Helsinki University of Technology
Department of Electrical and Communications Engineering

Heikki Rantanen

Analyzing the Random-Walk Algorithm for SAT

Master's Thesis

submitted in partial fulfillment of the requirements
for the degree of Master of Science in Technology.

Espoo, October 15, 2004

Supervisor: Prof. Ilkka Niemelä
Instructor: Prof. Ilkka Niemelä

Helsinki University of Technology

Master's Thesis Summary

Author: Heikki Rantanen

Title: Analyzing the Random-Walk Algorithm for SAT

Date: October 15, 2004 | **Pages**: 7+47

Department: Electrical and Communications Engineering | **Professorship**: T-119 Theoretical Computer Science

Supervisor: Prof. Ilkka Niemelä | **Instructor**: Prof. Ilkka Niemelä

Abstract: The propositional logic satisfiability (SAT) problem has gained growing interest in recent years. From complexity theory is known that many real-world mathematical and engineering problems can be translated into SAT problem instances. Advances on SAT solution algorithms in the past ten or fifteen years has enabled solving these problems by a computer. This, for one, has inspired developing even better algorithms.

In this work we study the so-called random-walk algorithm, which is a well-known probabilistic solution method for the SAT problem. The aim is to give insight into the algorithm with analytical methods and answer some open questions. For example, one would like to know the optimal value for the restart-moment—a parameter that has notable effect on the algorithm's performance.

The analysis leads to a stochastic model that is a Markov chain; more precisely a simple random-walk between an absorbing and a reflecting barrier. A scheme to solve the problem for fixed number of variables is obtained from the theory of Markov chains. In general case, however, things get more complicated. Namely such a random-walk, despite its simplicity, shows very complex behaviour and unfortunately no closed-form solution formula is known. Consequently, one must abide by approximative results.

The contribution of this work include solving the optimal restart-moment approximately and a rigorous performance analysis. The calculations give insight into the algorithm's nature underlining the importance of restarts. Also some experimental study is made, and the results support the developed stochastic model.

Keywords: Propositional satisfiability, random-walk algorithm, probabilistic local search, simple random-walk

Teknillinen korkeakoulu

Diplomityön yhteenveto

Tekijä: Heikki Rantanen

Työn nimi: Lauselogiikan toteutuvuusongelman satunnaiskulkualgoritmin analyyttinen tarkastelu

Päiväys: 15. lokakuuta 2004	**Sivumäärä**: 7+47
Osasto: Sähkö- ja tietoliikennetekniikka	**Professuuri**: T-119 Tietojenkäsittelyteoria
Valvoja: Prof. Ilkka Niemelä	**Ohjaaja**: Prof. Ilkka Niemelä

Tiivistelmä: Lauselogiikan toteutuvuuden (SAT) ongelmaa on tutkittu viime vuosina kasvavassa määrin. Kompleksisuusteoriasta tiedämme, että useat käytännön matemaattiset ja tekniset ongelmat voidaan muuntaa vastaamaan SAT-ongelmaa. SAT-ongelman ratkaisumenetelmien 10–15 viime vuoden aikana tapahtuneen kehityksen ansiosta näitä ongelmia voidaan ratkoa tietokoneella. Tämä puolestaan on innostanut kehittämään yhä parempia menetelmiä.

Työssä tutkitaan niin kutsuttua satunnaiskulkualgoritmia, joka on tunnettu probabilistinen SAT-ongelman ratkaisumenetelmä. Työn tarkoituksena on analyyttisen tarkastelun avulla auttaa ymmärtämään algoritmin luonteenpiirteet ja vasta eräisiin avoimiin kysymyksiin. Yksi tällainen kysymys liittyy algoritmin suorituskyvyn maksimointiin uudelleenkäynnistysten avulla — optimaalista hetkeä uudelleenkäynnistykselle ei tiedetä.

Analyysin tuloksena saadaan stokastinen malli, joka on luonteeltaan Markovin ketju, tarkemmin sanottuna yksinkertainen satunnaiskulku absorboivan ja heijastavan seinän välillä. Ratkaisumalli kiinnitetyn muuttujamäärän tapauksessa saadaan suoraan Markovin ketjujen teoriasta. Yleisessä tapauksessa kuitenkin törmätään vaikeuksiin. Kyseisen satunnaiskulun käytös on nimittäin yllättävän kompleksinen, eikä selkeää suljetun muodon ratkaisua tunneta. Tarvitaan siis approksimoivia menetelmiä.

Työn keskeiset tulokset ovat algoritmin optimaalisen uudelleenkäynnistysajankohdan ratkaiseminen approksimatiivisesti ja suorituskyvyn täsmällinen analyysi. Esitetty matemaattinen analyysi auttaa ymmärtämään algoritmin luonteenpiirteitä; se mm. korostaa uudelleenkäynnistysten merkitystä. Lopun kokeelliset tulokset näyttävät tukevan kehitettyä stokastista mallia.

Avainsanat: Lauselogiikan toteutuvuus, satunnaiskulkualgoritmi, probabilistinen paikallinen haku, yksinkertainen satunnaiskulku

Preface

The idea for the subject of my master's thesis came up to me when I was attending the course "T-79.194 Seminar on Theoretical Computer Science". The subject of the seminar was methods for propositional logic satisfiability checking and it was lectured by Prof. Ilkka Niemelä. I am fascinated by the probability calculus, so I chose to do my report on papers [30] and [13] because of the word *probabilistic* in their titles.

While preparing my report I started reconstructing the calculations since I wanted to *understand* how such a simple algorithm is able to be so effective. There seemed to be many questions that neither these two papers nor any other I found addressed. I started feeling that I really could shed more light on this probabilistic algorithm since the underlying stochastic model seemed to be the same I had run into with quantum computing. Even though the problem turned out much more difficult that I had expected, I feel that have managed to do it.

I would like to thank my instructor Prof. Ilkka Niemelä for doing his job very well and enthusiastically and for arranging me some financial support, as well as my fiancée Heli for her support and understanding. Thank you!

Heikki Rantanen

Otaniemi, October 15, 2004

Contents

List of Figures

List of Tables

List of Algorithms

List of Notations

RWA	Random-Walk Algorithm
CCRWA	Complement Checking RWA
CDF	Cumulative Distribution Function
CTRWA	Complement Tracing RWA
DPLL	Davis–Putnam–Logemann–Loveland (method for SAT)
$\overline{a}$	Complement of truth assignment a
k	Number of literals per clause
N	Number of atomic propositions
$[\boldsymbol{v}]_i$	i:th element of a vector $\boldsymbol{v}$
$\mathrm{E}[X]$	Expected value of a random variable X
$\Pr(A)$	Probability of an event A
$\Pr(A\|B)$	Probability of an event A given an event B (conditional probability)
$\mathrm{Cov}[X, Y]$	Covariance of random variables X and Y
$\mathrm{Var}[X]$	Variance of random variable X

1 Introduction

1.1 Propositional Logic Satisfiability Problem

Logic is present in our everyday life. We do logical reasoning ourselves and use technology that is based on it. Apart from the fact that digital circuits are based on logical operations, many of today's intelligent systems apply logic heavily on the software level. There are several types of mathematical logic. The context of this work is propositional logic [20], which is the elementary one having a set of atomic propositions together with *and*, *or*, and *not* operations; see Section 2.1 for a more precise definition.

Logic has origins in philosophy. The mathematical sub-discipline was pioneered by George Boole[1] and Augustus De Morgan[2] in the mid-nineteenth century. However, their work never raised conventional mathematicians' interest. Logic was widely denounced as trivial and useless—proving non-trivial propositions would have required a big amount of mechanical work. But a raising interest in automated computing was about to change the course. [20]

Along the invention of an electrical computer and digital technology in general, a bunch of practical applications for logic arose in both design and utilization of this technology. With the modern programmable computer one was able to automate logical proofs. The classical branch of philosophy was adopted by computer scientists and engineers. Today, logic is heavily applied in e.g. artificial intelligence, logical programming, VHDL, and verification of digital circuits, algorithms, and protocols [20, 3, 2].

The propositional logic satisfiability (SAT) problem goes as follows. If F is a proposition, the problem is to find out if there exists a truth assignment for the atomic propositions in F so that F evaluates *true*. If the answer is yes, then F is called *satisfiable* and the corresponding truth assignment a *model* of F. Usually the proposition is supposed to be in conjunctive normal form (CNF). If the CNF has at most k literals per clause, then we speak of k-SAT problem.

Deciding satisfiability is central since proving e.g. validity or equivalence of propositions is essentially a satisfiability problem. Moreover, propositional logic and the SAT problem have gained growing interest in recent years, since the following solution scheme has proven fruitful for many real-world mathematical and engineering problems [10, 9]. Namely,

1. Represent the problem by propositional logic.

2. Identify the proposition to be decided for satisfiability.

[1] Propositional logic is also known as Boolean logic.
[2] Presented the famous De Morgan laws.

3. Solve the SAT problem.

4. Interpret the result in the original domain.

There is a strong theoretical basis for this being possible. From complexity theory [22] is known that SAT belongs to the class of NP-complete problems, which means that a large set of other problems (class NP) can be converted to SAT efficiently.

The idea in this new paradigm for solving hard problems is that the hard work (phase 3) is done by a computer. Solving SAT lends itself very well to being put on a computer: In both, the power comes from doing simple operations but a huge amount. Instead of solving problems from different fields with expertise in the field in question, one can put all the effort in developing a good SAT solution algorithm and then apply this paradigm. Of course this doesn't work always. In some cases the size of the SAT problem grows drastically with respect to the size of the original problem. As the time requirement of solving SAT, for one, grows exponentially upon the problem size, one may be in trouble.

1.2 SAT Solution Methods

A trivial method for solving SAT would be trying each of the possible truth assignments in turn. In the worst case this would require going through all the 2^N, N being the number of different atomic propositions in F, possibilities. Several algorithms, both deterministic and probabilistic, for solving SAT in time less than $O(2^N)$ have been proposed and implemented; see [10, 12] for a survey. They can be categorized as deterministic and probabilistic, and on the other hand complete and incomplete. A complete algorithm can decide both satisfiability and unsatisfiability while incomplete only one of these.

The random-walk algorithm (RWA) for SAT originally presented by Papadimitriou [21] is probably the simplest, but yet a powerful SAT solution method. It is probabilistic and incomplete. The basic RWA goes as follows. For a start, draw a truth assignment a at random. Then check if a satisfies F. If not, pick one unsatisfied clause and one atom of that clause at random. Modify a so that you flip the truth value of the atom you picked. Check if a now satisfies F. If not, again pick an unsatisfied clause and so on. Continue this scheme until a satisfying assignment is found (or you become exhausted).

Despite its simplicity, this method is a clear advance on the trivial one. The probability that an existing model is found after K flips can be made arbitrarily close to one by setting K sufficiently large (this is justified in Chapter 3). For 2-SAT, Papadimitriou showed a growth rate $O(N^2)$ for the expected run time in the worst case. The same result is derived independently by Ross [27]. Note that proving a proposition unsatisfiable, however, would take infinite time, so RWA is incomplete.

Schöning's [30, 31] (the latter is a revision) novelty was to restart the algorithm after every $3N$ flips, which improvement made it potential for k-SAT in general. He showed an upper bound $O\left[(2 \cdot \frac{k-1}{k})^N\right]$ $(k \geq 3)$. This yields $O[(4/3)^N]$ for $k = 3$, which in fact was the best bound for 3-SAT known by that time. Since then bounds $O(1.3302^N)$ [13], $O(1.3290^N)$ [1], and $O(1.324^N)$ [16] have been achieved

with modifications to Schöning's algorithm. The last is currently the best known result for 3-SAT.

The RWA is an example of probabilistic local search. Other such methods are GSAT [33] and the famous Walksat introduced in [32], which is a combination of the previous two. See [10, Sec. 4] for more local-search methods. Deterministic complete SAT algorithms, for one, are usually based on the DPLL method [5], but state-of-the-art solvers apply different additional search pruning techniques on top of it [18]. Also these methods may benefit from restarts [11]. On the other hand, the DPLL can be randomized [24].

1.3 Objectives

Schöning [31] also makes a conjecture that gives rise expecting a rate even better. Namely, every time one checks if the current assignment is satisfying, its complement, which is an assignment that results when all the truth values are flipped, should be checked too. However, the effect of this on the performance is not studied.[3]

Several questions now arise from [30, 31]:

1. Is the bound $O\left[(2 \cdot \frac{k-1}{k})^N\right]$ strict or is it actually even better?
2. Is the restart moment $3N$ optimal, and if not, what would be?
3. What is the growth rate with the optimal restart moment?
4. What is the effect of complement checking?

The papers [30, 31] are very short in length and thus don't provide deep insight into the problem. For these reasons, it feels entitled to say that the nature of the random-walk algorithm is still more or less mystery. The object of this work is to carefully analyze the algorithm by means of probability theory and with this, to provide deeper understanding of its behavior and tackle the open questions above.

Especially the second question is of great practical importance, since there exists many probabilistic local search implementations, and timing restarts plays a central role in working with them. Numerous experiments have confirmed the fact that performance of these algorithms depends on the parameter of restart moment. But a generally good scheme for setting this parameter is hard to obtain empirically.[4] [14]

The work has the following outline. Chapter 2 formalizes the concept of propositional logic and presents pseudo code for the discussed algorithms. Chapter 3 lays the foundation for the algorithm analysis by building a mathematical model based

[3] Schöning [31, p. 3] writes: "*Also note that the apparent worst case of reaching state n is not bad at all, since the complementary assignment $\overline{a}$ is a satisfying assignment in this case. Therefore, one might modify the algorithm such that it always checks whether the complement of the actual assignment is satisfying. Instead of analyzing this somewhat complicated stochastic process we choose to analyze closely another process*"

[4] Hoos and Stützle [14, p. 32] write: "*But good maxSteps settings are extremely difficult to find a priori and currently, to our best knowledge, there exists no theoretical results on how to effectively determine good settings.*"

on Markov chains. Also the basic RWA (without restarts) is analyzed. The question 4 on page 3 is answered in Section 3.3. In Chapter 4 the restarts are taken into account. That chapter presents the key results of this work; by the end of the chapter we are able to answer the questions 1, 2 and 3. Implementing the RWA is discussed shortly in Chapter 5. Also a new RWA variant, which might be useful in practice when there is absence of true randomness, is proposed. In Chapter 6, some experimental study is made to see if the calculated results can be verified by experiments. The results and observations of this work are summarized and the final conclusions are made in Chapter 7. A little extra mathematical material and the important formulas as Matlab code are given in Appendices A and B, respectively.

2 Conceptual Background

This chapter introduces the concepts of propositional logic, the random-walk algorithm, and the O-notation in more detail.

2.1 Propositional Logic

Syntax

The language of *propositional logic* [20] consists of propositions. The alphabet has symbols

(i) $A, B, C, \ldots$ (*Atomic propositions*)

(ii) $\wedge, \vee, \neg$ (*Connectives*)

(iii) $(,\ )$ (*Parentheses*)

Propositions are formed using the following recursive definition:

Definition 1 (Proposition)**.** String γ is a *proposition*, if and only if γ is either an atomic proposition or of the form

(i) $(\alpha \wedge \beta)$

(ii) $(\alpha \vee \beta)$

(iii) $\neg\alpha$

where α and β are propositions.

Unnecessary parenthesis are usually left out, e.g. $(A \wedge (B \wedge C))$ can be written $A \wedge B \wedge C$. Atomic propositions are called atoms for short.

