are such that (d,k)-CSP is NP-complete, i.e. $d \geq 2$ and $k \geq 2$ and either $d > 2$ or $k > 2$. We further treat the case where $k = 2$ and G is a directed cycle of length $d \geq 3$ separately at the end. For all other cases we first bound the number $E[Y_v^{(0)} \mid Y_v < \infty]$, where $Y_v^{(0)} = Y_v - Y_v'$ is the number of leaves rooted at v, i.e. the number of nodes with distance 0.

Lemma 2.23 *Given that we do not have $k = 2$ and G as a directed cycle*

$$E[Y_v^{(0)} \mid Y_v < \infty] \leq c'$$

for some constant c' independent of n.

Proof The subtree rooted at v can only be finite if there are more leaves than nodes with two children. We show that this is unlikely if the number of leaves is very high. If $\delta(v) = 0$ (i.e. the initial assignment has already distance 0) then $Y_v^{(0)} = 1$, hence we only have to consider the case where v is not already a leaf.

We recall some facts established in the section on distance-regular graphs. We have established stability in lemma 1.18 saying that for a distance regular graph and two vertices u and w, we have

$$d_G(u,w) = 1 \Leftrightarrow d_G(w,u) = 1^*$$

where we defined the function $*$ in definition 1.22. For an undirected graph we have $1^* = 1$ and for an directed graph we have $1^* = g - 1$ where g is the girth of the graph. The parent of a leaf v' then necessarily has distance 1^*, as $p_{0,s} = 0$ for any $s \neq 1^*$ by above observations.

Consider the following three numbers

- Y_0, the number of nodes v' rooted at v with $\delta(v') = 1^*$ and a single child v'' with $\delta(v'') = 0$.
- Y_2, the number of nodes v' rooted at v with $\delta(v') = 1^*$ and exactly two children, one with $\delta(v'') = 1^*$ and one with $\delta(v'') = 1$
- $Y_0 + Y_2$

The number of parents of leaves is the same as the number of leaves, i.e.

$$Y_0 = Y_v^{(0)} \tag{2.3}$$

Hence

$$E[Y^{(0)} \mid Y_v < \infty] \leq E[Y_0 + Y_2 \mid Y_v < \infty] \tag{2.4}$$

We further have

$$E[Y_0 + Y_2 \mid Y < \infty] = \sum_{q=1}^{\infty} q \cdot \Pr[Y_0 + Y_2 = q \mid Y_v < \infty] \tag{2.5}$$

by the definition of the expectation.

For bounding $\Pr[Y_0 + Y_2 = q \mid Y_v < \infty]$ we use above observation that the number of leaves must be greater than the number of nodes with two children, as the tree cannot be finite otherwise. In particular this means $Y_0 \geq Y_2$. This give us the following relation

$$\begin{aligned}
\Pr[Y_0 + Y_2 = q \mid Y_v < \infty] &= \frac{\Pr[Y_0 + Y_2 = q \wedge Y_v < \infty]}{\Pr[Y_v < \infty]} \\
&= \frac{\Pr[Y_0 + Y_2 = q \wedge Y_v < \infty \wedge Y_0 \geq \frac{q}{2}]}{\Pr[Y_v < \infty]} \\
&\leq \frac{\Pr[Y_0 + Y_2 = q \wedge Y_0 \geq \frac{q}{2}]}{\Pr[Y_v < \infty]}
\end{aligned}$$

According to definition 2.17 the probability that a node with distance 1^* has a single child with distance 0 is $\frac{1}{k} \cdot \frac{1}{\deg(G)}$ while the probability for having two children is $\frac{k-1}{k}$. We can normalise to these two cases only, which corresponds to considering a separate source of randomness for these two cases and determining the probability that this source of randomness chooses the case with a single child of distance 0. For the first q such decisions we therefore have $Y_0 \sim \text{Bin}(q, p)$ for

$$\begin{aligned} p &= \frac{1}{k} \cdot \frac{1}{\deg(G)} \cdot \frac{1}{\frac{1}{k} \cdot \frac{1}{\deg(G)} + \frac{k-1}{k}} \\ &= \frac{1}{1 + \deg(G)(k-1)} \end{aligned}$$

Note that $p < \frac{1}{2}$ if $k > 2$ or $\deg(G) > 1$. For $d > 2$, the only distance-regular graph with $\deg(G) = 1$ is the directed cycle, which we treat later as a special case.

Since we have a binomial distribution we can apply Chernoff bounds (see the appendix) to bound the probability that the number of nodes with two children is different from the expectation. We get

$$\Pr[Y_0 \geq \frac{q}{2} \wedge Y_0 + Y_2 = q] = \Pr[Y_0 \geq (1+\delta)qp \wedge Y_0 + Y_2 = q]$$

for

$$\delta = \frac{1}{2p} - 1$$

where we have $\delta > 0$ for $p < \frac{1}{2}$. Chernoff bounds then give us

$$\begin{aligned} \Pr[Y_0 \geq (1+\delta)qp \wedge Y_0 + Y_2 = q] &\leq e^{-\frac{qp\delta^2}{3}} \\ &= e^{-\frac{qp(\frac{1}{2p}-1)^2}{2}} \\ &= e^{-c''q} \end{aligned}$$

for some constant c''. Plugging that in above we get

$$\begin{aligned} E[Y_0 + Y_2 \mid Y_v < \infty] &= \sum_{q=1}^{\infty} q \cdot \Pr[Y_0 + Y_2 = q \mid Y_v < \infty] \\ &\leq \sum_{q=1}^{\infty} q \cdot \frac{\Pr[Y_0 \geq \frac{q}{2} \wedge Y_0 + Y_2 = q]}{\Pr[Y_v < \infty]} \\ &\leq \sum_{q=1}^{\infty} q \cdot \frac{e^{-c''q}}{\Pr[Y_v < \infty]} \\ &\leq d \sum_{q=1}^{\infty} q \cdot e^{-c''q} \\ &= c' \end{aligned}$$

For the last step we use that $Pr[Y_v < \infty] \geq \frac{1}{d}$, as $\frac{1}{d}$ is the probability that v is already a leaf.

Hence the expected number of leaves in the tree rooted at v is also constant. □

We now show in the last step that the expected number of nodes c is $c = O(c') = O(1)$.

Lemma 2.24 *Given that we do not have $k = 2$ and G as a directed cycle*

$$E[Y_v \mid Y_v < \infty] \leq c$$

for some constant c independent of n.

Proof Consider the length of the path from a node v' to its corresponding leaf. We define the path to the corresponding leaf, such that if a node has two children, one with $\delta(v) = 1$ and one with the same label δ as its parent, the path continues with the one that has the same label. Considering the coupling in the proof of lemma 2.18, this would be the leaf that has the same label τ. For any node v' with $\delta(v') = s$ we know that with probability $\frac{1}{k} \cdot \frac{p_{(s-1)^*,s}}{\deg(G)}$ the node v' has a single child with distance $(s-1)^*$. Hence with probability $\prod_{i=1}^{s} \frac{1}{k} \cdot \frac{p_{(s-1)^*,s}}{\deg(G)}$ the path from v' to its corresponding leaf has length s. Since $p_{(i-1)^*,i} \geq 1$ for all $1 \leq i \leq l$ and $s \leq l$ by connectivity, we get for any node v' that the probability is at least $\left(\frac{1}{k \cdot \deg(G)}\right)^l = O(1)$ that the number of nodes on the path between v' and its leaf is at most l. We have therefore a geometric distribution and the expected number of nodes on this path is at most $l \cdot (k \cdot \deg(G))^l$. Hence we get

$$\begin{aligned} E[Y_v \mid Y_v < \infty] &\leq l \cdot (k \cdot \deg(G))^l \, E[Y_v^{(0)} \mid Y_v < \infty] \\ &= c \end{aligned}$$

□

The special case where G is a directed cycle and $k = 2$ is easier and has a very similar proof. For every inner node with two children there must be $d - 1$ inner nodes with only a single child. We have $p = \frac{1}{2}$, where p denotes the probability of a single child. However, at least $\frac{2}{3}$ of the nodes need to have a single child for the tree to be finite. A similar Chernoff bound argument then bounds the expected number of nodes rooted at some v by a constant.

As there are n independent trees, the expected size of V is therefore at most cn. We can now use Markov's inequality to to proof the following

Lemma 2.25 *For some $t = O(n)$ we have*

$$\Pr[Y \leq t] \geq \left(\sum_{s=0}^{l} \frac{|\Gamma_s| \cdot \lambda_s}{d} \right)^n \cdot \frac{1}{2}$$

Proof We set $t' = 2cn$ and using Markov's inequality we get

$$\begin{aligned}
\Pr[Y \leq t] &= \Pr[Y < \infty \wedge Y \leq t] \\
&= \Pr[Y < \infty] \cdot \Pr[Y \leq t \mid Y < \infty] \\
&\geq \Pr[Y < \infty] \cdot \Pr[Y \leq 2 \cdot E[Y \mid Y < \infty] \mid Y < \infty] \\
&\geq \Pr[Y < \infty] \cdot \frac{1}{2} \\
&= \left(\sum_{s=0}^{l} \frac{|\Gamma_s| \cdot \lambda_s}{d} \right)^n \cdot \frac{1}{2}
\end{aligned}$$

□

It is therefore sufficient to set t to something linear in n, which means that a single run of Sch-Searchgraph-Decoupled runs in time polynomial in n and m.

