Diplomarbeit

Evolutionäre Algorithmen als Adaptationsschemata

Dirk Sudholt

Diplomarbeit
am Fachbereich Informatik
der Universität Dortmund

16. November 2004

Gutachter:

Dr. Stefan Droste
Prof. Dr. Ingo Wegener

Inhaltsverzeichnis

Symbolverzeichnis

$A <_f B$	Alle Suchpunkte in der Menge A haben bezüglich f eine kleinere Fitness als alle Suchpunkte in der Menge B.
A	Das Ereignis, dass ein Schritt des (1+1) EA erfolgreich ist.
A^*	Das Ereignis, dass ein Schritt des (1+1)* EA erfolgreich ist.
$\deg(v)$	Der Grad eines Knotens v.
$\deg_x^+(v)$	Die Zahl der zum Knoten v inzidenten und in der Färbung x einfarbigen Kanten.
$\deg_x^-(v)$	Die Zahl der zum Knoten v inzidenten und in der Färbung x zweifarbigen Kanten.
e	Die Eulersche Zahl $e = 2{,}718\ldots$.
$f(P)$	Die Fitness der Population P mit Fitness Sharing.
$f(x)$	Die Fitness eines Suchpunkts x.
$f(x, E')$	Der Beitrag einer Kantenmenge E' zur Fitness von x.
$f(x, e)$	Der Beitrag einer Kante e zur Fitness von x.
$f(x, P)$	Die Fitness von x mit Fitness Sharing bezogen auf die Population P.
f_G	Kurze Schreibweise für Ising_G, die Ising-Funktion auf dem Graphen G.
$G = (V, E)$	Ein ungerichteter Graph mit Knotenmenge V und Kantenmenge E.
$H(x, y)$	Der Hammingabstand zwischen x und y: Die Zahl der Bitpositionen, in denen sich x und y voneinander unterscheiden.
Ising_G	Die Ising-Funktion auf dem Graphen G.
$\ell(x, x_i)$	Die lokale Fitnessveränderung, wenn im Suchpunkt x genau das Bit x_i flippt.
$\text{maj}(x, C)$	Die Mehrheit in der Clique C bezogen auf die Färbung x.
n	Die Dimension des Suchraums $\{0, 1\}^n$.

$\varphi(x)$	Die Potenzialfunktion φ, die Summe der Mehrheiten in der Färbung x.
p_m	Die Mutationswahrscheinlichkeit des (1+1) EA; die Wahrscheinlichkeit, mit der jedes Bit geflippt wird.
$\psi(x)$	Die minimale Tiefe eines Teilbaums unter allen verbessernden Teilbäumen in x.
R	Das Ereignis, dass ein Schritt des (1+1) EA relevant ist.
$Sh(x, P)$	Die Sharing-Funktion, ein Maß für die Ähnlichkeit von $x \in P$ zu allen Individuen der Population P.
$T(v)$	Ein Teilbaum mit Wurzel v.

Kapitel 1

Einleitung

Die vorliegende Diplomarbeit untersucht die Adaptationsfähigkeit einfacher evolutionärer Algorithmen anhand ausgewählter Fitnesslandschaften. Verschiedene Klassen interessanter Fitnesslandschaften werden mit Hilfe des so genannten Ising-Modells generiert, das sich gut zur Analyse der Adaptationsfähigkeit von Algorithmen eignet.

Ziel dieses ersten Kapitels ist es, den Leser in die in dieser Arbeit behandelten Themen, Begriffe und Methoden einzuführen. In Abschnitt 1.1 werden wir die Klasse der hier untersuchten Algorithmen kennen lernen, die Klasse der randomisierten Suchheuristiken, zu denen unter anderem evolutionäre Algorithmen zählen. Abschnitt 1.2 macht den Leser mit einigen ausgewählten Aspekten evolutionärer Algorithmen vertraut, deren Verständnis für die folgenden Themen hilfreich ist.