Semantics

Atomic propositions can be thought of as binary variables and propositions as functions of those variables. To each atomic proposition one can assign a truth value—either *true* or *false*. This is done by a truth assignment:

Definition 2 (Truth Assignment)**.** Let $\mathcal{A} = \{A, B, \ldots\}$ be the set of atomic propositions. A *truth assignment* is a function

$$a : \mathcal{A} \longmapsto \{true, false\}\,.$$

Given a truth assignment a, any proposition γ is either *true* or *false* denoted by $a \models \gamma$ or $a \not\models \gamma$, respectively. If γ is an atomic proposition, then $a \models \gamma$ is equivalent to $a(\gamma) = true$. Otherwise the truth value is determined according to the following rules:

(i) $a \models \alpha \wedge \beta$ if and only if $a \models \alpha$ and $a \models \beta$ (*Conjunction*)

(ii) $a \models \alpha \vee \beta$ if and only if $a \models \alpha$ or $a \models \beta$ (*Disjunction*)

(iii) $a \models \neg\alpha$ if and only if $a \not\models \alpha$ (*Negation*)

If $a \models \gamma$, a is called a *model* of γ. If a proposition has a model, it is *satisfiable*.

Atomic propositions and their negations are called *literal*s. A disjunction of literals is called a *clause*, and conjunction of clauses is called a *conjunctive normal form* (CNF). The following theorem gives a reason why CNF's are interesting.

Theorem 1. *Any proposition has an equivalent conjunctive normal form.*

Proof. See [20, p. 37] for a transformation procedure. □

2.2 Random-Walk Algorithm

The random-walk algorithm requires the proposition to be a CNF. By Theorem 1 any proposition can be transformed to a CNF. The transformation may, however, result in an exponential blow-up in the proposition length. This can be aided by relinquishing the equivalence requirement and using another transformation procedure that introduces new atoms but retains satisfiability.

Algorithm 1 illustrates the basic RWA as pseudo code. Argument F is a proposition in CNF and K serves as a time-out preventing the infinite loop that results if the proposition happens to be unsatisfiable. In line two the truth assignment a is initialized at random (uniform distribution). Then a is transformed gradually in the *for* loop until either it satisfies F or the loop has run K cycles. Each transformation cycle includes the following three steps:

(i) Pick one unsatisfied clause from F at random (line 7)

(ii) Pick one atom form this clause at random (line 8)

(iii) Modify a so that the truth value of this atom is flipped (line 9)

Adding restarts results in Algorithm 2, which we call restarting RWA. The difference compared to the basic RWA is that now the procedure SEARCH is restarted every time the *for* loop has run n_r cycles without success. This is done by nesting the procedure in another *for* loop (line 2). Argument K again serves as a time-out but counts now the number of restarts. Thus the maximum number of flips is $K \cdot n_r$.

The complement checking RWA referred by question 4 on page 3 is depicted by Algorithm 3. The only difference compared to the basic RWA is that every time one checks if the current assignment a is satisfying (line 4), the complement assignment $\overline{a}$, which is obtained by flipping all the truth values, is checked too (line 6).

Algorithm 1 Basic RWA

```
1: procedure SEARCH(F, K)
2:     a ← randomly chosen truth assignment
3:     for repeat K times do
4:         if a ⊨ F then
5:             return a
6:         else
7:             c ← randomly chosen unsatisfied clause of F(a)
8:             s ← randomly chosen atom of c
9:             a ← a except the truth value of s flipped
10:        end if
11:    end for
12:    return false
13: end procedure
```

Algorithm 2 Restarting RWA

```
1: procedure RSEARCH(F, K, n_r)
2:     for repeat K times do
3:         a ← SEARCH(F, n_r)
4:         if a ≠ false then
5:             return a
6:         end if
7:     end for
8:     return false
9: end procedure
```

Algorithm 3 Complement checking RWA

```
1: procedure CCSEARCH(F, K)
2:     a ← randomly chosen truth assignment
3:     for repeat K times do
4:         if a ⊨ F then
5:             return a
6:         else if ā ⊨ F then
7:             return ā
8:         else
9:             c ← randomly chosen unsatisfied clause of F
10:            s ← randomly chosen atom of c
11:            a ← a except the value of s flipped
12:        end if
13:    end for
14:    return false
15: end procedure
```

2.3 Big-O Notation

The following definitions explain the O-symbol [6, Sec. 1.2] (read big-O).

Definition 3 (Big-O)**.** Let f and g be functions of a real variable. Then $f(x)$ is $O[g(x)]$ as $x \to \infty$, denoted by

$$f(x) = O[g(x)] \quad (x \to \infty)\,,$$

if there exists real numbers a and A such that

$$x > a \iff |f(x)| \leq A\,|g(x)|\ .$$

Definition 4. If $f(x)$ is $O\left(a^x\right)$ where a is a real number such that

$$\min_{c \in \mathbb{R}} \left[f(x) = O(c^x)\right] = a\,,$$

then is said $f(x)$ is $O\left(a^x\right)$ strictly.

3 Stochastic Analysis

In this chapter we start analyzing the algorithm. Basic knowledge of probability and perhaps Markov chains will be a prerequisite to follow the text in full; [28, 35] among others serve as a good backup if needed. After identifying the essential random variables, a stochastic model for the algorithm is derived by utilizing Markov chain theory in Section 3.1. This model based on probability matrices will serve the purpose of calculating exact values with fixed N and k (the number of atoms and literals per clause, respectively) throughout this work. Performance analysis without restarts (Algorithm 1) is done in Section 3.2. In Section 3.3 the question 4 on page 3 is answered.

Throughout this work, when we do algorithm analysis, we do it concerning the *worst case*. To begin with, let us assume that the proposition is satisfiable. In the worst case there is only a single model a^* among the 2^N candidates. We would like to know the run time of Algorithm 1, i.e. the amount of time needed for the procedure SEARCH to return a^*. Denote this time by T. The run time T is a probabilistic quantity, that is a *random variable*, and its probability distribution determines how the algorithm performs. Usually the expected value $\mathrm{E}[T]$ is considered as a measure of performance.

Naturally the actual run time in seconds depends on the underlying hardware and software. In analysis one uses a normalized time-unit, typically some operation count. In addition, it typically suffices to find the run-time growth rate since this measure captures all the information on the algorithm's performance that is usually relevant. We choose the unit operation to be one flip, i.e. a whole cycle in the *for* loop of the procedure SEARCH. Hence, T is expressed in flips and the run time in seconds is given by $T \cdot t$ where t is the duration of one flip in seconds. The important thing is that although t depends on the size of the proposition under examination, i.e. t is a function of N, k, and the number of clauses, this dependence is polynomial. Consequently, to extract the exponential growth rate of the expected run time, it suffices to extract that of $\mathrm{E}[T]$.

Now consider the procedure's state after n flips. A reasonable measure of how close we are is the number of atoms whose value in a and a^* is different; denote this number by $\mathrm{d}(a, a^*)$. If X_n denotes $\mathrm{d}(a, a^*)$ after n flips, then X_n is a random variable with $0 \leq X_n \leq N$ and $X_T = 0$ by definition. If $X_n > 0$, there is at least one unsatisfied clause. We pick one of those, and make it satisfied by flipping one of its k atoms.

Unfortunately, we might flip an atom that already has the correct value and cause $X_{n+1} = X_n + 1$. In the best case there isn't such atoms, but in the worst case only one of the k atoms has an incorrect value. Hence for the probability of event $X_{n+1} = X_n - 1$, given that $0 < X_n < N$, holds

$$\frac{1}{k} \leq \Pr(X_{n+1} = X_n - 1 | 0 < X_n < N) \leq 1. \tag{3.1}$$

The boundary states zero and N are special. When $X_n = N$, the trial assignment is completely opposite and transition to $X_n - 1$ is guaranteed therefore. Once $X_n = 0$, we have found a satisfying assignment and the evolution terminates. This is expressed by

$$\Pr(X_{t+1} = X_t - 1 | X_t = N) = 1\,, \tag{3.2a}$$
$$\Pr(X_{t+1} = X_t | X_t = 0) = 1\,. \tag{3.2b}$$

Initially we draw the N truth values randomly and independently with success probability $1/2$. Thereby X_0 is binomially distributed with parameters N, $1/2$ and, consequently, has point probabilities, expected value, and variance

$$\Pr(X_0 = i) = \mathrm{bin}_{N,\frac{1}{2}}(i)\,, \tag{3.3a}$$
$$\mathrm{E}[X_0] = \frac{N}{2}\,, \tag{3.3b}$$
$$\mathrm{Var}[X_0] = \frac{N}{4}\,, \tag{3.3c}$$

respectively, where

$$\mathrm{bin}_{n,p}(i) = \binom{n}{i} p^i (1-p)^{n-i} \tag{3.4}$$

is the binomial point probability function. We will use ω_i as a shorthand for $\Pr(X_0 = i)$. The total probability of event A is obtained as a weighted average of its conditional probabilities:

$$\Pr(A) = \Pr(A|X_0 = 0)\omega_0 + \ldots + \Pr(A|X_0 = N)\omega_N\,. \tag{3.5}$$

Numbers $\{X_n,\, n \in \mathbb{N}\}$ form a sequence of random variables—a discrete time *stochastic process* [4, 25, 7, 26, 23] in other words. The process is restricted by boundary conditions (3.2a) and (3.2b) called reflecting and absorbing, respectively. The theory of stochastic processes will be our apparatus to approach the problem.

3.1 Markov Chain Model

Due to the fact that the state transition probabilities (3.1) depend on n and the proposition itself, the actual stochastic process is extremely complex. This restricts the analysis to the worst-case where the lower bound is taken. Then

$$\Pr(X_{t+1} = X_t - 1 | 0 < X_t < N) = \frac{1}{k} = \alpha\,, \tag{3.6a}$$
$$\Pr(X_{t+1} = X_t + 1 | 0 < X_t < N) = \frac{k-1}{k} = \beta \tag{3.6b}$$

independently of F.

Now that the transition probabilities are constants it's obvious that

$$\Pr(X_{n+1} = x | X_0 = x_0, \ldots, X_n = x_n) = \Pr(X_{n+1} = x | X_n = x_n), \tag{3.7}$$

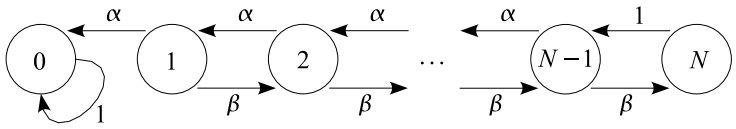


Figure 3.1 States and transition probabilities of the Markov chain $\{X_n\}$

which states that the state transition probabilities depend on the present state only, not the past ones. This is known as Markov property—a prerequisite for a stochastic process $\{X_n\}$ being a *Markov chain*[1]. The resulting Markov chain is illustrated in Figure 3.1. This kind of a chain where transitions to the adjacent states only are possible, is known as a *simple random walk* [4, Ch. 2], [7, Ch. 14].

What is the probability that the model is found by say n flips? The question is addressed by the n-step transition probabilities $\Pr(X_{m+n} = j | X_m = i) = p_{i,j}^{(n)}$ of the Markov chain. The quantity of interest would be the n-step absorption probability $\Pr(X_n = 0)$.

The n-step transition probabilities obey the Chapman-Kolmogorov equation

$$p_{i,j}^{(a+b)} = \sum_{k=0}^{N} p_{i,k}^{(a)} p_{k,j}^{(b)} \,. \tag{3.8}$$

They are usually calculated utilizing matrices. An n-step transition probability matrix $\boldsymbol{P}^{(n)}$ is a square matrix with elements $p_{i,j}^{(n)}$. The one-step transition probability matrix can be formed by Eq. (3.6) and (3.2), which yield a tri-diagonal matrix

$$\underset{N+1\times N+1}{\boldsymbol{P}^{(1)}} = \begin{pmatrix} 1 & 0 & & & & \\ \alpha & 0 & \beta & & 0 & \\ & \alpha & 0 & \beta & & \\ & & \ddots & \ddots & \ddots & \\ & 0 & & \alpha & 0 & \beta \\ & & & & 1 & 0 \end{pmatrix} = \boldsymbol{P} \,. \tag{3.9}$$

Now one can write Eq. (3.8) in matrix form $\boldsymbol{P}^{(a+b)} = \boldsymbol{P}^{(a)} \boldsymbol{P}^{(b)}$, which immediately implicates

$$\boldsymbol{P}^{(n)} = \boldsymbol{P}^{n} \,. \tag{3.10}$$

We will denote $\Pr(X_n = 0)$ by $\rho(n)$ for short. Using Eq. (3.5),

$$\rho(n) = \sum_{i=0}^{N} p_{i,0}^{(n)} \omega_i \tag{3.11}$$

$$= \mathrm{E}\left[p_{X_0,0}^{(n)} \right] \,. \tag{3.12}$$

[1] Markov chains are named after Russian mathematician A. A. Markov (1856–1922) who introduced them in 1907. Since then they have proven extremely useful in modeling various systems and become an essential part of the probability theory.

If $\boldsymbol{p}^{(n)}$ is a row-vector such that $p_i^{(n)} = \Pr(X_n = i)$, by Eq. (3.5) and (3.10) we have that $\boldsymbol{p}^{(n)} = \boldsymbol{\omega} P^n$. The n-step absorption probability $\rho(n)$ obtained by taking the first element of $\boldsymbol{p}^{(n)}$:

$$\rho(n) = [\boldsymbol{\omega} P^n]_0 \tag{3.13}$$

where $[\boldsymbol{v}]_0$ denotes the first element of a vector. Below is a simple example.

Example 1. Calculate the lower-bound for the probability that Algorithm 1 returns a model, in case of 3-SAT, $N = 10$, and $K = 100$.

Solution. For $k = 3$ we have that $\alpha = \frac{1}{3}$ and $\beta = \frac{2}{3}$ (Eq. (3.6)). Form the initial distribution vector by Eq. (3.3a), which yields

$$\boldsymbol{\omega} \approx (0.001\ 0.010\ 0.044\ 0.117\ 0.205\ 0.246\ 0.205\ 0.117\ 0.044\ 0.010\ 0.001)\,.$$

Then form the one-step transition-probability matrix as in Eq. (3.9) and calculate $\boldsymbol{p}^{(100)} = \boldsymbol{\omega} P^{100}$ (Eq. 3.9) with e.g. the Matlab software. This yields

$$\boldsymbol{p}^{(100)} \approx (0.071\ 0.001\ 0.002\ 0.005\ 0.010\ 0.021\ 0.043\ 0.087\ 0.175\ 0.350\ 0.234)\,.$$

The result is $\rho(100) \approx 0.07$. ♢

RWA's ability to solve SAT problems ensues from the following theorem.

Theorem 2. *If $\{X_n\}$ is the Markov chain of Figure 3.1 with $\alpha > 0$, then*

$$\lim_{n\to\infty} p_{i,0}^{(n)} = 1$$

for all $0 \leq i \leq N$ and finite N.

Proof. The theorem follows from a more general theorem concerning reducible Markov chains found in e.g. [4, p. 125]. The proof employs properties of the transition probability matrix; see [34] for a more extensive study. □

Function (3.13) with $\alpha = 1/3$ and three different N is plotted in Figure 3.2. The figure seems to accord with Theorem 2 and gives a flavor of the fact that the number of steps needed to attain a certain probability grows exponentially fast with respect to N.

3.2 Expected Run Time

The worst-case run time of the basic RWA is equal to the first passage time to state zero in the previous Markov chain, namely

$$T = \min_{n\in\mathbb{N}}\{X_n = 0\}\,. \tag{3.14}$$

In this section its expected value $\mathrm{E}[T]$ is calculated.