Note that while possible to calculate an exact number for the constant c, this approach only gives us a very rough bound. Hence it cannot be used to compare the performance of the modified algorithm to the $3n$ steps needed by the original algorithm. However, the total running time of Schöning's algorithm including repetition is only affected by a polynomial factor, as both cn and $3n$ are linear in n. The exact constant c is therefore not of particular interest in this context.

2.7 Analysis of the Equations

In the previous section we showed that the success probability is given by

$$\left(\sum_{s=0}^{l} \frac{|\Gamma_s| \cdot \lambda_s}{d} \right)^n$$

where the values λ_s are the probabilities, that the algorithm can correct a variable with distance s to the satisfying assignment while correcting all errors made in the process. Since the values λ_s are probabilities we have

$$\lambda_s \in [0, 1]$$

Furthermore we showed that the probabilities λ_s satisfy the following system of equations

$$\begin{aligned} \lambda_0 &= 1 \\ \lambda_s &= \frac{k-1}{k} \cdot \lambda_s \lambda_1 + \frac{1}{k} \left(\sum_{s'=0}^{l} \frac{p_{s',s}}{\deg(G)} \cdot \lambda_{s'} \right) \end{aligned}$$

where the second equation is true for all $1 \leq s \leq l$.

In this section we will analyse this system of equations. We will first show that every solution either supports our main theorem (theorem 2.4) or $\lambda_1 = 1$. It can easily be seen that $\lambda_s = 1$ for all s is indeed always a solution to the system of equations, however we will show $\lambda_1 < 1$.

The system of equations depends on the intersection numbers of the graph G, i.e. on the values $p_{s',s}$. Therefore the solutions for λ_s depend on the search graph. However, the success probability does not depend on the individual values as long as the sum

$$\sum_{s=0}^{l} |\Gamma_s| \cdot \lambda_s$$

is the same. This leads us to the following lemma

Lemma 2.26 *For every solution to the system of equations we either have*

$$\sum_{s=0}^{l} |\Gamma_s| \cdot \lambda_s = \frac{k}{k-1}$$

or

$$\lambda_1 = 1$$

Proof We assume $\lambda_1 \neq 1$ and define $X = \sum_{s=0}^{l} |\Gamma_s| \cdot \lambda_s$. Since $\lambda_0 = 1$ and

$|\Gamma_0| = 1$ we get

$$\begin{aligned}
X &= \sum_{s=0}^{l} |\Gamma_s| \cdot \lambda_s \\
&= 1 + \sum_{s=1}^{l} |\Gamma_s| \cdot \lambda_s \\
&= 1 + \frac{k-1}{k} \cdot \lambda_1 \cdot \sum_{s=1}^{l} |\Gamma_s| \cdot \lambda_s \\
&\quad + \frac{1}{k \cdot \deg(G)} \cdot \sum_{s=1}^{l} |\Gamma_s| \left(\sum_{s'=0}^{l} p_{s',s} \cdot \lambda_{s'} \right) \\
&= 1 + \frac{k-1}{k} \cdot \lambda_1 \cdot (X-1) \\
&\quad + \frac{1}{k \cdot \deg(G)} \cdot \sum_{s=1}^{l} |\Gamma_s| \left(\sum_{s'=0}^{l} p_{s',s} \cdot \lambda_{s'} \right)
\end{aligned}$$

We can change the order of the sums and observe that $p_{s',0} = 0$ for $s' \neq 1$. Furthermore we have $|\Gamma_0| p_{1,0} = \deg(G)$. Hence

$$\begin{aligned}
X &= 1 + \frac{k-1}{k} \cdot \lambda_1 \cdot (X-1) \\
&\quad + \frac{1}{k \cdot \deg(G)} \cdot \left(\sum_{s'=0}^{l} \lambda_{s'} \left(\sum_{s=1}^{l} |\Gamma_s| p_{s',s} \right) \right) \\
&= 1 + \frac{k-1}{k} \cdot \lambda_1 \cdot (X-1) \\
&\quad + \frac{1}{k \cdot \deg(G)} \cdot \left(\left(\sum_{s'=0}^{l} \lambda_{s'} \left(\sum_{s=0}^{l} |\Gamma_s| p_{s',s} \right) \right) - |\Gamma_0| p_{1,0} \lambda_1 \right) \\
&= 1 + \frac{k-1}{k} \cdot \lambda_1 \cdot (X-1) \\
&\quad + \frac{1}{k \cdot \deg(G)} \cdot \left(\left(\sum_{s'=0}^{l} \lambda_{s'} \left(\sum_{s=0}^{l} |\Gamma_s| p_{s',s} \right) \right) - \deg(G) \lambda_1 \right)
\end{aligned}$$

We can then use that for any $0 \leq s' \leq l$,

$$\sum_{s=0}^{l} |\Gamma_s| p_{s',s} = |\Gamma_{s'}| \deg(G)$$

which follows directly from the definition of $p_{s',s}$ and regularity. Therefore

$$\begin{aligned} X &= 1 + \frac{k-1}{k} \cdot \lambda_1 \cdot (X-1) \\ &\quad + \frac{1}{k \cdot \deg(G)} \cdot \left(\left(\sum_{s'=0}^{l} \lambda_{s'} |\Gamma_{s'}| \deg(G) \right) - \deg(G) \lambda_1 \right) \\ &= 1 + \frac{k-1}{k} \cdot \lambda_1 \cdot (X-1) + \frac{1}{k}(X - \lambda_1) \end{aligned}$$

We now simply solve this equation for X.

$$\begin{aligned} X &= 1 + \frac{k-1}{k} \cdot \lambda_1 \cdot (X-1) + \frac{1}{k}(X - \lambda_1) \\ kX &= k + (k-1) \cdot \lambda_1 \cdot (X-1) + (X - \lambda_1) \\ kX &= k + (k-1) \cdot \lambda_1 \cdot X - (k-1) \cdot \lambda_1 + X - \lambda_1 \\ kX - (k-1) \cdot \lambda_1 \cdot X - X &= k - (k-1) \cdot \lambda_1 - \lambda_1 \\ (k-1)(1-\lambda_1)X &= k(1-\lambda_1) \\ X &= \frac{k}{k-1} \end{aligned}$$

The last step is allowed only because of our assumption $\lambda_1 \neq 1$, as we would divide by 0 otherwise. Hence either $X = \frac{k}{k-1}$ or our assumption is wrong, i.e. $\lambda_1 = 1$. □

Plugging this in into the success probability we immediately get the following lemma.

Lemma 2.27 *Either*

$$\Pr[Y < \infty] = \left(\frac{1}{d} \cdot \frac{k}{k-1} \right)^n$$

or

$$\lambda_1 = 1$$

Note that by the system of equations and connectivity $\lambda_1 = 1$ implies that for all $0 \leq s \leq l$ we have $\lambda_s = 1$. Using lemma 2.21 this would imply that Sch-Searchgraph runs in polynomial time. To contradict this, we will prove an upper bound for λ_1. As we clearly have $\lambda_1 \leq \lambda_{1^*}$ we bound λ_{1^*} instead. Let $v \in V_0$ be some root with $\delta(v) = 1^*$. We analyse the probability that the the tree rooted at v is finite. Again we have to treat the case of $k = 2$ and a directed cycle separately. As in the coupling of lemma 2.18 we again consider adding the children of a single node v' with $\delta(v') = s \neq 0$ in each time step. By doing this, the number of leaves with $\delta(v') \neq 0$ rooted at v (we call them *open leaves*) changes in the following way.

- If $s = 1^*$ then the number of open leaves decreases with probability $\frac{1}{k \cdot \deg(G)}$ and increases with probability $\frac{k-1}{k}$.
- Otherwise the number of open leaves cannot decrease but increases with probability $\frac{k-1}{k}$.

As we are interested in an upper bound on the success probability only the first case is relevant. We therefore consider the following Markov chain.

Definition 2.28 *Let $\{\mathcal{W}_i\}_{i \geq 0}$ be a time-homogeneous Markov chain with state space $S = \mathbb{N}$, a deterministic starting state at 1 and transition probabilities of the form*

$$p(x, x+\delta) = \begin{cases} \frac{1}{k \cdot deg(G)} & \text{if } \delta = -1 \\ \frac{deg(G)-1}{k \cdot deg(G)} & \text{if } \delta = 0 \\ \frac{k-1}{k} & \text{if } \delta = 1 \\ 0 & \text{otherwise} \end{cases}$$

if $x > 0$ and

$$p(x, x+\delta) = \begin{cases} 1 & \text{if } \delta = 0 \\ 0 & \text{otherwise} \end{cases}$$

if $x = 0$.