In Abschnitt 1.3 werden wir das Ising-Modell vorstellen und auf die bisherigen Erkenntnisse über verschiedene Instanzen dieses Modells eingehen. Abschnitt 1.4 begründet anschließend die Motivation zu dieser Arbeit und den in dieser Arbeit untersuchten Fragestellungen.

Schließlich werden wir in Abschnitt 1.5 einige gebräuchliche Analysemethoden für evolutionärer Algorithmen kennen lernen, die in den folgenden Kapiteln als bekannt vorausgesetzt werden.

1.1 Randomisierte Suchheuristiken

Randomisierte Suchheuristiken sind allgemeine Verfahren zur Lösung von Optimierungsproblemen. Gegeben ist eine Funktion $f\colon S \to Z$, die von einem *Suchraum* in einen (partiell) geordneten Raum Z von Zielwerten abbildet und gesucht ist ein *Individuum* oder *Suchpunkt* $x \in S$, der den Funktionswert von f maximiert bzw. minimiert. Einen solchen Suchpunkt bezeichnen wir als *globales Optimum*; die Funktion f, die optimiert werden soll, bezeichnen wir als *Fitnessfunktion*.

Ein häufig verwendeter Suchraum ist $S = \{0,1\}^n$, bei dem jeder Suchpunkt $x = (x_1, \ldots, x_n)$ eine Folge von n Bits repräsentiert. Als Raum Z von Zielwerten wird

meist $\mathbb{R}$ gewählt; Funktionen $f\colon \{0,1\}^n \to \mathbb{R}$ werden *pseudoboolesche Funktionen* genannt. Zu minimierende Funktionen können leicht in zu maximierende Funktionen überführt werden, indem man anstelle der Funktion f die Funktion $f'(x) := (-1) \cdot f(x)$ betrachtet. Wir gehen daher im Folgenden davon aus, dass wir stets maximieren.

Eine allgemeine randomisierte Suchheuristik wählt einen Startpunkt $x^0 \in S$ und bestimmt dessen Fitness durch Auswertung von $f(x^0)$. Abhängig von den bisher ausgewerteten Suchpunkten und ihren Fitnesswerten wird dann ein neuer Suchpunkt ausgewählt. Die Wahl von x^t im t-ten Schritt hängt also nur ab von

$$(x^0, f(x^0)), \ldots, (x^{t-1}, f(x^{t-1})).$$

Die Wahl eines neuen Suchpunkts ist meist effizient möglich, während die Auswertung der Fitness bei praktischen Problemen u. U. lange dauern kann. Daher liegt das Augenmerk nicht auf der Zahl der Operationen, die ein Rechner insgesamt ausführt, sondern nur auf der Zahl der benötigten Fitnessauswertungen. Die Zahl der Fitnessauswertungen, die eine randomisierte Suchheuristik ausführt, bevor zum ersten Mal ein global optimaler Suchpunkt ausgewertet wird, bezeichnen wir als *Optimierungszeit*.

In praktischen Anwendungen wird üblicherweise ein *Stoppkriterium* angewendet, das den Algorithmus stoppt, sobald eine zufrieden stellende Lösung erreicht wurde oder sobald die zur Verfügung stehenden Ressourcen (Zeit, Geduld, Geld, ...) zur Neige gehen. Wir nehmen eine andere Sichtweise ein und betrachten randomisierte Suchheuristiken als unendliche Prozesse. Mit diesem Hintergrund analysieren wir die erwartete Zeit, bis zum ersten Mal ein besonderes Ereignis eintritt, z. B. die erste Auswertung eines globalen Optimums.