Expected first passage times of a simple random walk are usually calculated in the following manner [7, p. 348]. Take a look at Figure 3.1 again. Suppose we start from

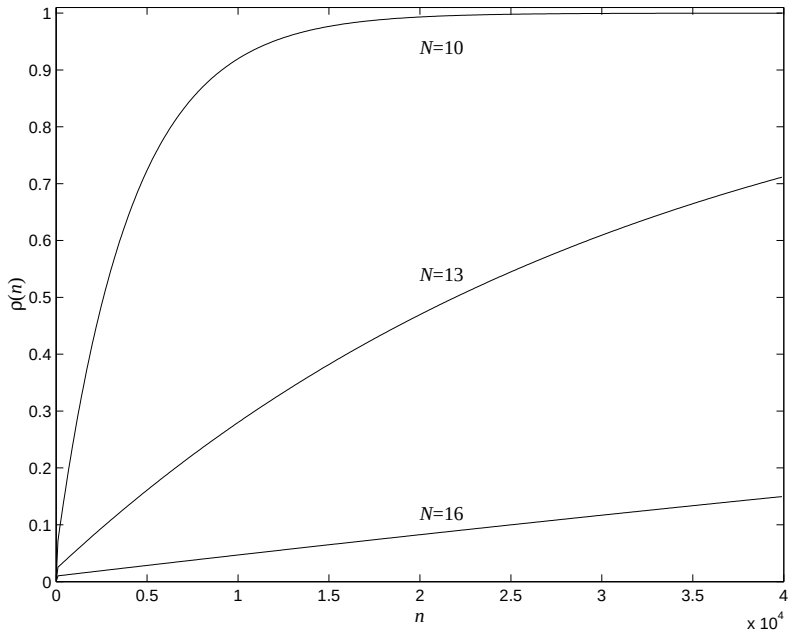


Figure 3.2 The n-step absorption probability of the Markov chain $\{X_n\}$ with $\alpha = \frac{1}{3}$ (Equation (3.13)). This corresponds to the worst-case success probability of the basic RWA after n flips with 3-SAT.

state i, and denote the expected number of steps to reach zero by y_i. Then suppose the first step is taken to the left. Now the situation is same than starting from $i-1$ except that we have already one step in our account—that is to say $y_i = 1 + y_{i-1}$. Considering both first step possibilities we conclude that

$$y_i = 1 + \alpha y_{i-1} + \beta y_{i+1} \tag{3.15}$$

when $0 < i < N$, and

$$y_0 = 0\,, \tag{3.16a}$$

$$y_N = 1 + y_{N-1}\,. \tag{3.16b}$$

Equation (3.15) can be easily solved with the standard method for linear difference equations [29, p. 214]. The general solution reads

$$y_i = \begin{cases} A\left(\frac{\alpha}{\beta}\right)^i + B - \frac{i}{\beta-\alpha} & \alpha \neq \beta \\ Ai + B - i^2 & \alpha = \beta = \frac{1}{2} \end{cases} \tag{3.17}$$

where A and B are arbitrary constants. They are determined by the boundary conditions (3.16). After short calculations,

$$A = \begin{cases} -B & \alpha \neq \beta \\ 2N & \alpha = \beta = \frac{1}{2} \end{cases}, \quad B = \begin{cases} C\left(\frac{\beta}{\alpha}\right)^N & \alpha \neq \beta \\ 0 & \alpha = \beta = \frac{1}{2} \end{cases}. \tag{3.18}$$

where $C = \frac{2\alpha\beta}{(\beta-\alpha)^2}$.

One obtains $\mathrm{E}[T]$ by simply averaging y_i over the initial distribution in Eq. (3.3a):

$$\mathrm{E}[T] = \sum_{i=0}^{N} \binom{N}{i} 2^{-N} y_i \tag{3.19}$$

Substituting Eq. (3.17) with $\alpha \neq \beta$ into this yields

$$\begin{aligned}
\mathrm{E}[T] &= \sum_{i=0}^{N} \binom{N}{i} 2^{-N} \left[A \left(\frac{\alpha}{\beta}\right)^i + B - \frac{i}{\beta - \alpha} \right] \\
&= 2^{-N} A \sum_{i=0}^{N} \binom{N}{i} \left(\frac{\alpha}{\beta}\right)^i + \sum_{i=0}^{N} \binom{N}{i} 2^{-N} B - \frac{2^{-N}}{\beta - \alpha} \sum_{i=0}^{N} \binom{N}{i} i \\
&= 2^{-N} A \left(1 + \frac{\alpha}{\beta}\right)^N + B - \frac{2^{-N}}{\beta - \alpha} N 2^{N-1} \\
&= A \left(\frac{1 + \frac{\alpha}{\beta}}{2}\right)^N + B - \frac{N}{2(\beta - \alpha)} ,
\end{aligned}$$

where the binomial theorem and another summation formula [29, p. 189] was used in getting the third equality. Substituting Eq. (3.18) into this yields

$$\begin{aligned}
\mathrm{E}[T] &= -C \left(\frac{\beta}{\alpha}\right)^N \left(\frac{1 + \frac{\alpha}{\beta}}{2}\right)^N + C \left(\frac{\beta}{\alpha}\right)^N - \frac{N}{2(\beta - \alpha)} \\
&= C \left(\frac{\beta}{\alpha}\right)^N - C \left(\frac{1}{2\alpha}\right)^N - \frac{N}{2(\beta - \alpha)} .
\end{aligned} \tag{3.20}$$

Similarly for Eq. (3.17) with $\alpha = \beta = \frac{1}{2}$:

$$\begin{aligned}
\mathrm{E}[T] &= \sum_{i=0}^{N} \binom{N}{i} 2^{-N} \left[Ai + B - i^2 \right] \quad \text{| Eq. (3.18)} \\
&= 2^{-N} 2N \sum_{i=0}^{N} \binom{N}{i} i - 2^{-N} \sum_{i=0}^{N} \binom{N}{i} i^2 \\
&= 2^{-N} 2N N 2^{N-1} - 2^{-N} (N^2 + N) 2^{N-2} \\
&= \frac{3}{4} N^2 - \frac{1}{4} N .
\end{aligned} \tag{3.21}$$

Equation (3.21) confirms the result that 2-SAT is solvable in polynomial time. Equation (3.20), for one, is $O[(\beta/\alpha)^N] = O[(k-1)^N]$, which means that the basic RWA is feasible for 2-SAT only.

3.3 Complement Checking

Next we deal with the complement checking RWA (Algorithm 3) and give an answer to the question 4 on page 3.

For $k \geq 3$ there is a positive drift in the chain and hitting the reflecting boundary is much more probable, as we could see from Example 1. But note that in state N the complement assignment is satisfying. This suggests checking always the complement assignment as well, like in Algorithm 3. This interesting idea was proposed by Schöning [31, p. 3], but he did not study it further.

In fact, analyzing the effect of complement checking isn't difficult. An immediate consequence in the Markov chain model is that the state N becomes absorbing as well. Redoing the calculations of the previous section with the boundary conditions changed accordingly, yields a polynomial run time for also 3-SAT and higher. So here we have a polynomial time algorithm for an NP-complete problem! Or have we? Unfortunately we also have to go back to Eq. (3.1) and ask "What is the worst case now?".

On the grounds of Eq. (3.1), the case $P(X_{n+1} = N - 2 | X_n = N - 1) = 1$ is possible indeed, meaning the state N may be unreachable. In this case checking the complement is naturally pointless.

The result that checking the complement is not worth the extra effort is easy to confirm by straight thinking. Assume $X_n = N - 1$. If complement checking is used, then the event $X_{n+1} = N$ is desired. But the assumption $P(X_{n+1} = N) = \frac{k-1}{k}$ would require, first of all, that the single atom with the correct value appears in an unsatisfied clause and such clause gets selected. This is of course anything but certain. Secondly, since one is now chasing the one correct atom among k possibilities, $P(X_{n+1} = N)$ is more like $\frac{1}{k}$. In addition, carrying out the same reasoning with assumption $X_n = N - 2$ shows that getting into $N - 1$ isn't itself that easy.

4 Effect of Restarts

Now the restarts (Algorithm 2) are taken into account. This chapter presents the key results of this work. An explicit expression for the run time is provided in Section 4.1.1. The growth rate of this expression is extracted in Section 4.1.2. In Section 4.2 the optimal restart moment is solved. By the end of this chapter we are able to answer the questions 1, 2 and 3 on page 3.

Obviously, if X_n is notably over $\mathrm{E}[X_0] = N/2$, it would be worthwhile to draw a new assignment out of the hat—to start the procedure all over again in other words. Of course one doesn't know when this is the case, since only whether X_n is zero or not is known. However, introducing a predefined restart moment n_r works well too, improving the performance tremendously for 3-SAT and higher. Schöning [30, 31] considers a case $n_r = 3N$.

Let a random variable S be the number of calls to the procedure SEARCH until the model is returned. The procedure has a success probability $\rho(n_r)$, and different procedure calls are independent of each other. This implies that S is geometrically distributed with parameter $\rho(n_r)$. Hence

$$\mathrm{E}[S] = \frac{1}{\rho(n_r)}\,, \tag{4.1a}$$

$$\mathrm{Var}[S] = \frac{1-\rho(n_r)}{\rho(n_r)^2}\,. \tag{4.1b}$$

The worst-case run time T is equal to the total number of flips needed. The model is returned at the Sth procedure call and so $n_r(S-1) \leq T \leq n_r S$. Again, we consider the worst case and put

$$T = n_r S\,. \tag{4.2}$$

By the previous three equations and the calculation rules of expected value and variance [29] we have that

$$\begin{aligned} \mathrm{E}[T] &= \mathrm{E}[n_r S] \\ &= n_r\,\mathrm{E}[S] \\ &= \frac{n_r}{\rho(n_r)} \end{aligned} \tag{4.3}$$

and

$$\begin{aligned} \mathrm{Var}[T] &= \mathrm{Var}[n_r S] \\ &= n_r^2\,\mathrm{Var}[S] \\ &= \left(\frac{n_r}{\rho(n_r)}\right)^2 [1-\rho(n_r)]\,. \end{aligned} \tag{4.4}$$

Function (4.3) using Eq. (3.13) is plotted against N in Figure 4.1. Figure 4.1a shows definitely a great potential for the RWA with restarts in solving 3-SAT. From Figure 4.1b is seen more clearly that the limit $3N$ is not optimal as using $n_r = 1.2N$ more than halves the run time. Notice from Eq. (4.4), however, that chance fluctuations in the run time can be remarkable. As $\mathrm{Var}[T] \approx \mathrm{E}[T]^2$, the expected deviation from $\mathrm{E}[T]$ is of the same order than $\mathrm{E}[T]$ itself.

4.1 Expected Run Time

4.1.1 Explicit Expression

In order to get analytical results, one has to solve the n-step absorption probability $\rho(n)$ appearing in Eq. (4.3). Unfortunately, a simple random-walk between an absorbing and a reflecting barrier is a very tough bite in general and leads to highly convoluted expressions [36]. In Schöning's analysis the reflecting barrier is removed, resulting in a simple random walk $\{Y_n\}$ on natural numbers with an absorbing barrier at zero. This half infinite version is illustrated in Figure 4.2.

Notice that if the initial state is i, it takes a least $2N - i$ steps to hit both of the barriers. Consequently, as long as n is smaller than $2N-i$, the absorption probability $p_{i,0}^{(n)}$ is the same whether there is a barrier at N or not. When $n \geq 2N - i$, absence of the reflecting barrier would make $p_{i,0}^{(n)}$ smaller and, especially, $p_{i,0}^{(\infty)} = 1$ would be no longer true. Taking the initial distribution into account, it holds that

$$\Pr(X_n = 0) = \Pr(Y_n = 0)\,, \quad n < N\,, \tag{4.5a}$$

$$\Pr(X_n = 0) > \Pr(Y_n = 0)\,, \quad n \geq N\,. \tag{4.5b}$$

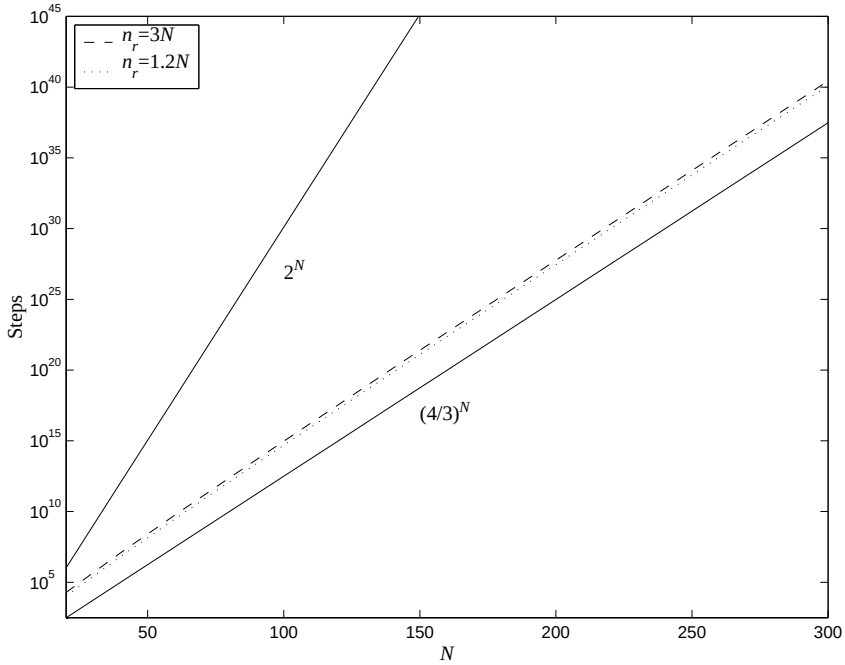
For Y_n a solution is available [25, p. 56], [7, p. 352]. One way for the solution is using the ballot theorem [7, p. 69] as Schöning did. We, however, will utilize the method of images [7, p. 369] because the expression for $\rho(n)$ obtained that way will make the task of Sections 4.2 and 4.2 easier.

Let us first ignore the boundaries and consider a simple random walk $\{Z_n\}$ on integers. This unrestricted case is defined by a recurrence $Z_n = Z_{n-1} + D_n$ where D_n is a random variable with $\Pr(D_n = -1) = \alpha$ and $\Pr(D_n = +1) = \beta$ for all n. Finding the n-step transition probabilities $\Pr(Z_n = j | Z_0 = i) \equiv v_{i,j}^{(n)}$, $i, j \in \mathbb{Z}$, is easy since $Z_n = i + D_1 + \ldots + D_n$ and the D_k's are independent corresponding to n coin flips. The number of $+1$'s is therefore binomially distributed with parameters n and β. If $+1$ occurs say k times, then $j = i + k - (n - k)$, which yields $k = \frac{n+j-i}{2}$. Hence

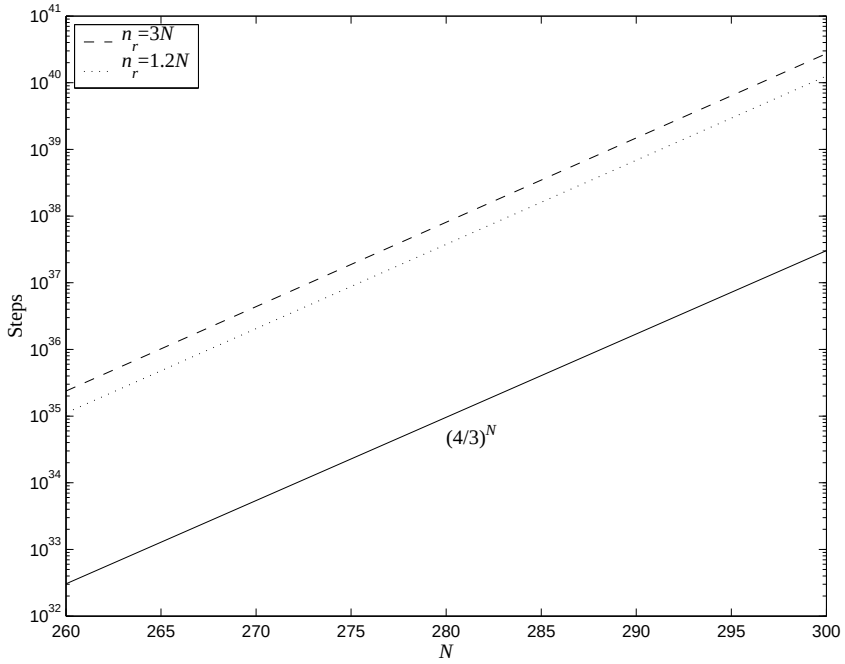
$$v_{i,j}^{(n)} = \mathrm{bin}_{n,\beta}\left(\frac{n+j-i}{2}\right)\,. \tag{4.6}$$

Probabilities $u_{i,j}^{(n)} = v_{i,j}^{(n)}$ as in Eq. (4.6) satisfy by definition a difference equation

$$u_{i,j}^{(n)} = \alpha u_{i,j+1}^{(n-1)} + \beta u_{i,j-1}^{(n-1)} \tag{4.7}$$



(a)



(b)

Figure 4.1 Expected time to absorption of the Markov chain $\{X_n\}$ with $\alpha = \frac{1}{3}$ and restart moment n_r (Equation (4.3)). This corresponds to the worst-case run time of the restarting RWA with 3-SAT and restart moment n_r.