The definition of the starting state at 1 captures that any process starts with exactly one leaf, namely the root v. We are interested in the probability that $\{\mathcal{W}_i\}_{i \geq 0}$ ever reaches the state 0 given that the starting state is 1. We have already analysed this kind of Markov chain in the introductory chapter. With regard to the case of the directed cycle we use the more general framework by Giurgiu [9] instead of relying on the results from earlier. We make more use of this framework in the alternative proof in the appendix, so we refer to lemma A.10 or directly to [9] for details. Lemma A.10 states that

$$\Pr[\exists i : \mathcal{W}_i = 0 \mid \mathcal{W}_0 = 1] = \lambda$$

where λ is the only solution in $(0, 1)$ of the equation

$$\lambda = GF_T(\lambda) \tag{2.6}$$

where the random variable T captures the transition probabilities with

$$\Pr[T = a] = p(x, x + (a-1))$$

for any non-negative $a \in \mathbb{N}_0$. In other words, T has the same distribution as the step size normalised such that it is non-negative. $GF_T(\lambda)$ then refers to

the generating function of T. In this case we have

$$\Pr[T = a] = \begin{cases} \frac{1}{k \cdot \deg(G)} & \text{if } a = 0 \\ \frac{\deg(G)-1}{k \cdot \deg(G)} & \text{if } a = 1 \\ \frac{k-1}{k} & \text{if } a = 2 \\ 0 & \text{otherwise} \end{cases}$$

Note that a condition of lemma A.10 is $E[T] > 1$ which is given except for the special case $k = 2$ and G being a directed cycle. We treat this case below. Using the definition of T and of the generating function (see the appendix) we get

$$GF_T(\lambda) = \frac{1}{k \cdot \deg(G)} + \frac{\deg(G) - 1}{k \cdot \deg(G)} \cdot \lambda + \frac{k-1}{k} \cdot \lambda^2$$

By solving equation 2.6 for λ we then get

$$\lambda = \frac{1}{\deg(G) \cdot (k-1)}$$

We therefore have the following lemma.

Lemma 2.29

$$\lambda_1 \leq \lambda_{1^*} \leq \frac{1}{deg(G) \cdot (k-1)} < 1$$

The success probability is smaller than 1 if $k > 2$ or $d > 2$, as we assume a connected graph G and explicitly do not allow a directed cycle while $k = 2$.

For the directed cycle with $d \geq 3$ and $k = 2$ we consider the accumulated distance to the origin. A leaf with distance $s \neq 0$ has to be flipped exactly $d - s$ times in order to become a leaf with distance 0. If a node has one child the accumulated distance decreases by one, while a node with two children increases the accumulated distance by $d - 1$. We therefore consider the following Markov chain.

Definition 2.30 *Let $\{\mathcal{W}_i\}_{i \geq 0}$ be a time-homogeneous Markov chain with state space $S = \mathbb{N}$, a deterministic starting state at 1 and transition probabilities of the form*

$$p(x, x+\delta) = \begin{cases} \frac{1}{2} & \text{if } \delta = -1 \\ \frac{1}{2} & \text{if } \delta = d-1 \\ 0 & \text{otherwise} \end{cases}$$

if $x > 0$ and

$$p(x, x+\delta) = \begin{cases} 1 & \text{if } \delta = 0 \\ 0 & \text{otherwise} \end{cases}$$

if $x = 0$.

As above we have λ_{1^*} as the only λ in $(0,1)$ satisfying

$$\lambda = GF_T(\lambda)$$

where

$$GF_T(\lambda) = \frac{1}{2} + \frac{1}{2} \cdot \lambda^d$$

The solution for this equation is maximised for $d = 3$, hence we only have to consider the solution in $(0,1)$ of the equation

$$\lambda = \frac{1}{2} + \frac{1}{2} \cdot \lambda^3$$

which is $\frac{\sqrt{5}-1}{2} < 1$.

Proposition 2.29 (and the specially treated case of the directed cycle) eliminates the possibility of $\lambda_1 = 1$ for lemma 2.27 . We have therefore determined the probability that the tree T is finite. We also showed with lemma 2.25 that this probability is within a constant factor of the success probability of Sch-Searchgraph-Decoupled for some linear t. By our coupling any lower bound on Sch-Searchgraph-Decoupled is also a lower bound for the algorithm Sch-Searchgraph. Hence the main theorem 2.4 follows.

2.8 Worst-Case Analysis

It this section we show that the analysis of the previous sections is indeed tight in a very strong sense.

Theorem 2.31 *There is family of formulas, such that if the number of variables n is large enough and the maximum number of iterations t polynomial then for any selection rule and any distance-regular search graph G the success probability of Sch-Searchgraph is at most*

$$\left(\frac{1}{d} \cdot \frac{k}{k-1}\right)^n \cdot poly(n)$$

An immediate consequence is the impossibility of a universal "angel", a selection rule that increases the success probability of Sch-Searchgraph for any input formula by more than a polynomial factor.

The proof is based on [10], where we proved the impossibility of a universal angel in the context of boolean satisfiability.

Let n be large and define $\mathrm{vbl}(F_{\mathrm{ang}}) := \{x_1, \ldots, x_n\}$. Further define α^* as the assignment $1^{\mathrm{vbl}(F_{\mathrm{ang}})}$. The formula F_{ang} contains the following clauses.

- All clauses C with $\mathrm{vbl}(C) = \{x_1, \ldots, x_k\}$ satisfied by α^*
- For any $j > k$ all clauses C of the form

$$\{(x_{j-k+1}, 1), (x_{j-k+2}, 1), \ldots, (x_{j-1}, 1), (x_j, a)\}$$

  for any $a \neq 1$

It can easily be seen that α^* is the only satisfying assignment of F_{ang}.

We will now define a labelled tree branching process T' such that the probability that the tree has a size smaller than t is an upper bound on the success probability of Sch-Searchgraph using the formula F_{ang} and any selection rule. The tree branching process T' will resemble the process T used in the previous sections. This will then allow us to use the tools established there. For this definition we use two different labels $\delta : V \to \{1, \ldots, l\}$ and $\tau : V \to \mathrm{vbl}(F_{\mathrm{ang}})$. We refer to δ as the distance, and τ as the variable.

Definition 2.32 *Let T' be a labelled tree branching process with labels $\delta : V \to \{1, \ldots, l\}$ and $\tau : V \to vbl(F_{ang})$. For the root set V_0 we have $|V_0| = n$ such that for each $x \in vbl(F_{ang})$ there is exactly one node $v \in V_0$ with $\tau(v) = x$. Furthermore for each $v \in V_0$ we have*

$$\Pr[\delta(v) = s] = \frac{|\Gamma_s|}{d}$$

for any $0 \leq s \leq l$ independent of each other.

Further any node v with $s = \delta(v) \neq 0$ and $\tau(v) = x_j$ with $j \geq k + 1$ has

- *One child with distance 1 and one child with distance s with probability $\frac{k-1}{k}$, such that the the child with distance s has variable x_j and the child with distance 1 has a random variable in $\{x_{j-k+1}, \ldots, x_{j-1}\}$*
- *One child with distance s' and variable x_j with probability $\frac{1}{k} \cdot \frac{p_{s',s}}{deg(G)}$ for any $0 \leq s' \leq l$*

and any node v with $\delta(v) = 0$ or $\tau(v) \in \{x_1, \ldots, x_k\}$ has zero children (i.e. is a leaf).

This definition is very similar to the definition of T in definition 2.17, except that there is an additional label τ. It can easily be seen that in the case $\tau(v) = x_j$ for some $j \geq k + 1$, the probability distribution on how many children v has and what labels δ they have are indeed the same as in T. The label τ of those children is $\tau(v)$ if it is a single child and if there are two children the one with the same δ as v also has the same τ, while the other one has a random label in $\{x_{j-k+1}, \ldots, x_{j-1}\}$. If however $\tau(v) = x_j$ for some $j \leq k$, then v is always a leaf. We call such a node a *forced leaf*. Note that the label δ is the same as in the definition of T, while the label τ maps every node

to a variable as the label τ used in the coupling of lemma 2.18. We formally define this relationship between T' and the algorithm Sch-Searchgraph with the formula F_{ang} in the following lemma. For $0 \leq j \leq n$ let $v_j \in V_0$ be the root with $\tau(v_j) = x_j$. Like in the previous section we use Y for the number of nodes in whole forest and Y' for the number of non-leaves in the whole forest. The notion Y_{v_j} and Y'_{v_j} denotes the corresponding number of nodes rooted at some root $v_j \in V_0$.

Lemma 2.33 *For any selection rule σ, the probability that Sch-Searchgraph(F_{ang},t,G,σ) terminates successfully is at most* $\Pr[Y' \leq t]$.