Randomisierte lokale Suche

Eine Charakteristik, die vielen randomisierten Suchheuristiken gemeinsam ist, ist die Annahme, dass in der Nähe guter Suchpunkte weitere gute Suchpunkte liegen. Daher arbeiten viele solcher Suchheuristiken mit einer *Nachbarschaft* $N(x)$ um einen Suchpunkt x und wählen den neuen Suchpunkt y aus dieser Nachbarschaft. Als Metrik wird im Suchraum $\{0,1\}^n$ üblicherweise der *Hammingabstand* $H(x,y)$ verwendet, der die Zahl der Bitpositionen angibt, in denen sich x und y voneinander unterscheiden:

$$H(x,y) := \sum_{i=1}^{n} x_i \oplus y_i.$$

Die *randomisierte lokale Suche* auf dem Suchraum $S = \{0,1\}^n$ hat eine sehr kleine Nachbarschaft von allen Hammingnachbarn, das sind alle Suchpunkte mit Hammingabstand 1:

$$N(x) := \{y \in S \mid H(x,y) = 1\}.$$

Die randomisierte lokale Suche speichert stets das bisher beste gesehene Individuum x. In einem Schritt wird uniform zufällig ein Individuum $x' \in N(x)$ gewählt und

ausgewertet; x' bezeichnen wir als *Nachkommen* von x und x als *Elter* von x'. Der Nachkomme x' ersetzt den Elter x genau dann, wenn die Fitness des Nachkommens nicht schlechter ist als die des Elters.

Algorithmus 1 (Randomisierte lokale Suche).

1. *Initialisierung: Wähle $x \in \{0,1\}^n$ uniform zufällig.*
2. *Wähle $x' \in N(x)$ uniform zufällig.*
3. *Falls $f(x') \geq f(x)$, ersetze x durch x'.*
4. *Weiter bei 2.*

Damit gehört die randomisierte lokale Suche zu den *„hill climber"* Algorithmen, die nur Verbesserungen akzeptieren. Dadurch besteht die Gefahr, dass die randomisierte lokale Suche in *lokalen Optima* stecken bleibt, das sind Suchpunkte, bei denen alle Hammingnachbarn eine kleinere Fitness haben.

In diesem Fall kann der Einsatz von *Multistarts* helfen, d. h. der Algorithmus wird mehrmals parallel gestartet und der beste Suchpunkt aller Läufe ausgegeben. So erhöht sich die Wahrscheinlichkeit, dass zumindest einer dieser Läufe ein gutes Ergebnis liefert. Alternativ zur parallelen Ausführung können Läufe auch sequentiell ausgeführt werden: Ein Lauf wird gestoppt und neu gestartet, wenn eine Stagnation festgestellt oder eine vorgegebene Zeitschranke überschritten wird.

Metropolis Algorithmus

Eine andere Möglichkeit, eine Stagnation in schlechten lokalen Optima zu vermeiden, ist, Suchpunkte mit schlechterer Fitness zu akzeptieren, so dass lokale Optima wieder verlassen werden können. Im Gegensatz zur rein zufälligen Suche sollen allerdings weiterhin Suchpunkte mit höherer Fitness bevorzugt werden. Ein Kompromiss zwischen diesen beiden Zielen stellt der *Metropolis-Algorithmus* dar.

Dieser akzeptiert einen neuen Suchpunkt x' als Nachkomme von x auf jeden Fall, wenn der neue Suchpunkt nicht schlechter ist als sein Elter. Anderenfalls, wenn also x' eine schlechtere Fitness als x hat, wird x' nur mit einer gewissen Wahrscheinlichkeit akzeptiert. Diese Wahrscheinlichkeit hängt unter anderem von der Fitnessdifferenz $f(x') - f(x)$ ab. Üblicherweise wählt man hier die so genannte *Boltzmann-Funktion*

$$\text{Prob}(x' \text{ wird akzeptiert}) = e^{\frac{f(x')-f(x)}{T}}$$

Der Parameter T wird als *Temperatur* bezeichnet. Bei einer hohen Temperatur ist die Wahrscheinlichkeit groß, dass schlechtere Suchpunkte akzeptiert werden. Senkt man die Temperatur, wird diese Wahrscheinlichkeit immer kleiner. Im Grenzübergang $T \to 0$ erhalten wir die randomisierte lokale Suche, da keine Verschlechterungen mehr akzeptiert werden.