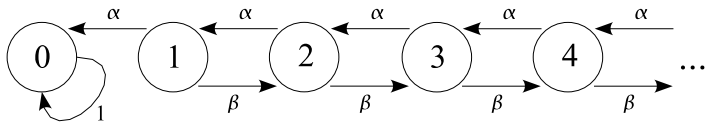


Figure 4.2 The half-infinite Markov chain $\{Y_n\}$

with an initial condition

$$u_{i,j}^{(0)} = \begin{cases} 1 & j = i \\ 0 & j \neq i \end{cases} . \tag{4.8}$$

This is also easy to verify.

Next we place an absorbing barrier at the origin, resulting the aforementioned random walk $\{Y_n\}$. If we put $u_{i,j}^{(n)} = \Pr(Y_n = j | Y_0 = i)$ with $i, j \geq 1$, Eq. (4.7) has a boundary condition $u_{i,1}^{(n)} = \alpha u_{i,2}^{(n-1)}$ or

$$u_{i,0}^{(n)} \equiv 0 . \tag{4.9}$$

We obtain the probabilities $u_{i,j}^{(n)}$ by solving the difference equation (4.7) with initial and boundary conditions (4.8) and (4.9), respectively.

An expression $v_{i,j}^{(n)} + c v_{-i,j}^{(n)}$ where c is a constant also satisfies Eq. (4.7) since it's a linear combination of two solutions. It satisfies the initial condition (4.8) as well, since $v_{-i,j}^{(0)}$ is zero in the domain in question ($j \geq 0$).

This is the so-called method of images: The point source at i is mirrored with respect to the barrier, and the resulting image source is multiplied with a constant c. The idea is that c is chosen so that the boundary condition gets satisfied. A short calculation yields $c = -(\alpha/\beta)^i$. Hence

$$u_{i,j}^{(n)} = v_{i,j}^{(n)} - \left(\frac{\alpha}{\beta}\right)^i v_{-i,j}^{(n)} , \tag{4.10}$$

satisfies Eq. (4.7), (4.8), and (4.9), and thus represents the transition probabilities $\Pr(Y_n = j | Y_0 = i)$.

So now, by virtue of Eq. (4.5a) and u's definition, $p_{i,j}^{(n)} = u_{i,j}^{(n)}$ when $j \geq 1$, $n < N$. Putting this together with Eq. (4.10) and (4.6) yields

$$p_{i,j}^{(n)} = \mathrm{bin}_{n,\beta}\left(\frac{n+j-i}{2}\right) - \left(\frac{\alpha}{\beta}\right)^i \mathrm{bin}_{n,\beta}\left(\frac{n+j+i}{2}\right) \tag{4.11}$$

for $j \geq 1$, $n < N$. Now we can calculate the absorption probability:

$$\begin{aligned} p_{i,0}^{(n)} &= 1 - \sum_{j=1}^{\infty} p_{i,j}^{(n)} \quad \mid \text{Eq. (4.11)} \\ &= 1 - \sum_{j=1}^{\infty} \mathrm{bin}_{n,\beta}\left(\frac{n+j-i}{2}\right) + \left(\frac{\alpha}{\beta}\right)^i \sum_{j=1}^{\infty} \mathrm{bin}_{n,\beta}\left(\frac{n+j+i}{2}\right) \\ &= 1 - \sum_{k=\lceil \frac{n+1-i}{2} \rceil}^{\infty} \mathrm{bin}_{n,\beta}(k) + \left(\frac{\alpha}{\beta}\right)^i \sum_{k=\lceil \frac{n+1+i}{2} \rceil}^{\infty} \mathrm{bin}_{n,\beta}(k) \\ &= \mathrm{Bin}_{n,\beta}\left(\frac{n-i}{2}\right) + \left(\frac{\alpha}{\beta}\right)^i \left[1 - \mathrm{Bin}_{n,\beta}\left(\frac{n+i}{2}\right)\right] \;, \quad n < N \,. \end{aligned} \tag{4.12}$$

where

$$\mathrm{Bin}_{n,\beta}(x) = \sum_{k=0}^{x} \mathrm{bin}_{n,\beta}(k) \tag{4.13}$$

is the binomial cumulative distribution function (CDF). Averaging Eq. (4.12) over the initial distribution gives the wanted

$$\rho(n) = \sum_{i=0}^{N} \omega_i \left[\mathrm{Bin}_{n,\beta}\left(\frac{n-i}{2}\right) + \left(\frac{\alpha}{\beta}\right)^i \left[1 - \mathrm{Bin}_{n,\beta}\left(\frac{n+i}{2}\right)\right]\right] \tag{4.14}$$

Substituting Eq. (4.14) into Eq. (4.3) together with Eq. (4.13) and (3.3a) yields

$$\mathrm{E}[T] = \frac{n_r 2^N}{\sum_{i=0}^{N}\sum_{k=0}^{K_1} \binom{N}{i}\binom{n}{k}\beta^k\alpha^{n-k} + \sum_{i=0}^{N}\sum_{k=K_2}^{n} \left(\frac{\alpha}{\beta}\right)^i \binom{N}{i}\binom{n}{k}\beta^k\alpha^{n-k}} \;, \; n < N \,. \tag{4.15}$$

where $K_1 = \frac{n-i}{2}$ and $K_2 = \lceil \frac{n+1+i}{2} \rceil$.

Schöning's equivalent to Eq. (4.12) looks different because he employed a different method (ballot theorem [7]) in deriving it. This form, however, will make the task of asymptotic analysis easier and also serves the purposes of Section 4.2 better. Equality between Eq. (4.12) and Schöning's equivalent is shown in Appendix A.

4.1.2 Asymptotic Analysis

In this subsection the growth rate of $\mathrm{E}[T]$ is extracted. Schöning shows that

$$\Pr(Y_{3N} = 0) = \mathrm{poly}(N) \cdot (2\beta)^{-N} \tag{4.16}$$

where $\mathrm{poly}(N)$ denotes a polynomial of N. This, together with Eq. (4.3) and (4.5b), provides an upper bound

$$\mathrm{E}[T] < \mathrm{poly}(N) \cdot (2\beta)^N \tag{4.17}$$

for $n_r = 3N$. Equation (4.17) now says that $\mathrm{E}[T] = O[(2\beta)^N]$, but nothing guarantees that $\mathrm{E}[T] = O[(2\beta)^N]$ strictly (see Definition 4 on page 8). A further question

that arises is, how does Eq. (4.16) change if Y is changed to X and $3N$ to the optimal value n_{opt}. We will be able to answer this question by the end of this chapter.

A rigorous way of showing that a function $f(x)$ is $O[\gamma^x]$ strictly is to find upper and lower bounds $f_+(x)$ and $f_-(x)$, respectively, so that $f_-(x) \leq f(x) \leq f_+(x)$, and then show that both $f_+(x)$ and $f_-(x)$ are $O[\gamma^x]$ strictly. The idea is seeking bounds in such a form that its strict growth-rate is seen trivially.

To begin with this, we write Eq. (4.14) in an alternative form (see Eq. (3.12))

$$\begin{aligned}\rho(n) &= \mathrm{E}\left[\mathrm{Bin}_{n,\beta}\left(\frac{n-X_0}{2}\right) + \left(\frac{\alpha}{\beta}\right)^{X_0}\left[1 - \mathrm{Bin}_{n,\beta}\left(\frac{n+X_0}{2}\right)\right]\right] \\ &= \mathrm{E}\left[\Pr\left(Z \leq \frac{n-X_0}{2}\right) + \left(\frac{\alpha}{\beta}\right)^{X_0}\Pr\left(Z > \frac{n+X_0}{2}\right)\right] \qquad (4.18)\end{aligned}$$

where Z is a $\mathrm{Bin}(n,\beta)$ distributed random variable. The following theorems will be utilized in extracting a lower and an upper bound for this function.

Theorem 3 (Markov's Inequality)**.** *If X is a non-negative random variable and $a > 0$, then*

$$\Pr(X \geq a) \leq \frac{\mathrm{E}[X]}{a}\,.$$

Theorem 4. *If X is a Bin(n,β) distributed random variable and $\beta \geq \frac{1}{2}$, then*

$$\Pr(X \leq a) \leq \left(\frac{\alpha}{\beta}\right)^{n-2a}\Pr(X \geq n-a)\,.$$

where $\alpha = 1-\beta$.

Proof. The point probabilities can be written

$$\begin{aligned}\Pr(X=k) &= \binom{n}{k}\beta^k\alpha^{n-k} \\ &= \binom{n}{n-k}\beta^{n-k}\alpha^k\beta^{2k-n}\alpha^{n-2k} \\ &= \Pr(X=n-k)\left(\frac{\beta}{\alpha}\right)^{2k-n}.\end{aligned}$$

Using this,

$$\begin{aligned}\Pr(X \leq a) &= \sum_{k=0}^{a}\Pr(X=k) = \sum_{k=0}^{a}\Pr(X=n-k)\left(\frac{\beta}{\alpha}\right)^{2k-n} \\ &\leq \left(\frac{\beta}{\alpha}\right)^{2a-n}\sum_{k=0}^{a}\Pr(X=n-k) \\ &= \left(\frac{\alpha}{\beta}\right)^{n-2a}\sum_{k=n-a}^{n}\Pr(X=j) = \left(\frac{\alpha}{\beta}\right)^{n-2a}\Pr(X \geq n-a)\,.\end{aligned}$$

□

Theorem 5. *If Z and X are random variables so that Z is Bin(n, β) distributed and X takes integer values with $X \geq \mathrm{E}[Z]$, then*

$$\mathrm{E}\left[\Pr(Z \geq X)\right] \geq \Pr(Z \geq \mathrm{E}[X])$$

Proof. Denote $\Pr(Z \geq z)$ by $Q(z)$ and let $k \in 0, 1, \ldots, n$. Then denote by $Q'(z)$ the function that results when points $Q(k)$ are connected with straight lines. Then it holds that $Q'(k) = Q(k)$ and in general $Q'(z) \geq Q(z)$. Since the greatest value of the difference

$$Q'(k) - Q'(k+1) = \Pr(Z = k)$$

is obtained when k is nearest to $\mathrm{E}[Z] = n\beta$, this point acts as an "inflection point" of $Q'(z)$. Hence $Q'(z)$ is concave when $z \leq n\beta$ and convex when $z \geq n\beta$. From Jensen's inequality [29, p. 411], which states that

$$\mathrm{E}[f(X)] \geq f(\mathrm{E}[X])$$

for a convex function f, follows $\mathrm{E}[Q'(X)] \geq Q'(\mathrm{E}[X])$ when $X \geq n\beta$. The theorem now follows from the fact that $Q'(X) = Q(X)$. □

To obtain lower and upper bounds for Eq. (4.18) we drop the first term and apply Theorem 4 to it, respectively:

$$\begin{aligned}\mathrm{E}\left[\left(\frac{\alpha}{\beta}\right)^{X_0} \Pr\left(Z > \frac{n + X_0}{2}\right)\right] &\leq \rho(n) \\ &\leq \mathrm{E}\left[\left(\frac{\alpha}{\beta}\right)^{X_0} \Pr\left(Z \geq \frac{n + X_0}{2}\right) + \left(\frac{\alpha}{\beta}\right)^{X_0} \Pr\left(Z > \frac{n + X_0}{2}\right)\right].\end{aligned}$$

One can replace the $>$ sign in the second term of the upper bound by $\geq$ since this increases it:

$$\mathrm{E}\left[\left(\frac{\alpha}{\beta}\right)^{X_0} \Pr\left(Z \geq \frac{n + X_0}{2}\right)\right] \leq \rho(n) \leq 2 \cdot \mathrm{E}\left[\left(\frac{\alpha}{\beta}\right)^{X_0} \Pr\left(Z \geq \frac{n + X_0}{2}\right)\right].$$

The expected values above can be written

$$\begin{aligned}\mathrm{E}\left[\left(\frac{\alpha}{\beta}\right)^{X_0} f(X_0)\right] &= \sum_{i=0}^{N} \binom{N}{i} 2^{-N} \left(\frac{\alpha}{\beta}\right)^{i} f(i) \\ &= (2\beta)^{-N} \sum_{i=0}^{N} \binom{N}{i} \alpha^i \beta^{N-i} f(i) \\ &= (2\beta)^{-N} \,\mathrm{E}\left[f(V)\right] \qquad (4.19)\end{aligned}$$

where V is Bin(N, α) distributed random variable. Consequently,

$$\frac{\mathrm{E}\left[\Pr\left(Z \geq \frac{n+V}{2}\right)\right]}{(2\beta)^N} \leq \rho(n) \leq 2 \cdot \frac{\mathrm{E}\left[\Pr\left(Z \geq \frac{n+V}{2}\right)\right]}{(2\beta)^N}.$$

Our next two steps are to apply Theorem 3 and Theorem 5 to the upper and lower bounds, respectively. The former can be done immediately. Theorem 5, however, is not valid since the random variable $(n+V)/2$ can be smaller than $n\beta$. But one can write

$$\frac{\mathrm{E}\left[\Pr\left(Z' \geq \frac{n+V}{2}\right)\right]}{(2\beta)^N} \leq \rho(n) \leq 2 \cdot \frac{\mathrm{E}\left[\frac{2\beta n}{n+V}\right]}{(2\beta)^N} .$$

where Z' is $\mathrm{Bin}(n, \alpha)$ distributed, because $\Pr\left(Z' \geq \frac{n+V}{2}\right) < \Pr\left(Z \geq \frac{n+V}{2}\right)$. Now $(n+V)/2 > n\alpha$ and so Theorem 5 can be applied to the lower bound. In the upper bound, we set V to its smallest value. This gives

$$\frac{\Pr\left(Z' \geq \frac{n+\alpha N}{2}\right)}{(2\beta)^N} \leq \rho(n) \leq 2 \cdot \frac{2\beta}{(2\beta)^N} . \tag{4.20}$$

The upper bound is now in a form whose growth rate is trivial. To deal with the lower bound we require that $\lim_{N\to\infty} \Pr\left(Z' \geq \frac{n+\alpha N}{2}\right) > 0$. This, for one, requires that the ratio of $\frac{n+\alpha N}{2}$ and the range of Z' limits to value smaller than one. In other words

$$\begin{aligned}
\lim_{N\to\infty} \Pr\left(Z' \geq \frac{n+\alpha N}{2}\right) > 0 &\iff \lim_{N\to\infty} \frac{n+\alpha N}{2n} < 1 \\
&\iff \lim_{N\to\infty} \left[1 + \frac{\alpha N}{n}\right] < 2 \\
&\iff \lim_{N\to\infty} \frac{\alpha N}{n} < 1
\end{aligned}$$

The requirement is fulfilled if $n = \eta N$ where $\eta > \alpha$. Putting this together with the condition of Eq. (4.14) we have that

$$\eta N < n < N , \quad \eta > \alpha . \tag{4.21}$$

Summing things up, assuming the condition (4.21) one can write the inequality (4.20) in a form

$$\frac{\vartheta}{(2\beta)^N} \leq \rho(n) \leq \frac{4\beta}{(2\beta)^N} \tag{4.22}$$

where $0 < \vartheta < 1$ is a constant.