Proof We use a coupling argument. In every iteration of Sch-Searchgraph at most one node's children are added. We prove that in this process any variable $x \in \{x_{k+1}, x_n\}$ with $\alpha(x) \neq \alpha^*(x)$ corresponds to exactly one leaf $v \in V$ with $\tau(v) = x$ and $\delta(v) = d_G(\alpha^*(x), \alpha(x))$. Consider the initial assignment α_0. For every variable $x_j \in \text{vbl}(F_{\text{ang}})$ there is a root $v_j \in V_0$ with $\tau(v_j) = x_j$ and $\delta(v) = d_G(\alpha^*(x), \alpha_0(x))$. The probability distribution of the label δ corresponds exactly to the distribution specified in definition 2.32. Whenever the selection rule σ selects a clause C of the form

$$\{(x_{j-k+1}, 1), (x_{j-k+2}, 1), \ldots, (x_{j-1}, 1), (x_j, a)\}$$

for $j \geq k+1$ there is a leaf v with $\tau(v) = x_j$ and $\delta(v) = d_G(\alpha^*(x_j), a)$. If we flip the variable x_j in the algorithm Sch-Searchgraph, we attach a single node v' to v with $\tau(v') = \tau(v)$ and a label δ corresponding to the new distance of x_j. If however we flip a variable $x_{j'}$ for some $j' \neq j$, we instead attach two nodes to v, a variable v' with $\tau(v') = x_j$ and $\delta(v') = \delta(v)$, and one node v' with $\tau(v') = x_{j'}$ and $\delta(v') = 1$. It can easily be seen that the distribution of this process is the same as in definition 2.32 if $\tau(v) \in \{x_{k+1}, \ldots, x_n\}$. The fact that any node v with $\tau(v) \in \{x_1, \ldots, x_k\}$ is automatically a leaf can only increase the probability that the tree is smaller than t. Furthermore Y' is always upper bounded by the number of iterations. □

As noted earlier, the tree branching process T' behaves exactly like the tree branching process T in definition 2.17 except for the forced leaves. We define ω_j as the number of forced leaves v rooted at v_j, i.e. the number of nodes with variable x_j for $j \leq k$.

Using this notation and above observations about the relation of T' to the tree branching process T in the previous section we get the following lemma.

Lemma 2.34 *The probability that the size of the tree rooted at v_j is at most t and ω_j is 0 is at most*

$$\Pr[Y_{v_j} \leq t \wedge \omega_j = 0] \leq \frac{1}{d} \cdot \frac{k}{k-1}$$

Note that $\frac{1}{d} \cdot \frac{k}{k-1}$ is the corresponding probability we derived in the previous section.

We now consider the case $\omega_j > 0$, i.e. there is at least one forced leaf. We want to argue that if there is such a forced leaf, then there is also a fairly large set of nodes V' rooted at v_j such that no $v' \in V'$ is a root, there is no forced leaf in the subtree of v' and no to $v', v'' \in V'$ lie on a common path from the root v_j to some leaf. In other words, each $v' \in V'$ is the root of an independent subtree. We first define the *chronological traversal* of a subtree of T', as we will need this definition for the proof later.

Definition 2.35 *The* chronological traversal *of a subtree rooted at a node v is a preorder traversal such that the node v is first traversed. If there is a single child v' of v, then the subtree rooted at v' is traversed chronologically. If there are two children v' and v'' such that $\tau(v') \neq \tau(v)$ and $\tau(v'') = \tau(v)$ then the subtree rooted at v' is traversed first.*

In the standard definition of a preorder traversal where the left child is traversed before the right child, the child which has a different label τ (and always $\delta(v') = 1$) would be the left child. Note that this traversal corresponds to the ordering in which the tree is built according to the coupling of lemma 2.33, as the selection rule σ cannot choose a clause

$$\{(x_{j-k+1}, 1), (x_{j-k+2}, 1), \ldots, (x_{j-1}, 1), (x_j, a)\}$$

if $\alpha(x_{j'}) \neq \alpha^*(x_{j'})$ for some $j - k + 1 \leq j' \leq j - 1$.

We are now ready for proving a lower bound on the size of such a set.

Lemma 2.36 *Given $\omega_j > 0$ the tree rooted at v_j has a set V' with*

$$|V'| \geq \left\lfloor \frac{j-k}{k-1} \right\rfloor$$

such that for every $v' \in V'$ there is no forced leaf rooted at v' and no two nodes in v' are on a common path from the root to a leaf.

Proof Consider the last forced leaf according to the chronological ordering. Let this leaf be v and consider the path from v to v_j. For every node v' with $\tau(v') = x_{j'}$ there is parent $\gamma(v')$ with $\tau(\gamma(v')) \in \{x_{j'}, \ldots, x_{j'+k-1}\}$ as can easily be verified by the definition of T'. Hence there are at least $\lfloor \frac{j-k}{k-1} \rfloor$ nodes v' on the path, such that its parent has a different label τ. Hence the parent must have two children, one of which is v'. Let the other child be v''. There is no forced leaf rooted at v'', as v would not be the last forced leaf otherwise. Hence there are at least $\lfloor \frac{j-k}{k-1} \rfloor$ of such nodes v''. □

Given that we have such a set V' the probability that each of its subtrees are finite is exactly the same is if we have V' as the root set. Hence the

probability that a subtree rooted at v with $\delta(v) = s$ is both finite and does not contain a forced leaf is upper bounded by $\lambda_s \leq \lambda_{1^*}$. This leads to the following lemma.

Lemma 2.37 *For any $k < b \leq t$,*

$$\Pr[Y_{v_j} \leq t \wedge \omega_j = b] \leq (\lambda_{1^*})^{\lfloor \frac{j-k}{k-1} \rfloor}$$

Note that for $b \leq k$ the probability that the tree is finite is 1, as the root is a forced leaf.

As ω_j is at most t we get for $j \geq k+1$

$$\begin{aligned}
\Pr[Y_{v_j} \leq t] &= \sum_{b=0}^{t} \Pr[Y_{v_j} \leq t \wedge \omega_j = b] \\
&\leq \frac{1}{d} \cdot \frac{k}{k-1} + \sum_{b=1}^{t} \Pr[Y_{v_j} \leq t \wedge \omega_j = b] \\
&\leq \frac{1}{d} \cdot \frac{k}{k-1} + t \cdot (\lambda_{1^*})^{\lfloor \frac{j-k}{k-1} \rfloor} \\
&\leq \frac{1}{d} \cdot \frac{k}{k-1} + t \cdot (\lambda_{1^*})^{\lfloor \frac{j-k}{k} \rfloor} \\
&= \frac{1}{d} \cdot \frac{k}{k-1} + t \cdot (\lambda_{1^*})^{\lfloor \frac{j}{k} - 1 \rfloor} \\
&\leq \frac{1}{d} \cdot \frac{k}{k-1} + t \cdot (\lambda_{1^*})^{\frac{j}{k} - 2} \\
&\leq \frac{1}{d} \cdot \frac{k}{k-1} + t \cdot (\lambda_{1^*})^{-2} \cdot (\lambda_{1^*})^{\frac{j}{k}}
\end{aligned}$$

Note that by lemma 2.29, $\lambda_{1^*} < 1$ for all graphs G and any NP-complete case of (d,k)-CSP.

By the independence of the subtrees we get

$$\begin{aligned}
\Pr[Y \leq t] &\leq \prod_{j=k+1}^{n} \Pr[Y_{v_j} \leq t] \\
&\leq \prod_{j=k+1}^{n} \left(\frac{1}{d} \cdot \frac{k}{k-1} + t \cdot (\lambda_{1^*})^{-2} \cdot (\lambda_{1^*})^{\frac{j}{k}} \right)
\end{aligned}$$

Let us define

$$A = -\frac{k}{\ln(\lambda_{1^*})} \cdot \ln(n) \tag{2.7}$$

Note since $\lambda_{1^*} < 1$ we have $A \geq k+1$ for n large enough. The definition of A is chosen such that

$$\begin{aligned}(\lambda_{1^*})^{\frac{A}{k}} &= (\lambda_{1^*})^{-\frac{\ln(n)}{\ln(\lambda_{1^*})}} \\ &= \frac{1}{n}\end{aligned}$$

Using this definition we get

$$\begin{aligned}\Pr[Y \leq t] &\leq \prod_{j=k+1}^{n} \left(\frac{1}{d} \cdot \frac{k}{k-1} + t \cdot (\lambda_{1^*})^{-2} \cdot (\lambda_{1^*})^{\frac{j}{k}}\right) \\ &\leq \prod_{j=\lfloor A \rfloor}^{n} \left(\frac{1}{d} \cdot \frac{k}{k-1} + t \cdot (\lambda_{1^*})^{-2} \cdot (\lambda_{1^*})^{\frac{j}{k}}\right) \\ &\leq \left(\frac{1}{d} \cdot \frac{k}{k-1} + t \cdot (\lambda_{1^*})^{-2} \cdot (\lambda_{1^*})^{\frac{A}{k}}\right)^{n-\lfloor A \rfloor} \\ &\leq \left(\frac{1}{d} \cdot \frac{k}{k-1} + t \cdot (\lambda_{1^*})^{-2} \cdot (\lambda_{1^*})^{\frac{A}{k}}\right)^{n-A} \\ &\leq \left(\frac{1}{d} \cdot \frac{k}{k-1} + t \cdot (\lambda_{1^*})^{-2} \cdot \frac{1}{n}\right)^{n-A} \\ &\leq \left(\frac{1}{d} \cdot \frac{k}{k-1} \cdot \left(1 + t \cdot (\lambda_{1^*})^{-2} \cdot d\frac{k-1}{k} \cdot \frac{1}{n}\right)\right)^{n-A} \\ &\leq \left(\frac{1}{d} \cdot \frac{k}{k-1}\right)^{n-A} \cdot \left(1 + t \cdot (\lambda_{1^*})^{-2} \cdot d\frac{k-1}{k} \cdot \frac{1}{n}\right)^{n-A}\end{aligned}$$