Das so genannte *Simulated Annealing* (simuliertes Abkühlen) ist eine Variante des Metropolis-Algorithmus, die an Verfahren zur Erzeugung besonders reiner Kristalle angelehnt ist. Man startet mit einer hohen Temperatur und senkt diese dann langsam ab. Dadurch hofft man, zu Anfang den Suchraum großräumig zu explorieren und beim langsamen Abkühlen in Regionen mit hoher Fitness zu verbleiben.

Evolutionäre Algorithmen

Evolutionäre Algorithmen verfolgen zwei andere Strategien, um Probleme mit lokalen Optima zu vermeiden. Zum einen arbeiten die verwendeten Operatoren meist auf einer wesentlich größeren Nachbarschaft als die randomisierte lokale Suche. Zum anderen besteht die Möglichkeit, *Populationen* zu verwenden, die mehrere Suchpunkte enthalten können. Im Gegensatz zu Multistarts mit unabhängigen Läufen können sich die Suchpunkte in der Population gegenseitig beeinflussen und so den Optimierungsprozess gemeinsam lenken.

Allgemein erzeugen evolutionäre Algorithmen neue Suchpunkte durch Anwendung von *Mutations-* und/oder *Rekombinationsoperatoren*. Eine Mutation erzeugt aus einem Suchpunkt x einen neuen Suchpunkt x'. Eine Rekombination erzeugt aus mehreren Suchpunkten der Population einen oder mehrere Suchpunkte, die allen beteiligten Eltern ähnlich sind.

Der Einsatz von Populationen hat gegenüber der allgemeinen Form randomisierter Suchheuristiken den Vorteil, dass nicht alle bisher ausgewerteten Suchpunkte, sondern nur eine Teilmenge davon gespeichert werden muss. Die Wahl eines neuen Suchpunkts hängt dann nur von den Individuen der aktuellen Population ab. Dabei werden oft mehrere Individuen auf einen Schlag mit Hilfe der gleichen Population erzeugt; ein Großschritt, den wir als eine *Generation* bezeichnen.

Zusätzlich zu Mutations- und Rekombinationsoperatoren gibt es *Selektionsoperatoren*, die die Verwaltung der Population bestimmen. Zum einen wählen sie aus der aktuellen Population die Eltern aus, aus denen Nachkommen erzeugt werden sollen (Selektion zur Reproduktion). Zum anderen wird durch Selektion bestimmt, welche Suchpunkte in die Population der nächsten Generation übernommen werden sollen (Selektion zur Übernahme).

Generell gibt es hier zwei verschiedene Strategien der Populationsverwaltung. Zum einen gibt es evolutionäre Algorithmen mit so genannte *Plusstrategien*, die bei einer Populationsgröße von μ und λ Nachkommen auch als $(\mu+\lambda)$ EA bezeichnet werden. Es werden λ Nachkommen erzeugt und anschließend aus den μ Suchpunkten der aktuellen Generation und den λ Nachkommen μ Suchpunkte selektiert, die die neue Population bilden. Bei einer *Kommastrategie*, kurz (μ,λ) EA, werden $\lambda \geq \mu$ Nachkommen erzeugt und die Suchpunkte der neuen Population werden nur aus der Menge der Nachkommen gewählt. Bei einer Kommastrategie kann es vorkommen, dass die durchschnittliche Fitness der Population sinkt, während bei einer Plusstrategie und entsprechenden Selektionsoperatoren sicher gestellt werden kann, dass die durchschnittliche Fitness der Population über die Zeit monoton ist.

Zu den evolutionären Algorithmen zählen wir sowohl *Evolutionsstrategien* als auch *genetische Algorithmen*, die verschiedene historische Ursprünge haben. Evolutionsstrategien setzen den Schwerpunkt auf Mutationen und werden hauptsächlich zur Optimierung von Funktionen eingesetzt. Genetische Algorithmen arbeiten meist auf dem Suchraum $\{0,1\}^n$ und verwenden hauptsächlich Rekombinationen; Mutationen dienen oft nur dazu, die Diversität der Population zu erhöhen und Suchpunkte zu erzeugen, die mit Rekombinationen allein nicht erreichbar sind. Auch wenn beide Formen verschiedene Ursprünge haben, fasst man sie üblicherweise unter dem Oberbegriff „Evolutionäre Algorithmen“ zusammen.