By the inequality (4.22) and Eq. (4.3),

$$\frac{n}{4\beta}(2\beta)^N \leq \mathrm{E}[T] \leq \frac{n}{\vartheta}(2\beta)^N , \tag{4.23}$$

which with the condition (4.21) trivially implies

$$\mathrm{E}[T] = O\left[(2\beta)^N\right] = O\left[(2 \cdot \frac{k-1}{k})^N\right] \tag{4.24}$$

strictly.

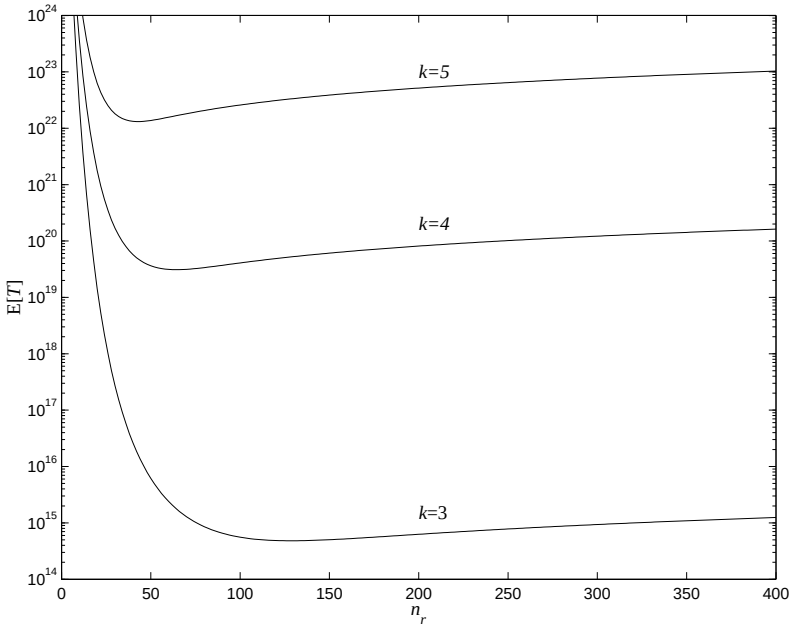


Figure 4.3 Dependence of the expected time to absorption for Markov chain $\{X_n\}$ on the restart moment n_r when $\alpha = 1/k$ and $N = 100$ (Equation (4.3)). This corresponds to dependence of the worst-case run time of the restarting RWA on the restart moment for k-SAT.

4.2 Optimal Timing

The goal of this section is to minimize the expected run time in Eq. (4.3) with respect to n_r. Figure 4.3 implies that the minimum exists and depends on k. We denote the value of n_r that yields the minimum by n_{opt}. First thing to notice in thinking about finding n_{opt} is that Eq. (4.15) is valid for $n < N$ only. Second, this expression is convoluted and of a discrete variable, so it wouldn't be very appealing for minimization anyway. Therefore one must abide by approximative results.

First of all, we approximate X_n by Y_n and thus consider the previous results valid for all $n \geq 0$. From Figure 4.3 is seen that the error of this approximation vanishes when $k \geq 4$ since $n_{opt} < N$ in those cases. Most probably, it also vanishes for $k = 3$ asymptotically as N grows.

The next approximation step involves the following theorem [7].

Theorem 6 (DeMoivre–Laplace)**.** *If X is a Bin(n, p) distributed random variable, then for a fixed real number z*

$$\lim_{n\to\infty} \Pr\left(\frac{X - np}{\sqrt{npq}} \leq z\right) = \Phi(z)$$

where $q = 1 - p$ and $\Phi(x)$ is the standard normal cumulative distribution function.

Proof. This is a special case of the central limit theorem [7, p. 244]. □

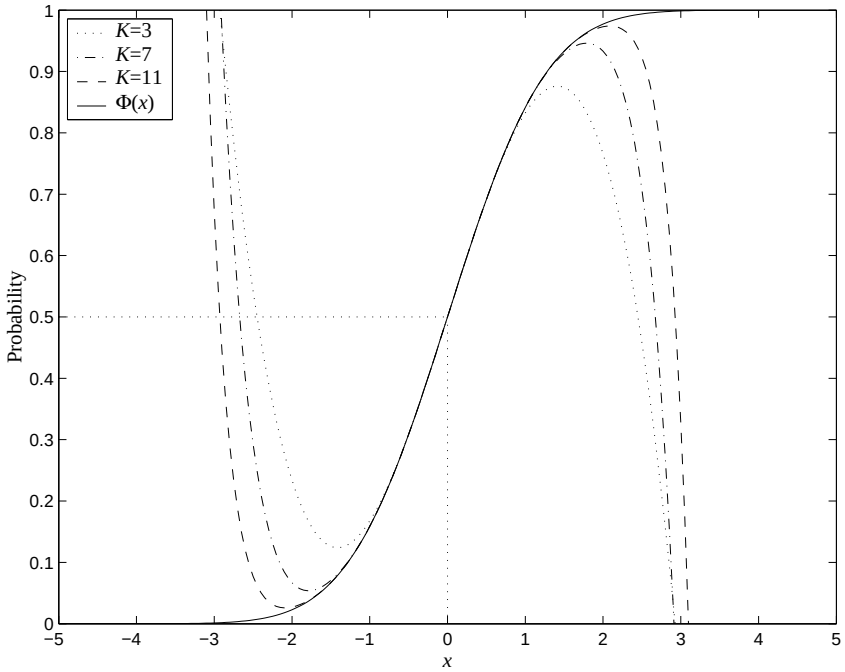


Figure 4.4 Standard normal-CDF and its Taylor polynomials of degree K

Function Φ is seen in Figure 4.4. As a corollary of Theorem 6 one obtains an approximation

$$\mathrm{Bin}_{n,p}(x) \approx \Phi\left(\frac{x-np}{\sqrt{npq}}\right), \tag{4.25}$$

the error of which is guaranteed to vanish as $n \to \infty$ if x is of the form $np + z\sqrt{npq}$ where z is constant. We apply this approximation to Eq. (4.12), which yields

$$\begin{aligned} p_{i,0}^{(n)} &\approx \Phi\left(\frac{\frac{n-i}{2} - \beta n}{\sqrt{\beta\alpha n}}\right) + \left(\frac{\alpha}{\beta}\right)^i \left[1 - \Phi\left(\frac{\frac{n+i}{2} - \beta n}{\sqrt{\beta\alpha n}}\right)\right] \\ &= \Phi\left(-\frac{(2\beta-1)n+i}{2\sqrt{\beta\alpha n}}\right) + \left(\frac{\alpha}{\beta}\right)^i \left[1 - \Phi\left(-\frac{(2\beta-1)n-i}{2\sqrt{\beta\alpha n}}\right)\right] \\ &= \Phi\left(-\frac{(\beta-\alpha)n+i}{2\sqrt{\beta\alpha n}}\right) + \left(\frac{\alpha}{\beta}\right)^i \Phi\left(\frac{(\beta-\alpha)n-i}{2\sqrt{\beta\alpha n}}\right). \end{aligned} \tag{4.26}$$

An alert reader probably noticed that the arguments of the binomial CDF in Eq. (4.12) are not of the required form. This manifests itself in the fact that the arguments of Φ in Eq. (4.26) depend on n. Consequently, nothing guarantees that the resulting error will vanish asymptotically. However, approximation (4.25) even without the precondition on x is very commonly used and proven fruitful.[1]

[1] Taylor and Karlin [35] write: "*Intuition is sometimes enhanced by the looser statement that, for large n, the sum S_n is approximately normally distributed with mean $n\mu$ and variance $n\sigma^2$. In practical terms we expect the normal distribution arise whenever the numerical outcome of an experiment results from numerous small additive effects, all operating independently, and where no single or small group of effects is dominant.*"

The argument to Φ in the second term of Eq. (4.26) is zero when $n = \frac{i}{\beta-\alpha}$. The argument in the first term, for one, is always negative and decreases limitlessly when n grows, taking the term quickly to zero as is seen from Figure 4.4. Our third approximation step to drop the first term in Eq. (4.26). Also use Eq. (3.12) to get

$$\rho(n) \approx \mathrm{E}\left[\left(\frac{\alpha}{\beta}\right)^{X_0} \Phi\left(\frac{(\beta-\alpha)n - X_0}{2\sqrt{\beta\alpha n}}\right)\right]. \tag{4.27}$$

Now Eq. (4.19) can be applied to this, yielding

$$\rho(n) \approx (2\beta)^{-N}\,\mathrm{E}\left[\Phi\left(\frac{(\beta-\alpha)n - V}{2\sqrt{\beta\alpha n}}\right)\right]. \tag{4.28}$$

where V is $\mathrm{Bin}(N, \alpha)$ distributed.

Two problems still remain in Eq. (4.28): How to deal with the expected value and function Φ, for which no closed form expression exists. Fortunately, using Taylor series offers a solution to both problems.

The standard normal-CDF has a Taylor series representation [8]

$$\Phi(x) = \frac{1}{2} + \frac{1}{\sqrt{2\pi}} \sum_{k=0}^{\infty} \frac{(-1)^k}{2^k(2k+1)k!} x^{2k+1}. \tag{4.29}$$

Truncating the series above results a Taylor polynomial $b_0 + b_1 x + \ldots + b_R x^R$. This is plotted in Figure 4.4. As a fourth step, we approximate Eq. (4.28) with a form

$$\begin{aligned}
\rho(n) &\approx (2\beta)^{-N}\,\mathrm{E}\left[\sum_{k=0}^{R} b_k \left(\frac{(\beta-\alpha)n - Z}{2\sqrt{\beta\alpha n}}\right)^k\right] \\
&= (2\beta)^{-N}\,\mathrm{E}\left[\sum_{k=0}^{R} b_k \left(\frac{1}{2\sqrt{\beta\alpha n}}\right)^k \sum_{l=0}^{k} \binom{k}{l} \left[(\beta-\alpha)n\right]^{k-l} (-Z)^l\right] \\
&= (2\beta)^{-N} \sum_{k=0}^{R} b_k \left(\frac{1}{2\sqrt{\beta\alpha n}}\right)^k \sum_{l=0}^{k} \binom{k}{l} \left[(\beta-\alpha)n\right]^{k-l} (-1)^l\,\mathrm{E}\left[Z^l\right] \\
&= (2\beta)^{-N} \sum_{k=0}^{R} \sum_{l=0}^{k} \frac{b_k}{\left(2\sqrt{\beta\alpha}\right)^k} \binom{k}{l} (-1)^l (\beta-\alpha)^{k-l} \mu_l' n^{\frac{k}{2}-l}
\end{aligned} \tag{4.30}$$

where $\mu_l' = \mathrm{E}\left[Z^l\right]$ is the lth moment of random variable Z.

At first glance, comparing Eq. (4.30) and (4.14) seems like getting out of the frying pan into the fire. However, the effort of evaluating Eq. (4.30) is not only constant with respect to N but also small—we will see shortly that as small as $R = 3$ gives useful results.

As a fifth, and last, approximation step we utilize Theorem 6 again and replace the binomial moments μ_l' by the corresponding normal moments, i.e. moments of a $\mathrm{Norm}(\alpha N, \sqrt{\alpha\beta N})$ distributed random variable; see Section A.2. The following example will clarify all this.

Example 2. Calculate an approximate for the optimal restart moment in case of 3-SAT and $N = 200$ by using Eq. (4.30) with $K = 3$.

Solution. First we have to calculate the Taylor coefficients b_k and the moments $\mu_k^{\cdot}$, $k = 0, \ldots, 3$. By Eq. (4.29)

$$\begin{aligned} b_0 &= \frac{1}{2}\,, \\ b_1 &= \frac{1}{\sqrt{2\pi}}\frac{1}{1\cdot 1\cdot 1} = \frac{1}{\sqrt{2\pi}}\,, \\ b_2 &= 0\,, \\ b_3 &= \frac{1}{\sqrt{2\pi}}\frac{-1}{2\cdot 3\cdot 1} = -\frac{1}{6\sqrt{2\pi}}\,. \end{aligned}$$

Substituting $\mu = \alpha N$ and $\sigma = \sqrt{\alpha\beta N}$ to the moments calculated in section A.2 gives

$$\begin{aligned} \mu_0^{\cdot} &= 1\,, \\ \mu_1^{\cdot} &= \alpha N\,, \\ \mu_2^{\cdot} &= (\alpha N)^2 + \alpha\beta N\,, \\ \mu_3^{\cdot} &= (\alpha N)^3 + 3\beta(\alpha N)^2\,. \end{aligned}$$

Substituting these into Eq. (4.30) with $N = 200$, $\alpha = 1/3$, and $\beta = 2/3$ yields

$$\begin{aligned} \rho(n) &\approx \left(\frac{3}{4}\right)^{200} \sum_{k=0}^{3}\sum_{l=0}^{k} \frac{b_k}{\left(\frac{2\sqrt{2}}{3}\right)^k}\binom{k}{l}(-1)^l \left(\frac{1}{3}\right)^{k-l} \mu_l^{\cdot} n^{\frac{k}{2}-l} \\ &= \left(\frac{3}{4}\right)^{200}\left(\frac{128750}{3\sqrt{\pi}}n^{-\frac{3}{2}} - \frac{2725}{4\sqrt{\pi}}n^{-\frac{1}{2}} + \frac{1}{2} + \frac{27}{8\sqrt{\pi}}n^{\frac{1}{2}} - \frac{1}{192\sqrt{\pi}}n^{\frac{3}{2}}\right)\,. \end{aligned}$$

Function $n/\rho(n)$ as above is plotted in Figure 4.5. From the figure is seen that the desired value is found at the second local extremum greater than zero.