As $A = O(\ln n)$ a factor c^{-A} is polynomial for any constant c and we get

$$\begin{aligned}\Pr[Y \leq t] &\leq \left(\frac{1}{d} \cdot \frac{k}{k-1}\right)^{n} \cdot \left(1 + t \cdot (\lambda_{1^*})^{-2} \cdot d\frac{k-1}{k} \cdot \frac{1}{n}\right)^{n} \cdot \mathrm{poly}(n) \\ &\leq \left(\frac{1}{d} \cdot \frac{k}{k-1}\right)^{n} \cdot e^{t \cdot (\lambda_{1^*})^{-2} \cdot d\frac{k-1}{k}} \cdot \mathrm{poly}(n) \\ &\leq \left(\frac{1}{d} \cdot \frac{k}{k-1}\right)^{n} \cdot \mathrm{poly}(n)\end{aligned}$$

This directly gives us theorem 2.31.

Chapter 3

Conclusions and Open Problems

3.1 Summary and Conclusions

In the introductory chapter we have introduced Schöning's algorithm for constraint satisfaction problems and its analysis. We have also detailed the coupling argument that reduces the algorithm to a random walk on the integers.

In the following chapter we then considered a modification to Schöning's algorithm due to Scheder [24]. Scheder improves upon the derandomization of Schöning's algorithm by Dantsin et al. [3] by defining a graph structure on the set of d colours. We considered the equivalent modification for the randomized algorithm by Schöning and showed that for any graph satisfying the symmetry property of distance-regularity we get the same running time up to polynomial factors.

Instead of a random walk as in the introductory chapter we coupled the algorithm to a tree branching process with Markovian properties that allowed us to analyse every variable independently of each other. We then showed both an upper bound on the running time of Schöning's algorithm and gave a formula, such that we reach the bound up to a polynomial factor independently of the selection rule.

Both a modified (distance-regular) search graph and a modified selection rule therefore do not improve on the original algorithm by Schöning. The surprising part is that for a large set of search graphs the running time is the same up to polynomial factors. We can not provide an intuitive explanation why this is the case, it does however indicate some barriers when trying to improve Schöning's algorithm and give some indication on why there are only very few known direct improvements on Schöning's algorithm. All

such improvements either address the distribution of the initial assignment or combine Schöning's algorithm with other algorithms such as PPSZ.

3.2 Asymmetric Graphs and Open Problems

All results in this thesis only apply when the search graph is distance-regular. A set of interesting open questions arise from graphs that are not distance-regular. For instance, could we take a star (i.e. one vertex in the centre and all other vertices only connected to this centre) and receive a better upper bound on the running time?

For distance-regular graphs we could assume any mapping from the colour list to the vertex set of the graphs, as symmetry dictated the same success probability. We then simply assumed some trivial mapping. For asymmetric graphs we have to refine this assumption. For instance, in the case of the star it can easily be shown that if there is only a single satisfying assignment and for each variable the value corresponds to a vertex not in the centre of the graph, then the algorithm performs a lot worse than with a distance-regular graph. Instead, one could analyse an algorithm, where the mapping is chosen randomly and independent for each variable. The algorithm would then look like Algorithm 8, where $\mathcal{M}_d$ denotes the set of possible mappings.

Algorithm 8 Sch-Searchgraph (F,t,G,σ_F)

$m \in_{\text{u.a.r.}} \mathcal{M}_d^{\text{vbl}(F)}$
$\alpha \in_{\text{u.a.r.}} \{1,\dots,d\}^{\text{vbl}(F)}$
if α satisfies F **then**
  return (true, α)
end if
for $i = 1$ **to** t **do**
  $C \leftarrow \sigma_F(m, \mathfrak{a}, \mathfrak{r})$
  $x \in_{\text{u.a.r}} \text{vbl}(C)$
  $a_{\text{new}} \in_{\text{u.a.r}} \Gamma_G(\alpha(x))$
  $\alpha(x) \leftarrow a_{\text{new}}$
  if α satisfies F **then**
    return (true, α)
  end if
end for
return (false, nil)

Note that in order to keep the selection rule as general as possible, we assume that it can use information about the mappings of the variables.

The question of how to analyse this algorithm is closely related to analysing other forms of heterogeneous formulas and algorithms, for instance Schöning's algorithm when applied to formulas with different clause lengths or variables with different colour sets. Note that asymmetric graphs are not an interesting case for the derandomization of Schöning's algorithm, as we would have to choose a deterministic colour mapping.

Another interesting question arises from hypercube graphs. Hypercubes are distance-regular and our modified algorithm using a hypercube can easily be represented as a run of the original algorithm by Schöning for boolean satisfiability for a particular formula. This formula then has a particular form where groups of variables always appear in the same clauses. When using the classical upper bound on the running time for this formula, we get a worse running time than what the upper bound obtained in this thesis for hypercubes. In other words, the set of such formulas is an easy case for Schöning's algorithm. It is however not clear if it is still an easy case when the formulas are modified slightly.

More interesting questions can arise from considering our modification in combination with other ideas to improve Schöning's algorithm. While (distance-regular) search graphs alone do not improve on the algorithm, the fact that it indeed does improve on its derandomization suggests that it might also make a difference on other variants of Schöning's algorithm.

Appendix A

Alternative Proof

In this section we give an alternative proof for the special case where $d = 4$ and G is an (undirected) two-dimensional hypercube, i.e. a 4-cycle. The coupling argument is the same as in the general case, but we determine the success probability of Sch-Searchgraph-Decoupled (Algorithm 7) with a weak selection rule differently. Instead of using a labelled tree branching process, we use a Markov chain.

As we have shown for the general case, the selection rule does not affect the probability that β reaches β^*. We can therefore fix the selection rule to something specific that is easier to analyse. In this section we consider the values $X = |S_1(\beta)|$ and $Y = |S_2(\beta)|$ (i.e. number of variables with distance 1 and 2 respectively) for our analysis. Our selection rule works as follows: If $Y > 0$ we choose $\xi' \in S_2(\beta)$. If $Y = 0$ we choose $\xi' \in S_1(\beta)$.

The transition probabilities of X and Y only depend on ξ', it particular they only depend on if $\xi' \in S_1(\beta)$ or $\xi' \in S_2(\beta)$. If $\xi' \in S_1(\beta)$, then with probability $\frac{k-1}{k}$, X increases by 1 and Y stays the same. With probability $\frac{1}{2k}$ X decreases by 1, and with probability $\frac{1}{2k}$ X decreases by 1 while Y increases by 1. If $\xi' \in S_2(\beta)$ we increase X with probability $\frac{k-1}{k}$ and with probability $\frac{1}{k}$ we decrease Y and increase X.

With this selection rule, the transition probabilities of X and Y only depend on their current values. Hence we are dealing with a Markov chain now (see definition 1.1). We first define our Markov chain given by Sch-Searchgraph-Decoupled and above selection rule in our notation for Markov chains.

Definition A.1 *Let $\{\mathcal{W}_i^{(n)}\}_{i\geq 0}$ be a time-homogeneous Markov chain with state*

space $S = \mathbb{N} \times \mathbb{N}$, *a multinomial starting distribution of the form*

$$\Pr[\mathcal{W}_0^{(n)} = (x_0, y_0)] = \binom{n}{x_0, y_0} \left(\frac{1}{2}\right)^{x_0} \left(\frac{1}{4}\right)^{y_0} \left(\frac{1}{4}\right)^{n-x_0-y_0} \tag{A.1}$$

for any $x_0, y_0 \in S$ *and transition probabilities of the form*

$$p((x,y),(x+\delta_x, y+\delta_y)) = \begin{cases} \frac{1}{k} & \text{if } \delta_x = 1 \text{ and } \delta_y = -1 \\ \frac{k-1}{k} & \text{if } \delta_x = 1 \text{ and } \delta_y = 0 \\ 0 & \text{otherwise} \end{cases} \tag{A.2}$$

if $y > 0$,

$$p((x,y),(x+\delta_x, y+\delta_y)) = \begin{cases} \frac{1}{2k} & \text{if } \delta_x = -1 \text{ and } \delta_y = 0 \\ \frac{1}{2k} & \text{if } \delta_x = -1 \text{ and } \delta_y = 1 \\ \frac{k-1}{k} & \text{if } \delta_x = 1 \text{ and } \delta_y = 0 \\ 0 & \text{otherwise} \end{cases} \tag{A.3}$$

if $y = 0$ *and* $x > 0$ *and*

$$p((x,y),(x+\delta_x, y+\delta_y)) = \begin{cases} 1 & \text{if } \delta_x = 0 \text{ and } \delta_y = 0 \\ 0 & \text{otherwise} \end{cases} \tag{A.4}$$

if $x = y = 0$.