In dieser Arbeit betrachten wir hauptsächlich evolutionäre Algorithmen mit Plusstrategien auf dem Suchraum $\{0,1\}^n$. Die Nachbarschaft eines Suchpunkts umfasst hier den gesamten Suchraum, $N(x) := \{0,1\}^n$, allerdings werden Suchpunkte in der Nähe des aktuellen Suchpunkts bevorzugt. Der Mutationsoperator erzeugt einen neuen Suchpunkt x' aus x dadurch, dass jedes Bit unabhängig von den anderen Bits mit einer bestimmten *Mutationswahrscheinlichkeit* p_m geflippt (invertiert) wird.

Der wohl einfachste evolutionäre Algorithmus dieser Art ist der (1+1) EA mit Populationsgröße 1, der in jedem Schritt einen Nachkommen erzeugt.

Algorithmus 2 ((1+1) EA mit Mutationswahrscheinlichkeit p_m).

1. *Initialisierung: Wähle $x \in \{0,1\}^n$ uniform zufällig.*
2. *$x' := x$. Flippe jedes Bit in x' mit Wahrscheinlichkeit p_m unabhängig von den anderen Bits.*
3. *Falls $f(x') \geq f(x)$, ersetze x durch x'.*
4. *Weiter bei 2.*

Die Standardwahl für p_m ist $p_m := 1/n$, bei der im Erwartungswert genau ein Bit geflippt wird. Unter dem Begriff „(1+1) EA“ verstehen wir in Zukunft den (1+1) EA mit Mutationswahrscheinlichkeit $1/n$.

Bei einer Mutationswahrscheinlichkeit von $1/n$ ist die Wahrscheinlichkeit, dass genau ein Bit geflippt wird,

$$\left(1 - \frac{1}{n}\right)^{n-1} \geq e^{-1},$$

so dass der Operator dem der randomisierten lokalen Suche ähnelt. Andererseits sind mit diesem Mutationsoperator auch größere Änderungen möglich. Jeder andere Suchpunkt y hat eine positive Wahrscheinlichkeit von

$$\frac{1}{n^{H(x,y)}} \cdot \left(1 - \frac{1}{n}\right)^{n-H(x,y)} \geq n^{-n},$$

aus x durch Mutation erzeugt zu werden. Der (1+1) EA kann nicht in schlechten lokalen Optima hängen bleiben, da in jedem Schritt die Möglichkeit besteht, direkt zu einem globalen Optimum zu springen (auch wenn die Wahrscheinlichkeiten sehr klein sein können). Mit diesen Überlegungen kann man zudem zeigen, dass die erwartete Optimierungszeit für jede Fitnessfunktion durch n^n nach oben beschränkt ist.

1.2 Aspekte evolutionärer Algorithmen

An dieser Stelle sollen einige ausgewählte Aspekte evolutionärer Algorithmen erläutert werden, auf die wir später zurück kommen werden.

1.2.1 Diversitätserhaltung

Ein Problem bei evolutionären Algorithmen mit Populationen ist der so genannte *Hitchhiking-Effekt*: Mit der Zeit übernimmt ein gutes Individuum die gesamte Population, so dass am Ende die Population nur aus Kopien eines einzigen Suchpunkts besteht. Um dieses Problem zu vermeiden, gibt es verschiedene Methoden der Diversitätserhaltung.

Zu diesen Methoden gehören so genannte *Niching-Techniken*, die die Nischenbildung von Suchpunkten belohnen: Suchpunkte in „Nischen“ des Suchraums sollen bevorzugt behandelt werden, so dass die Nische exploriert werden kann, ohne dass der Suchpunkt durch andere Suchpunkte aus der Population verdrängt wird.

Die vielleicht bekannteste Niching-Technik ist das *Fitness Sharing*. Der Name rührt daher, dass mehrere Kopien eines Suchpunkts die Fitness des Suchpunkts unter sich aufteilen. Wenn es k Instanzen eines Suchpunkts x gibt, erhält jede Instanz nur eine effektive Fitness von $f(x)/k$, so dass die Bildung von „Klumpen“ bestraft wird und Kopien aus der Population verdrängt werden.