Notice that a minimum of $n/\rho(n)$ is maximum of $\rho(n)/n$ and vice versa. Derivative of $\rho(n)/n$ is

$$\left(\frac{3}{4}\right)^{200}\frac{\mathrm{d}}{\mathrm{d}n}\left[\frac{128750}{3\sqrt{\pi}}n^{-\frac{5}{2}} - \frac{2725}{4\sqrt{\pi}}n^{-\frac{3}{2}} + \frac{1}{2}n^{-1} + \frac{27}{8\sqrt{\pi}}n^{-\frac{1}{2}} - \frac{1}{192\sqrt{\pi}}n^{\frac{1}{2}}\right] = \left(\frac{3}{4}\right)^{200}\frac{-\frac{1}{384}n^3 - \frac{27}{16}n^2 - \frac{\sqrt{\pi}}{2}n^{3/2} + \frac{8175}{8}n - \frac{321875}{3}}{\sqrt{\pi}n^{7/2}}\,.$$

Hence, we have to find the roots of the expression

$$f(n) = \frac{1}{384}n^3 + \frac{27}{16}n^2 + \frac{\sqrt{\pi}}{2}n^{\frac{3}{2}} - \frac{8175}{8}n + \frac{321875}{3}\,. \tag{4.31}$$

From Figure 4.6 is seen that this has two real solutions. They are $n \approx 157.98$ and $n \approx 245.27$. The answer is 245. ♢

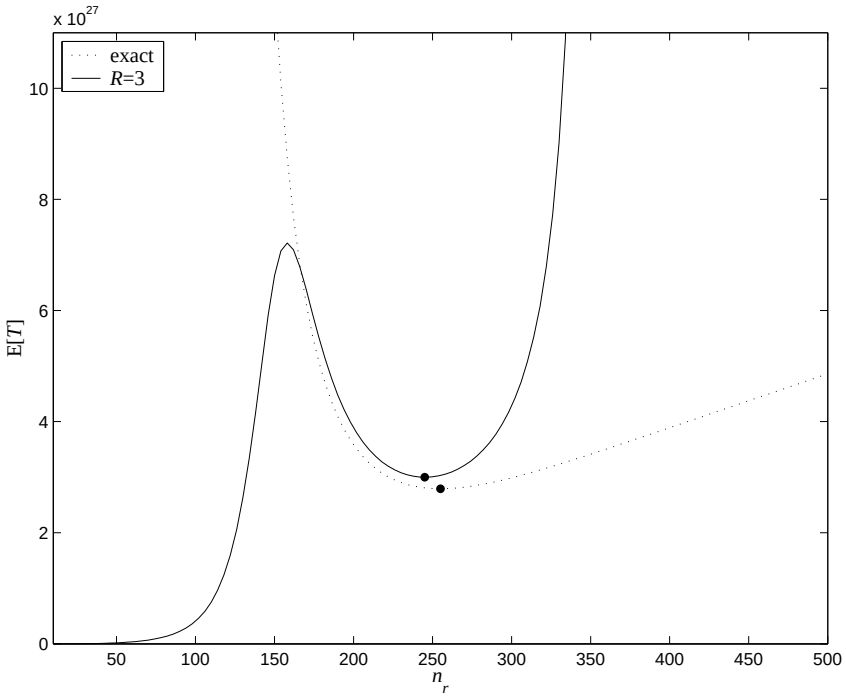


Figure 4.5 Minimizable approximation of the expected time to absorption when $\alpha = 1/3$ and $N = 200$. The minimum is marked with a dot.

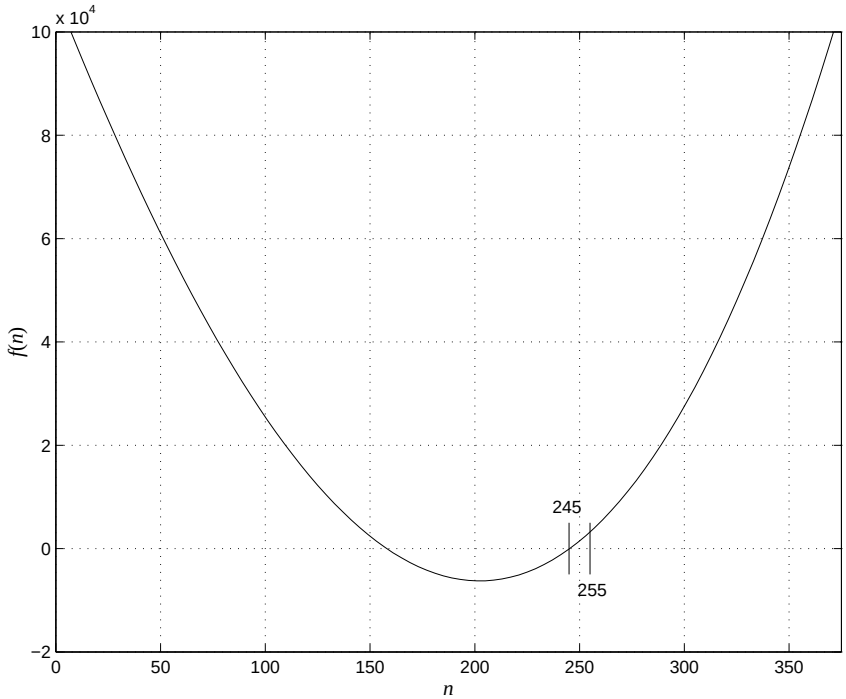
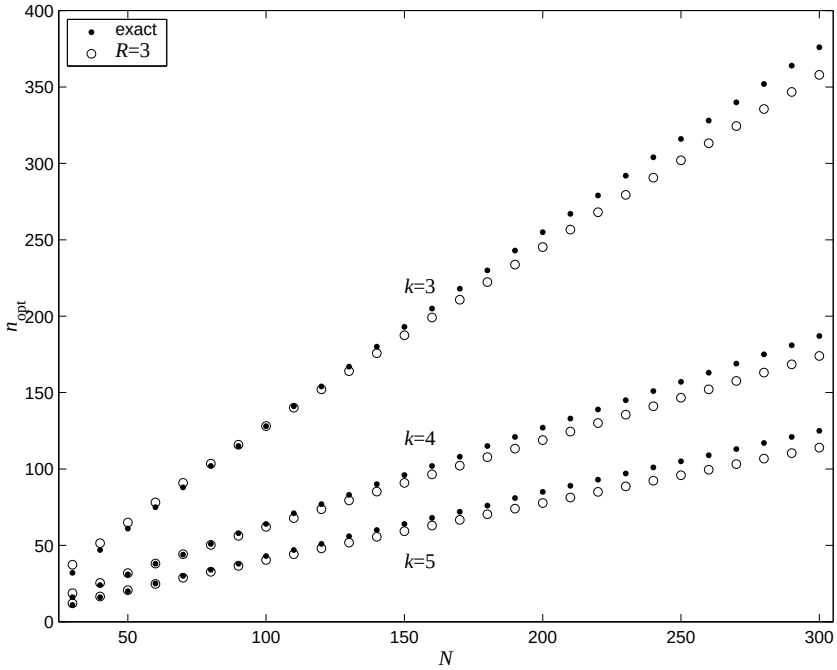
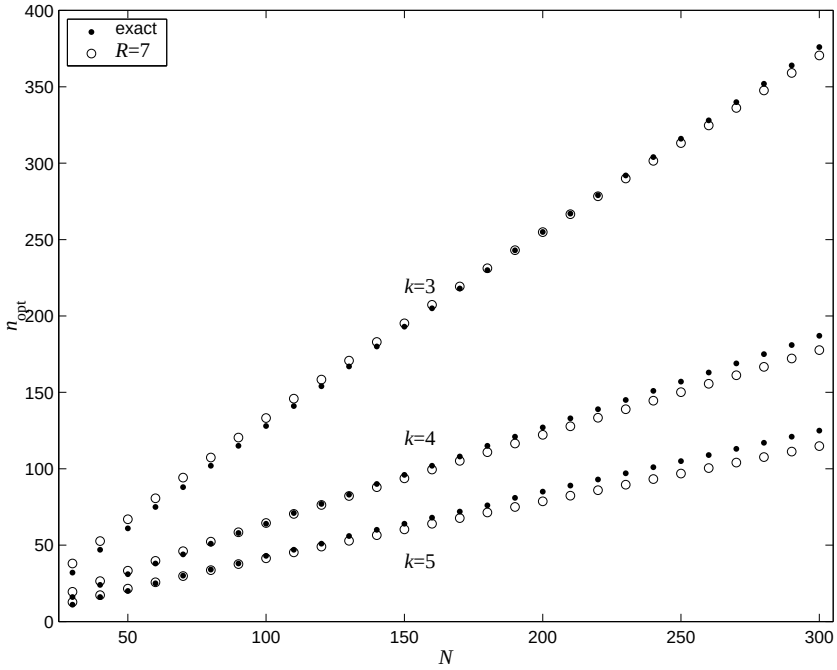


Figure 4.6 Function (4.31) whose second positive root represents an approximate to $n_{opt} = 255$ for $\alpha = 1/3$ and $N = 200$



(a)



(b)

Figure 4.7 Optimal restart moment; exact values and the approximations involving Taylor polynomial of degree R

In the previous example we maximized $\rho(n)/n$ instead of minimizing $n/\rho(n)$. This is reasonable in general since the latter expression has poles in the domain of interest. The former, for one, leads to a polynomial-like expression that is differentiable everywhere and an easy bite for numerical root-finding.

A wider picture of the accuracy of this approximation is obtained from Figure 4.7. The figure suggests that the dependence between n_{opt} and N is linear and that the approximative values are little smaller than the correct ones. Hence the approximative values should be multiplied with something like 1.1. Notice from Figure 4.3 that having the restart moment greater than the optimal value is better than having it smaller.

On the grounds of Figure 4.7, we conclude that n_{opt} fulfills the condition (4.21) for $k \geq 4$. Then Eq. (4.24) is valid providing an answer to the questions 1 and 3: The growth rate with the optimal restart moment is strictly $O\left[(2 \cdot \frac{k-1}{k})^N\right]$. For $k = 3$ Figure 4.7 implies that $N < n_{opt} < 2N$. In this case one has to rely on the following conjecture.

Conjecture 1. $\Pr(X_n = 0)$ and $\Pr(Y_n = 0)$ with $\alpha = \frac{1}{3}$ are polynomially related in terms of N, if $0 \leq n \leq 2N$.

Foundation. The conjecture is true for $0 \leq n < N$ since $\Pr(X_n = 0) = \Pr(Y_n = 0)$ in that case. For $N \leq n \leq 2N$ it is true if $\Pr(X_{2N} = 0)$ and $\Pr(Y_{2N} = 0)$ are polynomially related. Function $f(N) = \frac{\Pr(X_{2N}=0)}{\Pr(Y_{2N}=0)}$ is plotted in Figure 4.8, which strongly suggests that this is the case.

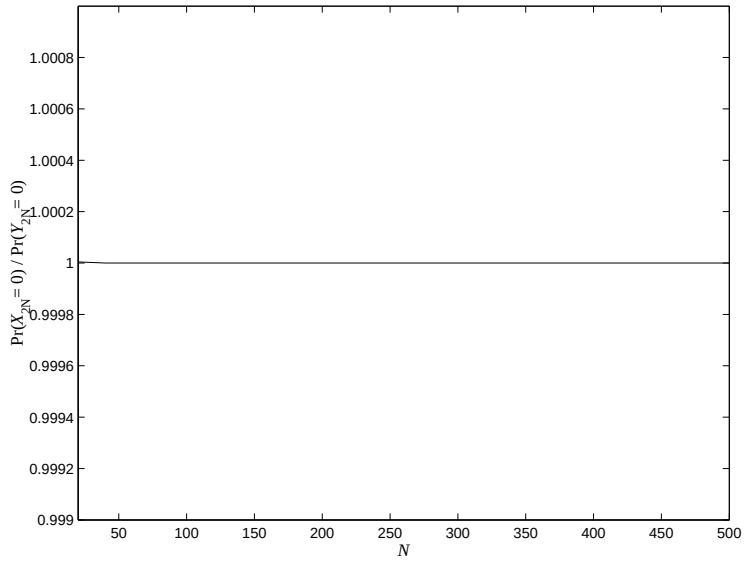


Figure 4.8 Ratio of $\Pr(X_{2N} = 0)$ and $\Pr(Y_{2N} = 0)$, which are the $2N$-step absorption probabilities of Markov chains $\{X_n\}$ (Figure 3.1) and $\{Y_n\}$ (Figure 4.2), respectively

Conjecture 1 implies that extending condition (4.21) to cover n_{opt} for $k = 3$ results in a polynomial factor in the bounds of inequality (4.22), which does not change Eq. (4.24).

5 Implementation Issues

Implementing the random-walk algorithm is straightforward. This chapter shortly discusses the issue, especially how to exploit the ideas of this work in practice. Also a new RWA variant, which might be useful in practice when there is absence of true randomness, is proposed in Section 5.2.

5.1 Exploiting the Ideas

Implementations usually have the restart moment as a parameter set by the user. Example 2 provides a procedure to calculate a setting that minimizes the expected worst-case run time. This procedure should be straightforward to implement. It involves solving an expression corresponding to Eq. (4.31) numerically with e.g. the Newton's method. Using a numerical optimization algorithm works as well. Be that as it may, a starting point is needed. We will find such next.

Recall that the argument of Φ in Eq. (4.31) is zero when $n = \frac{V}{\beta-\alpha}$. expected value of this is

$$\begin{aligned} n &= \frac{\alpha}{\beta-\alpha}N \\ &= \frac{N}{k-2} \end{aligned} \tag{5.1}$$

This point is special since it's the inflection point of function Φ. It turns out that this serves as a rough approximate to n_{opt}.

A real-life SAT instance is hardly ever a clean k-SAT, but has variable length clauses. This is not a problem. Notice that the only assumption about $\alpha = \frac{1}{k}$ made in Chapter 4 was $\alpha \neq \frac{1}{2}$. That is to say, setting e.g. $k = 2.31$ or $k = 5.94$ is possible indeed. This suggests calculating the average clause length and setting k accordingly. In addition, probably one should not select the unsatisfied clause at random, but select the shortest clause since this way the α at each step is maximized.

5.2 Complement Checking: A Second Try

Here the idea of complement checking is carried one step further. Take a look at Algorithm 4. The difference between this and the normal RWA is that two branches are traced interleaved. Again a random initial assignment a is drawn in the beginning, but then an additional branch is initiated by assignment $\hat{a}$ which is the complement, $\hat{a} = \overline{a}$. Then these two branches are traced independently. We will call this complement tracing RWA (CTRWA).

Algorithm 4 Complement tracing RWA

```
1: procedure ctSearch(F, K)
2:     a ← randomly chosen truth assignment
3:     â ← ā
4:     for repeat K times do
5:         if F(a) = true then
6:             return a
7:         else
8:             c ← randomly chosen unsatisfied clause in F(a)
9:             s ← randomly chosen atom of c
10:            a ← a except s flipped
11:        end if
12:        if F(â) = true then
13:            return â
14:        else
15:            c ← randomly chosen unsatisfied clause in F(â)
16:            s ← randomly chosen atom of c
17:            â ← â except s flipped
18:        end if
19:    end for
20:    return false
21: end procedure
```

An appropriate measure of the algorithm state after the n:th *for* cycle would be $\hat{X}_n = \min[\mathrm{d}(a, a^*), \mathrm{d}(\hat{a}, a^*)]$, where a^* is the model. The idea in CTRWA is that $\Pr(\hat{X}_0 \leq \frac{1}{2}) = 1$ whereas for two independent starts it would be only 3/4. Since one *for* cycle in Algorithm 4 takes two flips, the n-step absorption probability $\hat{\rho}(n)$ is

$$\begin{aligned}
\hat{\rho}(2n) &= \sum_{i=0}^{N} \omega_i [1 - (1 - p_{i,0}^{(n)})(1 - p_{N-i,0}^{(n)})] \\
&= \sum_{i=0}^{N} \omega_i \left[1 - \left(1 - p_{N-i,0}^{(n)} - p_{i,0}^{(n)} + p_{i,0}^{(n)} p_{N-i,0}^{(n)}\right)\right] \\
&= \sum_{i=0}^{N} \omega_i p_{N-i,0}^{(n)} + \sum_{i=0}^{N} \omega_i p_{i,0}^{(n)} - \sum_{i=0}^{N} \omega_i p_{i,0}^{(n)} p_{N-i,0}^{(n)} \quad |\omega_i = \omega_{N-i} \\
&= \sum_{i=0}^{N} \omega_i p_{i,0}^{(n)} + \sum_{i=0}^{N} \omega_i p_{i,0}^{(n)} - \sum_{i=0}^{N} \omega_i p_{i,0}^{(n)} p_{N-i,0}^{(n)} \\
&= 2\rho(n) - \sum_{i=0}^{N} \omega_i p_{i,0}^{(n)} p_{N-i,0}^{(n)} \,. \qquad (5.2)
\end{aligned}$$

Function $\hat{\rho}(n)$ is, after a small threshold, greater than $\rho(n)$, meaning the CTRWA performs better if no restarts are present. However, the difference vanishes as N grows. With restarts the situation is the opposite. Namely, if the algorithm is

restarted every $\hat{n}_r$ cycles, i.e. every $2\hat{n}_r$ flip, the expected run time is

$$\begin{aligned} \mathrm{E}[T] &= \frac{2\hat{n}_r}{\hat{\rho}(2\hat{n}_r)} \quad | \text{ Eq. (5.2)} \\ &> \frac{2\hat{n}_r}{2\rho(\hat{n}_r)} = \frac{\hat{n}_r}{\rho(\hat{n}_r)} \end{aligned} \tag{5.3}$$

where the difference between the right and the left side of the inequality vanishes as N grows. This is because the positive term in Eq. (5.2) dominates the negative one. Hence, the RWA and CTRWA are asymptotically equivalent.