We will sometimes use x and y to refer to some state $\mathcal{W}_i^{(n)} = (x, y)$ for some i or x_i and y_i to refer to some state $\mathcal{W}_i^{(n)} = (x_i, y_i)$ for some specific time i. Also, we will use $\mathcal{W}_i$ instead of $\mathcal{W}_i^{(n)}$ in unambiguous context.

It can easily be checked that this Markov chain indeed behaves like Sch-Searchgraph-Decoupled given above selection rule and $t = \infty$. Hence the success probability of the algorithm with $t = \infty$ is lower bounded by the probability that the Markov chain $\{\mathcal{W}_i^{(n)}\}_{i \geq 0}$ reaches the state $(0, 0)$. The state (x, y) of the Markov chain corresponds to the state $X = x$ and $Y = y$ of the algorithm. The transition probabilities of A.2 correspond to the transitions if $\xi' \in S_2(\beta)$, while A.3 corresponds to $\xi' \in S_1(\beta)$. While the algorithm terminates if the state $X = 0$ and $Y = 0$ is reached, the Markov chain instead stays in $(0, 0)$ indefinitely (equation A.4).

The sequence $\{\mathcal{W}_i\}_{i \geq 0}$ has a characteristic behaviour. It starts in some starting state $\mathcal{W}_0 = (x, y)$. As long as $y > 0$, the transition probabilities are given by A.2 and x can never decrease. In fact, x increases in every step. Once $y = 0$, the transition probabilities change to A.3, so y might increase to 1 again in the next step. If this is the case, the transition probabilities of A.2 describe the behaviour of the Markov chain again until $y = 0$. We therefore can allow us to take a limited view on this process. We are interested in the

distribution of x the first time that $y = 0$. Once we have reached $y = 0$, we are interested in how x has changed the next time we again have $y = 0$. Let us first analyse the subproblem of what happens if $y > 0$. In particular we want to know how much x grows until y is decreased by 1.

Definition A.2 *Let $\mathcal{W}_i = (x_i, y_i)$ with $y_i > 0$. Further let $i' > i$ be the smallest number, such that $\mathcal{W}_{i'} = (x_{i'}, y_{i'})$ with $y_{i'} = y_i - 1$. Now B denotes the random variable $x_{i'} - x_i$.*

Note that B is independent of i and n.

It will prove useful later to describe B as its generating function. For a small introduction to the concept of generating functions see appendix B.

Lemma A.3 *The generating function of B is*

$$GF_B(\lambda) = \frac{\lambda}{k - (k-1)\lambda} \tag{A.5}$$

Proof As we have seen earlier, x increases in every step with probability 1, and y decreases with probability $\frac{1}{k}$ and stays constant otherwise. Hence we have $B \sim \text{Geom}(\frac{1}{k})$. The generating function of a geometrically distributed random variable X with success probability p is given by

$$\begin{aligned} GF_X(\lambda) &= \sum_{i=1}^{\infty} \Pr[X = i]\lambda^i \\ &= \sum_{i=1}^{\infty} p(1-p)^{i-1}\lambda^i \\ &= \frac{p\lambda}{1 - (1-p)\lambda} \end{aligned}$$

Hence since $p = \frac{1}{k}$ for B, we have

$$\begin{aligned} GF_B(\lambda) &= \frac{\frac{1}{k}\lambda}{1 - \frac{k-1}{k}\lambda} \\ &= \frac{\lambda}{k - (k-1)\lambda} \end{aligned}$$

□

We will now try to describe the distribution of x, the first time that $y = 0$.

Definition A.4 *Let $i \in \mathbb{N}$ be the smallest time such that $y_i = 0$. Then let ${\mathcal{W}'}_0^{(n)}$ be a random variable such that*

$$\Pr[{\mathcal{W}'}_0^{(n)} = a] = \Pr[x_i = a] \tag{A.6}$$

We will later refer to it as ${\mathcal{W}'}_0$ if the context is unambiguous.

Lemma A.5 *The generating function of* $\mathcal{W}'^{(n)}_0$ *is given by*

$$GF_{\mathcal{W}'^{(n)}_0}(\lambda) = \left(\frac{1 + 2\lambda + GF_B(\lambda)}{4}\right)^n \tag{A.7}$$

Proof We first define for all $0 < i \leq n$ the random variable X_i with a probability distribution given by

$$\Pr[X_i = a] = \begin{cases} \frac{1}{4} & \text{if } a = 0 \\ \frac{1}{2} + \frac{1}{4}\Pr[B = 1] & \text{if } a = 1 \\ \frac{1}{4}\Pr[B = a] & \text{otherwise} \end{cases} \tag{A.8}$$

It can easily by seen that

$$\begin{aligned} GF_{X_i}(\lambda) &= \frac{1}{4} + \frac{1}{2}\lambda + \frac{1}{4}GF_B(\lambda) \\ &= \frac{1 + 2\lambda + GF_B(\lambda)}{4} \end{aligned}$$

We now show that the two random variables $\mathcal{W}'^{(n)}_0$ and $\sum_{i=1}^n X_i$ have the same distribution i.e. for any a

$$\Pr[\mathcal{W}'^{(n)}_0 = a] = \Pr[\sum_{i=1}^n X_i = a] \tag{A.9}$$

The starting state $\mathcal{W}^{(n)}_0$ is given by a multinomial distribution and is therefore distributed like a sum of independent, identically distributed random variables K_i i.e. for any a

$$\Pr[\mathcal{W}^{(n)}_0 = a] = \Pr[\sum_{i=1}^n K_i = a]$$

with

$$\begin{aligned} \Pr[K_i = (0,0)] &= \frac{1}{4} \\ \Pr[K_i = (1,0)] &= \frac{1}{2} \\ \Pr[K_i = (0,1)] &= \frac{1}{4} \end{aligned}$$

In this notation the sum is to be understood element-wise. We can now do an induction proof over n for A.9. The base case is trivial as both sides are 0 with probability one. For the induction step we can split the starting distribution as $\mathcal{W}^{(n)}_0 = \mathcal{W}^{(n-1)}_0 + K_n$. If $K_n = (0,0)$ we have $\mathcal{W}^{(n)}_0 = \mathcal{W}^{(n-1)}_0$.

Since the transition probabilities do not depend on n we can conclude ${\mathcal{W}'}_0^{(n)} = {\mathcal{W}'}_0^{(n-1)} + 0$. If $K_n = (1,0)$ we can use the fact that the transition probabilities do not depend on x, and hence we get ${\mathcal{W}'}_0^{(n)} = {\mathcal{W}'}_0^{(n-1)} + 1$. If $K_n = (0,1)$, we have ${\mathcal{W}'}_0^{(n-1)}$ as the distribution of x the first time that $y = 1$. By the definition of B we therefore have ${\mathcal{W}'}_0^{(n)} = {\mathcal{W}'}_0^{(n-1)} + B$. By the induction hypothesis and the definitions of K_i and X_i we can now directly conclude A.9.

The generating function of a sum of independent variables is the product of its generating functions (see the appendix on generating functions for details). Therefore we have $GF_{{\mathcal{W}'}_0^{(n)}}(\lambda) = (GF_{X_1}(\lambda))^n$ which concludes the proof. □

By a very similar proof we can get the generating function for ${\mathcal{W}'}_0^{(n)}$ conditioned on $\mathcal{W}_0^{(n)} = (x_0, y_0)$. We can split

$$\mathcal{W}_0^{(n)} = x_0 \cdot (1,0) + y_0 \cdot (0,1) + (n - x_0 - y_0) \cdot (0,0)$$

and get

$${\mathcal{W}'}_0^{(n)} = x_0 + \sum_{i=1}^{y_0} B_i$$

and we eventually reach the following lemma.

Lemma A.6 *Conditioned on $\mathcal{W}_0^{(n)} = (x_0, y_0)$ the generating function of ${\mathcal{W}'}_0^{(n)}$ is given by*

$$GF_{{\mathcal{W}'}_0^{(n)}}(\lambda) = (\lambda)^{x_0} \, (GF_B(\lambda))^{y_0}$$

We now consider the second part of our process. When $y = 0$ and $x \neq 0$, then the transition probabilities are different. For the analysis we consider any path that increases y first and then decreases y again as a single step. Formally we define a random variable T, which is 0 if we decrease x, 1 if x stays the same etc.