There is, however, another aspect supporting the CTRWA. Namely producing true randomness is very difficult, and in practice one has to abide by pseudo random numbers. As a result, the distribution of X_0 is perturbed with $\mathrm{E}[X_0] = N/2 + err$. If err happens to be positive, this decreases the performance. Using the complement tracing scheme, however, makes the algorithm immune to a bias in $\mathrm{E}[X_0]$.

6 Experiments

In this chapter the random-walk algorithm is studied empirically. We run a RWA implementation upon some problem instances and compare the observed behavior to the one predicted by the stochastic model developed in Section 3.1.

6.1 Description

Methodology

Instead of measuring run times with different restart moments, we utilize a more elegant scheme similar to what is used in [14]. Run times are recorded in flips instead of CPU seconds since this makes the experiments independent of the underlying machinery and, thus, reduces error sources as well as makes the results more comparable.

The total number of flips in a successful run is $T = n_r(S-1)+L$ where n_r is the restart moment, S is the number tries ($S-1$ is the number of restarts), and L the number of flips since the last restart. The expected value and the variance of T are thus

$$\begin{aligned} \mathrm{E}[T] &= \mathrm{E}[n_r(S-1)+L] \\ &= n_r(\mathrm{E}[S]-1)+\mathrm{E}[L] \end{aligned} \tag{6.1}$$

and

$$\begin{aligned} \mathrm{Var}[T] &= \mathrm{Var}[n_r(S-1)+L] \\ &= \mathrm{Var}[n_r(S-1)] + \mathrm{Var}[L] + 2\,\mathrm{Cov}[n_r(S-1), L] \\ &= n_r^2\,\mathrm{Var}[S] + \mathrm{Var}[L] + 2n_r\,\mathrm{Cov}[S, L], \end{aligned} \tag{6.2}$$

respectively, where appropriate calculation rules [29, p. 413] were used. Parameters $\mathrm{E}[S]$ and $\mathrm{Var}[S]$, $\mathrm{E}[L]$ and $\mathrm{Var}[L]$, and $\mathrm{Cov}[S, L]$ (covariance) are determined by experiments.

In order to obtain random samples from random variables S and L, the algorithm is run with a restart moment R and a time-out K until it finds a model (K sufficiently large), and the values of S and L are recorded. This is repeated M times. Suppose values $s_1, \ldots, s_M$ and $l_1, \ldots, l_M$ are observed. These sequences represent random samples of random variables S and L, respectively, in the case of $n_r = R$. The sample size is M. Nonetheless, samples for cases $n_r < R$ can be extracted from this same data. Suppose values $s_1 = 931$, $s_2 = 586$ and $l_1 = 311$, $l_2 = 147$ are observed when $R = 400$. Notice that the outcome would have been the same even if R had been 399 or anything between 311 and 400. Hence, these are valid samples for $311 \leq n_r \leq 400$. If R had been between 147 and 310, there had been one successful

Table 6.1 Problem instances used in the experiments

name	N	k	clauses	solutions
aim-50-1_6-yes1-2	50	3	80	1
aim-50-1_6-yes1-4	50	3	80	1
aim-50-2_0-yes1-1	50	3	100	1
G3-50-250-1	50	3	250	1

run with $s_1 = 931 + 586$ and $l_1 = 147$. Hence the data contains one sample pair, $s_1 = 1517$ and $l_1 = 147$, for $147 \leq n_r \leq 310$. Case $n_r < 147$ cannot be addressed by this data.

The previous procedure generalizes as follows. Suppose that samples $s_1, \ldots, s_M$ and $l_1, \ldots, l_M$ are observed with a setting $n_r = R$. For each $n_r \leq R$ one obtains samples $\hat{s}_1, \ldots, \hat{s}_m$ and $\hat{l}_1, \ldots, \hat{l}_m$ by first forming a sequence a which enumerates the indexes at which $l_i \leq n_r$, that is $a = \{i \,|\, l_i \leq n_r\}$. Then $\hat{s}_1 = \sum_{i=1}^{a_1} s_i$, $\hat{s}_j = \sum_{i=a_{j-1}+1}^{a_j} s_i$, $j = 2, \ldots, m$, and $\hat{l}_j = l_{a_j}$. The sample size is $m = |a|$. Now one can calculate the expected value and the variance of S and L, $\mathrm{Cov}[S, L]$, and thus quantities (6.1) and (6.2) for each $n_r \leq R$.

Problem Instances

In order to verify the previous calculations by experiments, one needs also problem instances that are as close to the theoretical worst case as possible. Hence we are seeking problem instances having a unique solution. Such are found in the AIM instance set of the DIMACS benchmark collection [15] and a set provided by M. Motoki and O. Watanabe as a sample of their unique solution instance generator [19]. We confine ourselves to the case $N = 50$ since neither of the second smallest cases available in these sets ($N = 100$, $N = 75$) could be solved in a reasonable time. In addition, only 3-SAT instances are available. From the AIM subset having $N = 50$, three instances that describe well the whole subset are presented here. The selected instances are listed in Table 6.1.

Implementation

As an implementation served Walksat version 45 [17], which has a parameter `-random` for selecting a pure random-walk heuristic.

6.2 Results

The tests were run on a PC with 2.4 GHz CPU and 1 GB RAM. The program reported around $1.5 \cdot 10^6$ flips per second tops, which means 11.3 seconds for the hardest of the problem instances in Table 6.1. For each instance, the expected run time as a function of restart moment is calculated by Eq. (6.1) and the result is plotted in Figure 6.1. The minimum point of this function as well as the standard

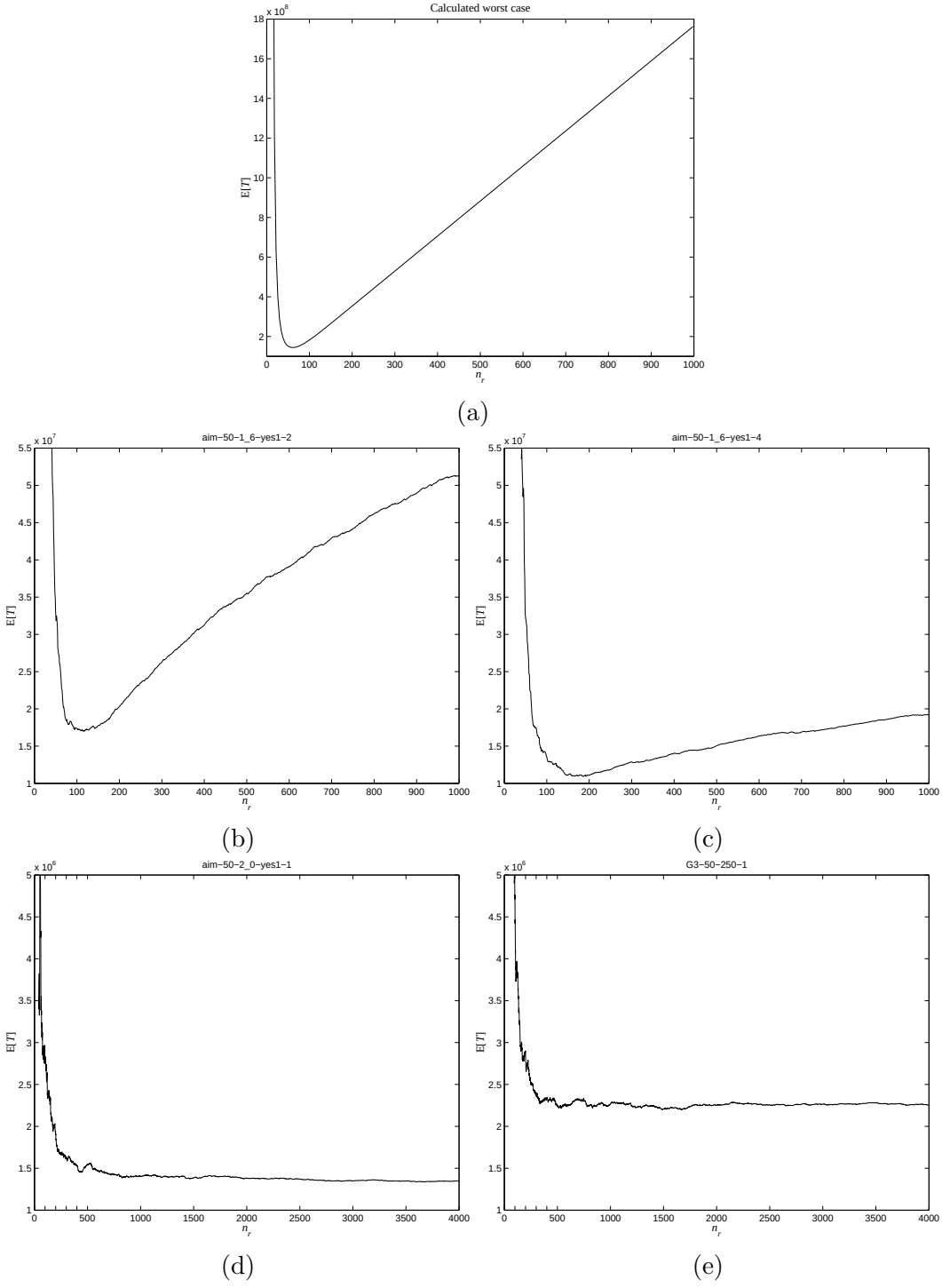


Figure 6.1 Expected run time versus restart moment calculated (a) and determined by experiments (b-e) in case of $N = 50$ and $k = 3$.

Table 6.2 Experiment results

Setting			Results			
instance	R	M	n_{opt}	$\mathrm{E}[T_{opt}]$	$\sqrt{\mathrm{Var}[T_{opt}]}$	Fig.
aim-50-1_6-yes1-2	1000	1000	114	$1.7009 \cdot 10^7$	$1.6992 \cdot 10^7$	6.1b
aim-50-1_6-yes1-4	1000	1000	185	$1.0928 \cdot 10^7$	$1.1288 \cdot 10^7$	6.1c
aim-50-2_0-yes1-1	4000	2000	3681	$1.3380 \cdot 10^6$	$1.4092 \cdot 10^6$	6.1d
G3-50-250-1	4000	2000	1666	$2.1975 \cdot 10^6$	$2.2171 \cdot 10^6$	6.1e
calculated worst case			61	$1.4423 \cdot 10^8$	$1.4423 \cdot 10^8$	6.1a

deviation at this point are seen in Table 6.2. The same quantities calculated by Eq. (4.3), (4.4) and (3.13) are included for verification.

According to Table 6.2, the greatest expected run time observed (aim-50-1_6-yes1-2) is almost thirteen times greater than the smallest one (aim-50-2_0-yes1-1), so there is great variations in hardness of solving with RWA among different instances even if they have the same clause length and the same number of solutions. These variations are probably difficult to determine a priori. From Figure 6.1 is seen that the greater the runtime is, the closer to the calculated worst case is the curve shape. The instance that was observed as the hardest shows behavior (Figure 6.1b) pretty similar to the calculated worst case (Figure 6.1a), although the run time of the latter is still about eight times greater than that of the former. The standard deviation, for one, seems what the theory predicts being about equal to the run time.

The observed optimal restart moment depends heavily on the observed instance hardness. The hardest has it still somewhat larger that the calculated worst case, which is no wonder considering the difference in their run times. The performance for the easiest instance seems very robust with respect to the restart moment as long as it is greater than about $10N$. Notice that even if the restart moment is set according to the hardest instance, $n_r = 114$, the easiest still remains pretty easy having $\mathrm{E}[T] \approx 2.5 \cdot 10^6$. Also the rest of the four AIM instances having $N = 50$ and 80 clauses were tested, but their curves were very similar to those in Figures 6.1b and 6.1c.

In the light of this evidence, the experimental findings support the used stochastic model but suggest that the theoretical worst case is too pessimistic in practice. This is no wonder recalling the fact that the analysis is based on assuming the probability of a successful flip being $1/k$. But since we are talking about millions of flips, the worst-case scenario is unrealizable in practice. Making conjectures about a practical worst case or an average case would require more extensive experiments though.

7 Conclusions

An important aspect in understanding the random-walk algorithm for SAT is the fact that its capability rely heavily on restarts. Even though an existing model is found eventually with or without restarts (Theorem 2), without them the algorithm is reasonable, i.e. faster than just enumerating all the possibilities, for 2-SAT only (Section 3.2). For k-SAT, $k \geq 3$, if n_{opt} denotes the restart moment that minimizes the expected worst-case run time $\mathrm{E}[T]$, then n_{opt} is of the same order than N, which is the number of variables.

Key results of this work are the following two: n_{opt} is solved approximately (Section 4.2), and $\mathrm{E}[T]$ with n_{opt} is shown to be $O\left[(2\frac{k-1}{k})^N\right]$ strictly (Definition 4 on page 8) for the exponential part (Section 4.1.2). The former should contribute directly to actual implementations, which have lacked a method for determining a good restart moment parameter [14, p. 32]. The latter extends the result of Schöning [30, 31], who showed this upper bound for a specific non-optimal restart moment. Papadimitriou's [21] result $\mathrm{E}[T] = O(N^2)$ for 2-SAT is confirmed too (Section 3.2). The standard deviation is shown to be almost as large as $\mathrm{E}[T]$ (p. 16), which explains the noticeable chance fluctuations in the run time.

From the fact that $n_{opt} = O(N)$ follows that the number of restarts needed is $O\left[(2\frac{k-1}{k})^N\right]$ too. How this should be interpreted? Well, it behooves one to seek a nearly satisfying initial assignment rather than trusting the ability of the actual search to turn an arbitrary assignment into a model (although it is capable of that). If n_{opt} steps were taken without success, the initial assignment obviously wasn't good enough and a new one is drawn by restarting the search. Improving the algorithm thus involves improving either the initial assignment selection or the search itself (or both of course). A scheme for the former is presented and analyzed in [13]. The Walksat procedure [32] does the latter. More sophisticated search probably causes the optimal restart moment being bigger.