Definition A.7 *Given i such that $y_i = 0$ and $x_i > 0$, let j be the smallest time such that $j > i$ and $y_j = 0$. Then let T be a random variable such that*

$$\Pr[T = s] = \Pr[x_j = x_i - 1 + s] \tag{A.10}$$

In other words, T is the offset of x_i between moments where $y = 0$ minus 1. Note that the probability of y never reaching 0 again is 0. Hence the sum of the probabilities of all possible outcomes of T is 1 and T is therefore well-defined. T is independent of i since our Markov chain is

time-homogeneous. It can easily be seen from the transition probabilities A.2 and A.3 that the distribution of T is given by

$$\Pr[T = a] = \begin{cases} \frac{1}{2k} & \text{if } a = 0 \\ \frac{k-1}{k} + \frac{1}{2k}\Pr[B = 2] & \text{if } a = 2 \\ \frac{1}{2k}\Pr[B = a] & \text{otherwise} \end{cases} \tag{A.11}$$

and therefore its generating function is

$$GF_T(\lambda) = \frac{1}{2k} + \frac{k-1}{k}\lambda^2 + \frac{1}{2k}GF_B(\lambda) \tag{A.12}$$

We will now define the auxiliary Markov chain $\{\mathcal{W}'_i\}_{i\geq 0}$ with state space the set of nonnegative integers which we will then couple to our original Markov chain $\{\mathcal{W}_i\}_{i\geq 0}$.

Definition A.8 *Let T be given by definition A.7 and the starting distribution $\mathcal{W}'^{(n)}_0$ given by definition A.4, $\{\mathcal{W}'^{(n)}_i\}_{i\geq 0}$ be a time-homogeneous Markov chain whose state space is the set of nonnegative integers and whose transition probabilities are given as follows:*

$$p(j, j+\delta) = \begin{cases} \Pr[T = \delta + 1] & \text{if } \delta \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{A.13}$$

if $j > 0$ and

$$p(0, \delta) = \begin{cases} 1 & \text{if } \delta = 0 \\ 0 & \text{otherwise} \end{cases} \tag{A.14}$$

otherwise.

The two Markov chains $\{\mathcal{W}'_i\}_{i\geq 0}$ and $\{\mathcal{W}_i\}_{i\geq 0}$ are coupled in the following way: For an outcome of $\{\mathcal{W}_i\}_{i\geq 0}$, the corresponding outcome of $\{\mathcal{W}'_i\}_{i\geq 0}$ is the sequence of all x, where $\mathcal{W}_i = (x, 0)$. It can easily be checked that both T and $\mathcal{W}'_0$ are designed in a way that the probabilities of the corresponding outcomes are the same. In particular, this coupling gives us the following lemma.

Lemma A.9 *The two Markov chains $\{\mathcal{W}'_i\}_{i\geq 0}$ and $\{\mathcal{W}_i\}_{i\geq 0}$ have the same success probability.*

$$\Pr[\exists i : \mathcal{W}_i = (0, 0)] = \Pr[\exists i : \mathcal{W}'_i = 0]$$

Andrei Giurgiu has shown in his Master Thesis [9] the following lemma.

Lemma A.10 *For a nonnegative integer random variable T with $E[T] > 1$ and $\Pr[T = 0] > 0$, the probability that the Markov chain with transition probabilities*

as in equations A.13 and A.14 ever reaches 0 *given that it starts at* j *is* $f_j = \lambda^j$*, where* λ *is the unique value in* $(0,1)$ *that satisfies*

$$\lambda = GF_T(\lambda) \tag{A.15}$$

Proof The proof is in [9], however Giurgiu restricts T further such that it is bounded in a sense that there is a $k \in \mathbb{N}$ with $\Pr[T > k] = 0$. For most of the proof this restriction can easily be dropped, so we will only outline the differences. The proof that there is such a λ, such that the success probability is given by λ^j (corollary 2.18) does not require T to be bounded. The same is true for the proof that f_j converges to 0 as j goes to infinity (lemma 2.21). The only difference is the proof that equation A.15 has a unique solution in $(0,1)$ (lemma 2.23). By the definition of $GF_T(\lambda)$ as a probability generating function it can be written as $\sum_{i=0}^{\infty} a_i \lambda^i$ where $\sum_{i=0}^{\infty} a_i = 1$ and $a_i \geq 0$. Therefore 1 is a solution to equation A.15. Furthermore $GF_T(\lambda)$ is convex in the interval $[0,1]$ because it is a sum of positive multiples of x^s for $s \geq 0$ while λ is linear. Therefore there is at most one solution in $[0,1)$. To prove that there is at least one solution we consider roots of

$$h(\lambda) = GF_T(\lambda) - \lambda$$

We have $h(0) > 0$ since $GF_T(0) = \Pr[T = 0] > 0$ by assumption. Furthermore, we have $h'(1) > 0$ because $h'(1) = GF'_T(1) - 1 > 0$ using the fact that $GF'_T(1) = E[T]$ (see the appendix in generating functions) and $E[T] > 1$. Since $h(\lambda)$ is continuous the existence of a root follows directly. □

We can now apply this lemma for analysing $\{\mathcal{W}'_i\}_{i\geq 0}$. It can easily be checked that T as given by definition A.7 fulfils all conditions in lemma A.10 and therefore $f_j = (\lambda')^j$ where λ' is the only solution in $(0,1)$ satisfying

$$\lambda' = \frac{1}{2k} + \frac{k-1}{k}\lambda'^2 + \frac{1}{2k} GF_B(\lambda') \tag{A.16}$$

where we plugged in A.12. The solutions for this equation are

$$\lambda'_1 = 1$$
$$\lambda'_2 = \frac{k+1+\sqrt{(k^2+1)}}{2(k-1)}$$
$$\lambda'_3 = \frac{k+1-\sqrt{(k^2+1)}}{2(k-1)}$$

Since $\lambda'_2 > 1$ we can conclude that

$$\lambda' = \frac{k+1-\sqrt{(k^2+1)}}{2(k-1)} \tag{A.17}$$

We can now give a formula for the success probability of $\{\mathcal{W}'_i\}_{i\geq 0}$.

Lemma A.11 *The success probability* $f = \Pr[\exists i : \mathcal{W'}_i = 0]$ *is given by*

$$f = \left(\frac{k}{4(k-1)}\right)^n \tag{A.18}$$

Proof We can plug in the starting distribution $\mathcal{W'}_0^{(n)}$ given by definition A.4 and get

$$\begin{aligned} f &= \sum_{j=0}^{\infty} \Pr[\mathcal{W'}_0^{(n)} = j] f_j \\ &= \sum_{j=0}^{\infty} \Pr[\mathcal{W'}_0^{(n)} = j] (\lambda')^j \\ &= GF_{\mathcal{W'}_0^{(n)}}(\lambda') \\ &= \left(\frac{1}{4}(1 + 2\lambda' + GF_B(\lambda')\right)^n \end{aligned}$$

Furthermore we can derive from A.16

$$GF_B(\lambda') = 2k\lambda' - 2(k-1)(\lambda')^2 - 1 \tag{A.19}$$

Plugging this in we get

$$\begin{aligned} f &= \left(\frac{1}{4}(2\lambda' + 2k\lambda' - 2(k-1)(\lambda')^2)\right)^n \\ &= \left(\frac{1}{2}((k+1)\lambda' - (k-1)(\lambda')^2)\right)^n \\ &= \left(\frac{1}{2(k-1)}((k+1)(k-1)\lambda' - (k-1)^2(\lambda')^2)\right)^n \end{aligned}$$

Now we can use A.17 and get

$$\begin{aligned} f &= \left(\frac{1}{4(k-1)}\left((k+1)\left(k+1-\sqrt{k^2+1}\right) - \frac{\left(k+1-\sqrt{k^2+1}\right)^2}{2}\right)\right)^n \\ &= \left(\frac{1}{4(k-1)}\left((k+1)^2 + (k+1-k-1)\sqrt{k^2+1} - \frac{1}{2}\left((k+1)^2 - k^2 - 1\right)\right)\right)^n \\ &= \left(\frac{1}{4(k-1)}\left(\frac{1}{2}\left((k+1)^2 - k^2 - 1\right)\right)\right)^n \\ &= \left(\frac{k}{4(k-1)}\right)^n \end{aligned}$$

□

Remains to show that the expected number of steps for $\{\mathcal{W}_i^{(n)}\}_{i\geq 0}$ to reach $(0,0)$, conditioned that it reaches $(0,0)$ at all, is linear in n. The proof is omitted here. We can analyse the expected number of steps for the Markov chain $\{\mathcal{W}'^{(n)}_i\}_{i\geq 0}$ using the results by Giurgiu [9] and then conclude on the Markov chain $\{\mathcal{W}_i^{(n)}\}_{i\geq 0}$ using the coupling between the two chains.