The stochastic model of the algorithm, on which the analysis is based on, is a Markov chain (Section 3.1). Theory of Markov chains provides means to calculate exact values of $\mathrm{E}[T]$, n_{opt}, and other quantities for a fixed N and k by utilizing matrices. The approximations needed for addressing the general case can be verified this way. This scheme, however, becomes laborious for a large N as it involves calculating powers of an $N + 1 \times N + 1$ matrix. Cases N up to about 500 can be addressed by a 2.2 GHz workstation. Matlab code for doing this is provided in Appendix B.

The experimental findings (Chapter 6) support the developed stochastic model but suggest that the theoretical worst case is too pessimistic in practice, as is usual in algorithm analysis. Consequently, the optimal restart moment is somewhat bigger. Being more explicit requires more experiments. A subject for a further study could be analyzing the search with problem–solution pairs step by step recording

the success probability α at each flip. The aim would be determining such constant α that fits best in the findings. As long as α is constant throughout the search, one obtains the optimal restart moment by calculating n_{opt} with this value.

Another thing worth studying might be whether there exists a heuristic method of deciding with high probability, given a truth assignment a, which one, a or its complement $\overline{a}$, is closer to a model. This knowledge would halve the run time since tracing both branches of the complement tracing RWA (Section 5.2) would be no longer necessary. Counting the number of satisfied clauses might be a candidate.

Bibliography

[1] Sven Baumer and Rainer Schuler. Improving a probabilistic 3-SAT Algorithm by Dynamic Search and Independent Clause Pairs. *Electronic Colloquium on Computational Complexity*, Report No. 10, 2003. http://eccc.uni-trier.de/eccc/.

[2] A. Biere, A. Cimatti, E. Clarke, M. Fujita, and Y. Zhu. Symbolic Model Checking Using SAT Procedures instead of BDDs. In *Proceedings of the 36th Design Automation Conference (DAC)*, pages 317–320. ACM/IEEE, ACM, 1999.

[3] A. Biere, E. Clarke, R. Raimi, and Y. Zhu. Verifying Safety Properties of a PowerPC Microprocessor Using Symbolic Model Checking without BDDs. In *11th International Conference on Computer Aided Verification (CAV)*, number 1633 in Lecture Notes in Computer Science, pages 60–71. Springer–Verlag, 1999.

[4] D. R. Cox and H. D. Miller. *The Theory of Stochastic Processes*. Chapman and Hall, 1978.

[5] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem proving. *Communications of the ACM*, 5:394–397, 1962.

[6] N. G. de Bruijn. *Asymptotic Methods in Analysis*. North-Holland Publishing Company, third edition, 1970.

[7] William Feller. *An Introduction to Probability Theory and Its Applications*, volume 1. John Wiley & Sons, third edition, 1968.

[8] M. Fogiel, editor. *Handbook of Mathematical, Scientific and Engineering Formulas, Tables, Functions, Graphs, Transforms*. Staff of Research and Education Association, 1988.

[9] John Franco. Some interesting research directions in satisfiability. *Annals of Mathematics and Artificial Intelligence*, 28:7–15, 2000.

[10] Ian P. Gent and Toby Walsh. The search for Satisfaction. Internal report, Department of Computer Science, University of Strathclyde, 1999.

[11] Carla P. Gomes, Bart Selman, and Henry Kautz. Boosting Combinatorial Search Through Randomization. In *Proceedings of the 15th National Conference on Artificial Intelligence*, pages 431–437. American Association for Artificial Intelligence (AAAI), 1998.

[12] Jun Gu, Paul W. Purdom, John Franco, and Benjamin W. Wah. Algorithms for the Satisfiability (SAT) Problem: A Survey. In *Satisfiability Problem: Theory and Applications*, volume 35 of *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science*, pages 19–151. American Mathematical Society, 1997.

[13] Thomas Hofmeister, Uwe Schöning, Rainer Schuler, and Osamu Watanabe. A probabilistic 3-SAT algorithm further improved. In *Proceedings of the 19th Symposium on Theoretical Aspects of Computer Science (STACS)*, number 2285 in Lecture Notes in Computer Science, pages 192–202. Springer–Verlag, 2002.

[14] Holger H. Hoos and Thomas Stützle. Local Search Algorithms for SAT: An Empirical Evaluation. *Journal of Automated Reasoning*, 24(4):421–481, 2000.

[15] Holger H. Hoos and Thomas Stützle. SATLIB: An Online Resource for Research on SAT. In Ian P. Gent, Hans van Maaren, and Toby Walsh, editors, *SAT 2000*, pages 283–292. IOS Press, 2000. Available online at www.satlib.org.

[16] Kazuo Iwama and Suguru Tamaki. Improved Upper Bounds for 3-SAT. *Electronic Colloquium on Computational Complexity*, Report No. 53, 2003. http://eccc.uni-trier.de/eccc/.

[17] Henry Kautz. Walksat version 45, 2004. A computer program available at http://www.cs.washington.edu/homes/kautz/walksat/ (August 5 2004).

[18] Inês Lynce and João Marques-Silva. Building State-of-the-Art SAT Solvers. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*. IOS Press, 2002.

[19] M. Motoki and O. Watanabe. Sample Instances, 1999. Files available at http://www.is.titech.ac.jp/˜watanabe/gensat/a1/#sample (August 5 2004).

[20] Anil Nerode and Richard A. Shore. *Logic for Applications*. Springer, second edition, 1997.

[21] Christos H. Papadimitriou. On selecting a satisfying truth assignment. In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 163–169. IEEE, 1991.

[22] Christos H. Papadimitriou. *Computational Complexity*. Addison–Wesley, 1994.

[23] Emanuel Parzen. *Stochastic processes*. Holden–Day, 1962.

[24] R. Paturi, P. Pudlàck, M. E. Saks, and F. Zane. An improved exponential-time algorithm for k-SAT. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 628–637. IEEE, 1998.

[25] N. U. Prabhu. *Stochastic Processes*. The Macmillan Company, 1965.

[26] Sheldon M. Ross. *Stochastic Processes*. John Wiley & Sons, second edition, 1996.

[27] Sheldon M. Ross. *Introduction to Probability Models*. Academic Press, sixth edition, 1997.

[28] Sheldon M. Ross. *Introduction to Probability Models*. Academic Press, eighth edition, 2003.

[29] Lennart Råde and Bertil Westergren. *Mathematics Handbook for Science and Engineering*. Studentlitteratur, fourth edition, 1998.

[30] Uwe Schöning. A Probabilistic Algorithm for k-SAT and Constraint Satisfaction Problems. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 410–414. IEEE, 1999.

[31] Uwe Schöning. A Probabilistic Algorithm for k-SAT Based on Limited Local Search and Restart. *Algoritmica*, 32:615–623, 2002.

[32] Bart Selman, Henry Kautz, and Bram Cohen. Noise Strategies for Improving Local Search. In *Proceedings of the 12th National Conference on Artificial Intelligence*, pages 337–343. American Association for Artificial Intelligence (AAAI), 1994.

[33] Bart Selman, Hector Levesque, and David Mitchell. A New Method for Solving Hard Satisfiability Problems. In *Proceedings of the 10th National Conference on Artificial Intelligence*, pages 440–446. American Association for Artificial Intelligence (AAAI), 1992.

[34] Eugene Seneta. *Non-negative matrices and Markov chains*. Springer, New York, second edition, 1981.

[35] Howard M. Taylor. *An Introduction to Stochastic Modeling*. Academic Press, third edition, 1998.

[36] B. Weesakul. The Random Walk Between a Reflecting and an Absorbing Barrier. *The Annals of Mathematical Statistics*, 32(3):765–769, 1961.

Appendix A

A.1 Schöning's Expression

Equality between Eq. (4.12) and Schöning's equivalent [31, p. 4] is shown in this section.

If the distribution of the first passage time to zero is known, one can calculate the n-step absorption probabilities by the identity

$$p_{i,0}^{(n)} = \sum_{k=0}^{n} \Pr(T = k | X_0 = i)\,. \tag{A.1}$$

Since state zero is absorbing we have that

$$\begin{aligned}
\Pr(T = n | X_0 = i) &= \alpha p_{i,1}^{(n-1)} \quad \mid \text{Eq. (4.11)} \\
&= \alpha \left[\mathrm{bin}_{n-1,\beta}\left(\frac{n-i}{2}\right) - \left(\frac{\alpha}{\beta}\right)^{i} \mathrm{bin}_{n-1,\beta}\left(\frac{n+i}{2}\right) \right] \\
&= \alpha \left[\binom{n-1}{\frac{n-i}{2}} \beta^{\frac{k-i}{2}} \alpha^{\frac{k+i}{2}-1} - \left(\frac{\alpha}{\beta}\right)^{i} \binom{n-1}{\frac{k+i}{2}} \beta^{\frac{k+i}{2}} \alpha^{\frac{k-i}{2}-1} \right] \\
&= \binom{n-1}{\frac{n-i}{2}} \beta^{\frac{n-i}{2}} \alpha^{\frac{n+i}{2}} - \binom{n-1}{\frac{n-i}{2}-1} \beta^{\frac{n-i}{2}} \alpha^{\frac{n+i}{2}} \\
&= \frac{n+i}{2n} \binom{n}{\frac{n-i}{2}} \beta^{\frac{n-i}{2}} \alpha^{\frac{n+i}{2}} - \frac{n-i}{2n} \binom{n}{\frac{n-i}{2}} \beta^{\frac{n-i}{2}} \alpha^{\frac{n+i}{2}} \\
&= \frac{i}{n} \binom{n}{\frac{n-i}{2}} \beta^{\frac{n-i}{2}} \alpha^{\frac{n+i}{2}}\,, \quad 0 < n < N\,.
\end{aligned} \tag{A.2}$$

Substituting this into Eq. (A.1) yields

$$\begin{aligned}
p_{i,0}^{(n)} &= \sum_{k=1}^{n} \frac{i}{k} \binom{k}{\frac{k-i}{2}} \beta^{\frac{k-i}{2}} \alpha^{\frac{k+i}{2}} \\
&= \sum_{j=\frac{1-i}{2}}^{\frac{n-i}{2}} \frac{i}{2j+i} \binom{2j+i}{j} \beta^{j} \alpha^{j+i}\,, \quad n < N\,,
\end{aligned} \tag{A.3}$$

So now

$$\begin{aligned}
\Pr(X_n = 0) &= \sum_{i=0}^{N} \Pr(X_0 = i)\, p_{i,0}^{(n)} \\
&= 2^{-N} \sum_{i=0}^{N} \binom{N}{i} \sum_{j=\frac{1-i}{2}}^{\frac{n-i}{2}} \frac{i}{2j+i} \binom{2j+i}{j} \beta^{j} \alpha^{j+i}\,, \quad n < N\,,
\end{aligned} \tag{A.4}$$

which is the expression that Schöning used.

A.2 Normal Moments

The kth moment (about the origin) of random variable X is defined by

$$\mu_k^{'} = \mathrm{E}[X^k] \,. \tag{A.5}$$

They are obtained from the moment generating function $m_X(s)$ of X by

$$\mu_k^{'} = \left[\frac{\mathrm{d}^k}{\mathrm{d}s^k} m_X(s) \right]_{s=0} \,. \tag{A.6}$$

The moment generating function of $\mathrm{Norm}(\mu, \sigma)$ distributed X reads [29, p. 422]

$$m_X(s) = \mathrm{e}^{\mu s + (\sigma s)^2} \,. \tag{A.7}$$

Calculating few of them give

$$\begin{aligned}
\mu_0^{'} &= 1 \,, \\
\mu_1^{'} &= \mu \,, \\
\mu_2^{'} &= \mu^2 + \sigma^2 \,, \\
\mu_3^{'} &= \mu^3 + 3\mu\sigma^2 \,, \\
\mu_4^{'} &= \mu^4 + 6\mu^2\sigma^2 + 3\sigma^2 \,, \\
\mu_5^{'} &= \mu^5 + 10\mu^3\sigma^2 + 15\mu\sigma^4 \,, \\
\mu_6^{'} &= \mu^6 + 15\mu^4\sigma^2 + 45\mu^2\sigma^4 + 15\sigma^6 \,, \\
\mu_7^{'} &= \mu^7 + 21\mu^5\sigma^2 + 105\mu^3\sigma^4 + 105\mu\sigma^6 \,.
\end{aligned}$$

Appendix B Matlab Code

Here the key results are presented as Matlab code. Many of the figures were generated utilizing these functions.

Function `pmat` returns the right-hand side of Eq. (3.9).

```
function P = pmat(N,k)

% One-step transition probability matrix

a = 1/k;
b = 1-a;

P = diag(ones(N,1),-1);
P = a*P + b*transpose(P);

P(1,1) = 1;
P(1,2) = 0;
P(end,end-1) = 1;
```

Function `initp` returns the initial distribution vector $\boldsymbol{\omega}$ as in Eq. (3.3a).

```
function p = initp(N)

% Initial probability distribution

p = binopdf(0:N,N,1/2);
```

Function `prob` calculates the $\rho(n)$ as in Eq. (3.13).

```
function r = prob(n,N,k)

% n-step absobtion probability

P = pmat(N,k);
w = initp(N);
p = w * P^round(n);
r = p(1);
```

Function `probY` calculates $\Pr(Y_n = 0)$ as in Eq. (4.14).

```
function y = probY(n,N,k)

% n-step absorption probability of Markov chain Y

a = 1/k;
b = 1-a;
i = 0:N;

v=binocdf((n-i)/2,n,b)+(a/b).^i.*(1-binocdf((n+i)/2,n,b));

% The following is needed to patch an error in 'binopdf';
% the possible NaN's are replaced by zeros.
v1=v1(1:1+min([n,N]));
v1=horzcat(v1,zeros(1,N-min([n,N])));

y = initp(N)*v1'; % Total probability
```

Function `optimum` finds the minimum $n/\rho(n)$ (Eq. (4.3)).

```
function [v,i] = optimum(N,k)

% Finds minimum of n/prob(n,N,k)

% interval to search within
b1 = round(N/(k-2));
b2 = 2*N;

len = b2-b1+1;
y = zeros(1,len);

P = pmat(N,k);
p = initp(N) * P^b1;

y(1) = b1/p(1);
for j = 2:len
        p = p*P;
        y(j) = (b1+j-1)/p(1);
end

[v,j] = min(y);

if j==1 | j==len
    v=-1;
    i=-1;
else
    i=b1+j-1;
end
```

Function `rootfunc` returns an expression corresponding to the right-hand side of Eq. (4.31).

```
function y = rootfunc(n,N,k,T)

% Function whose root represents approximate n_opt

a = 1/k;
b = 1-a;

% Moments
myy = 0:T;
syms x;
for i = 0:T
    m = exp(a*N*x + a*b*N/2*x^2);
    myy(i+1) = subs(diff(m,'x',i),x,0);
end

% Taylor coefficients
coef = zeros(size(myy));
coef(1) = 1/2;
for i = 0:floor((T-1)/2);
    coef(2*i+1+1) = 1/sqrt(2*pi) * (-1)^i/(factorial(i)*2^i*(2*i+1));
end

% The root-function
r = 0;
for i = 0:T
    h = 0;
    for j = 0:i
        h = h + mfun('binomial',i,j)*(-1)^j*(b-a)^(i-j)*myy(j+1)*x^(i/2-j-1);
    end
    r = r + coef(i+1)/(2*sqrt(b*a))^i*h;
end
r = diff(r,'x');
r = -sqrt(pi)*x^(T/2+2)*r;
r = simplify(r);
y = subs(r,x,n);
```

Function `optimum_app` returns an approximate n_{opt} by finding a root of the expression that `rootfunc` returns.

```
function i = optimum_app(N,k,T)

% approximate n_opt

i=fzero(@rootfunc,[N/(k-2),2*N/(k-2)],[],N,k,T);
```