Appendix B

Probability Theory

B.1 Chernoff Bounds

We consider n independent and identically distributed random variables $X_1, \ldots, X_n$ with $\Pr[X_i = 1] = p$ and $\Pr[X_i = 0] = 1 - p$ with the aim of analysing their sum

$$X = \sum_{i=1}^{n} X_i$$

We know that X is *binomially distributed*, i.e.

$$\Pr[X = k] = \binom{n}{k} p^k (1-p)^{n-k} \tag{B.1}$$

We then write $X \sim Bin(n, p)$. The probability density function of a binomially distributed random variable X is shaped like a bell with

$$\mu := E[Bin(n, p)] = np \tag{B.2}$$

Chernoff bounds give a bound on the probability that the random variable X deviates from the mean. They are less general than the inequalities of Markov and Chebyshev, as those can be applied to any (positive) random variables but they are however considerably tighter. We will use Chernoff bounds in the following form

Lemma B.1 *Let $X_1, \ldots, X_n$ be independent and identically distributed Bernoulli random variables with $\Pr[X_i = 1] = p$ and $\Pr[X_i = 0] = 1 - p$. Then the*

following inequalities hold for $X = \sum_{i=1}^{n} X_i$ *and* $\mu = E[X] = np$

$$\Pr[X \geq (1+\delta)\mu] \leq e^{-\frac{\mu\delta^2}{3}} \qquad \textit{for all } 0 < \delta \leq 1 \tag{B.3}$$

$$\Pr[X \leq (1-\delta)\mu] \leq e^{-\frac{\mu\delta^2}{2}} \qquad \textit{for all } 0 < \delta \leq 1 \tag{B.4}$$

$$\Pr[|X-\mu| \geq \delta\mu] \leq 2e^{-\frac{\mu\delta^2}{3}} \qquad \textit{for all } 0 < \delta \leq 1 \tag{B.5}$$

$$\Pr[X \geq t] \leq 2^{-t} \qquad \textit{for } t \geq 2e\mu \tag{B.6}$$

B.2 Generating Functions

Generating function are a very powerful tool in discrete mathematics, as they often allow strong and elegant solutions and many tedious calculations can be shortened using generating functions.

Definition B.2 *The* (probability) generating function *of a nonnegative integer-valued random variable* X *is defined by*

$$GF_X(\lambda) := \sum_{k=0}^{\infty} \Pr[X = k] \cdot \lambda^k = E[\lambda^k]$$

It is known from analysis that the representation of $GF_X(\lambda)$ as a power series is unique. Hence the density function of X is uniquely determined by $GF_X(\lambda)$.

Since

$$GF'_X(\lambda) = \sum_{k=1}^{\infty} k \cdot \Pr[X = k] \cdot \lambda^{k-1} \tag{B.7}$$

we can get the probabilities of X with

$$\Pr[X = i] = \frac{GF_X^{(i)}(0)}{i!} \tag{B.8}$$

There is also a simple connection between generating functions and the expectation. Namely

$$GF'_X(1) = \sum_{k=1}^{\infty} k \cdot \Pr[X = k] = E[X] \tag{B.9}$$

Below are two lemmas about generating functions that are used in this thesis. Both of them can be used to reduce the complexity of some computations.

Lemma B.3 *Let $X_1, \ldots, X_n$ be independent nonnegative integer-valued random variables, and let $Z = X_1 + \ldots + X_n$. Then*

$$GF_Z(\lambda) = GF_{X_1}(\lambda) \cdot \ldots \cdot GF_{X_n}(\lambda)$$

Lemma B.4 *Let $X_1, X_2, \ldots$ be independent and identically distributed random variables that are nonnegative and integer-valued with probability generating function $GF_X(\lambda)$. Let N be another independent nonnegative integer-valued random variable with probability generating function $GF_N(\lambda)$. Then the random variable $Z = X_1 + \ldots + X_N$ has the probability generating function $GF_Z(\lambda) = GF_N(GF_X(\lambda))$. Furthermore*

$$E[Z] = E[N] \cdot E[X]$$

Both sections on probability theory are based on [28].

Bibliography

[1] A. E. Brouwer, A. M. Cohen, and A. Neumaier. *Distance-Regular Graphs*. Springer Verlag, 1984.

[2] R. M. Damerell. Distance-Transitive and Distance-Regular Digraphs. *J. Combin. Theory Ser. B*, 31:46–53, 1981.

[3] Evgeny Dantsin, Andreas Goerdt, Edward A. Hirsch, Ravi Kannan, Jon Kleinberg, Christos Papadimitriou, Prabhakar Raghavan, and Uwe Schöning. A deterministic $(2 - 2/(k + 1))^n$ algorithm for k-SAT based on local search. *Theor. Comput. Sci.*, 289(1):69–83, 2002. ISSN 0304-3975. doi: http://dx.doi.org/10.1016/S0304-3975(01)00174-8.

[4] Martin Davis and Hilary Putnam. A Computing Procedure for Quantification Theory. *J. ACM*, 7(3):201–215, 1960. ISSN 0004-5411. doi: http://doi.acm.org/10.1145/321033.321034.

[5] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Commun. ACM*, 5(7):394–397, 1962. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/368273.368557.

[6] Tomás Feder and Rajeev Motwani. Worst-case time bounds for coloring and satisfiability problems. *J. Algorithms*, 45(2):192–201, 2002. ISSN 0196-6774. doi: http://dx.doi.org/10.1016/S0196-6774(02)00224-9.

[7] Jon William Freeman. *Improvements to propositional satisfiability search algorithms*. PhD thesis, Philadelphia, PA, USA, 1995.

[8] Francis Galton and H. W. Watson. On the Probability of the Extinction of Families. *Journal of the Anthropological Institute of Great Britain*, 4: 138–144, 1875.

[9] Andrei Giurgiu. Random Walk Algorithms for SAT. Master's thesis, ETH Zürich, 2009. doi: http://dx.doi.org/10.3929/ethz-a-005939758.

[10] Andrei Giurgiu, Robin Moser, and Stefan Schneider. Notes on Schöning's Algorithm. Manuscript, 2010.

[11] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1994. ISBN 0201558025.

[12] Geoffrey R. Grimmett and David R. Stirzaker. *Probability and Random Processes*. Oxford University Press, August 2001. ISBN 0198572220.

[13] Thomas Hofmeister, Uwe Schöning, Rainer Schuler, and Osamu Watanabe. A Probabilistic 3-SAT Algorithm Further Improved. In *STACS '02: Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science*, pages 192–202, London, UK, 2002. Springer-Verlag. ISBN 3-540-43283-3.

[14] Kazuo Iwama and Suguru Tamaki. Improved upper bounds for 3-SAT. In *SODA '04: Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 328–328, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics. ISBN 0-89871-558-X.

[15] D. A. Leonard and K. Nomura. The girth of a directed distance-regular graph. *J. Combin. Theory Ser. B*, 58:34–39, 1993.

[16] Liang Li, Xin Li, Tian Liu, and Ke Xu. From k-SAT to k-CSP: Two Generalized Algorithms. *CoRR*, abs/0801.3147, 2008.

[17] Joaão P. Marques-silva and Karem A. Sakallah. GRASP: A Search Algorithm for Propositional Satisfiability. *IEEE Transactions on Computers*, 48:506–521, 1999.

[18] Christos H. Papadimitriou. On selecting a satisfying truth assignment (extended abstract). In *SFCS '91: Proceedings of the 32nd annual symposium on Foundations of computer science*, pages 163–169, Washington, DC, USA, 1991. IEEE Computer Society. ISBN 0-8186-2445-0. doi: http://dx.doi.org/10.1109/SFCS.1991.185365.

[19] R. Paturi, P. Pudlak, and F. Zane. Satisfiability Coding Lemma. In *FOCS '97: Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, page 566, Washington, DC, USA, 1997. IEEE Computer Society. ISBN 0-8186-8197-7.

[20] R. Paturi, P. Pudlák, M.E. Saks, and F. Zane. An Improved Exponential-Time Algorithm for k-SAT. *Foundations of Computer Science, Annual IEEE Symposium on*, page 628, 1998. ISSN 0272-5428. doi: http://doi.ieeecomputersociety.org/10.1109/SFCS.1998.743513.

[21] Daniel Rolf. 3-SAT in RTIME(O(1.32793^n)) - Improving Randomized Local Search by Initializing Strings of 3-Clauses. *Electronic Colloquium on Computational Complexity (ECCC)*, (054), 2003.

[22] Daniel Rolf. Improved bound for the PPSZ/Schöning-algorithm for 3-SAT. *Electronic Colloquium on Computational Complexity (ECCC)*, 159: 2006, 2005.

[23] Dominik Scheder. Guided search and a faster deterministic algorithm for 3-SAT. In *LATIN'08: Proceedings of the 8th Latin American conference on Theoretical informatics*, pages 60–71, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 3-540-78772-0, 978-3-540-78772-3.

[24] Dominik Scheder. Using a Skewed Hamming Distance to Speed Up Deterministic Local Search. *CoRR*, abs/1005.4874, 2010.

[25] Uwe Schöning. A probabilistic algorithm for k-SAT and constraint satisfaction problems. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science, FOCS'99*, pages 410–414, 1999.

[26] Bart Selman, Hector J. Levesque, and D. Mitchell. A New Method for Solving Hard Satisfiability Problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 440–446, Menlo Park, California, 1992. AAAI Press.

[27] Bart Selman, Henry Kautz, and Bram Cohen. Local Search Strategies for Satisfiability Testing. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 521–532, 1995.

[28] Angelika Steger. Randomized Algorithms and Probabilistic Methods. Lecture notes, ETH Zürich, 2009.

[29] Emo Welzl. Boolean Satisfiability - Combinatorics and Algorithms. Lecture notes, ETH Zürich, 2009.

[30] Hantao Zhang. SATO: an Efficient Propositional Prover. In *In Proceedings of the International Conference on Automated Deduction*, pages 272–275. Springer-Verlag, 1997.