Allgemein wird die Fitness eines Suchpunkts auf die Population P bezogen. Dieser neue Fitnesswert $f(x, P)$ errechnet sich aus der „echten“ Fitness, indem man den Fitnesswert $f(x)$ durch die *Sharing-Funktion* $Sh(x, P)$ teilt.

$$f(x, P) = \frac{f(x)}{Sh(x, P)}$$

Die Sharing-Funktion ist dabei ein Maß für die Nähe zwischen x und den anderen Suchpunkten aus der Population P.

Eine andere, natürliche Art der Diversitätserhaltung bieten die bereits erwähnten Multistarts, da die Läufe unabhängig voneinander sind. Wenn beispielsweise die Wahrscheinlichkeit für einen (wie auch immer definierten) Erfolg durch einen polynomiell kleinen Wert $1/p(n)$ nach unten beschränkt ist, können wir $k \cdot p(n)$ unabhängige Läufe ausführen. Damit ist die Wahrscheinlichkeit, in allen Läufen Misserfolg zu haben, exponentiell klein in k:

$$\text{Prob(Misserfolg in allen } k \cdot p(n) \text{ Läufen)} \leq \left(1 - \frac{1}{p(n)}\right)^{k \cdot p(n)} \leq e^{-k}.$$

1.2.2 Schema-Theorie und die Building-Block-Hypothese

Die *Schema-Theorie* von Holland [11] ist in der Forschung genetischer Algorithmen weit verbreitet und erlaubt lokale Aussagen über die Verbreitung bestimmter Individuen in der Population.

Man teilt den Suchraum $\{0,1\}^n$ in Teilmengen ein, so genannte *Schemata*, indem man bestimmte Bitpositionen festlegt und die übrigen Positionen variabel lässt. So ist beispielsweise $10*1*$ ein Schema, das die Suchpunkte

$$10010, 10011, 10110, 10111$$

enthält. Anschließend betrachtet man den Anteil an Individuen in der Population, die zu einem bestimmten Schema ξ gehören. Abhängig von der durchschnittlichen Fitness aller Individuen in ξ kann man dann mit Erwartungswerten vorhersagen, wie sich der Anteil der Individuen in der Population, die zu ξ gehören, beim Übergang zur nächsten Generation verändern wird.

Die Hoffnung bei der Anwendung der Schema-Theorie ist, dass man „gute“ Schemata mit hoher durchschnittlicher Fitness finden kann und diese sich schnell in der Population ausbreiten. Aus dieser Hoffnung ist die so genannte *Building-Block-Hypothese* hervorgegangen: Die festgelegten Positionen guter Schemata bezeichnet man als „Bausteine“ oder „building blocks“ guter Lösungen. Wenn eine Funktion mehrere gute Schemata besitzt, die in der Population vertreten sind, können die Bausteine durch Rekombination der entsprechenden Individuen zusammengesetzt werden, um eine neue, bessere Lösung zu bilden. Genetische Algorithmen arbeiten demnach implizit wie Algorithmen der dynamischen Programmierung, indem sie gute Lösungen für Teilprobleme evolvieren und diese dann zu einer guten Gesamtlösung zusammensetzen.

Spätestens an dieser Stelle sind einige kritische Worte angebracht. Die Building-Block-Hypothese steht und fällt mit der Annahme, dass die Fitnessfunktion (zumindest annähernd) *separabel* ist. Bei einer separablen Fitnessfunktion lässt sich die Variablenmenge X eines Suchpunkts x in $k \geq 2$ nicht leere Teilmengen $X_1, \ldots, X_k$ und die Funktion f sich entsprechend in Teilfunktionen $f_1, \ldots, f_k$ partitionieren, so dass die Fitness die Summe dieser Teilfunktionen ist:

$$f(x) = \sum_{i=1}^{k} f_i(X_i).$$

Eine solche Eigenschaft ist jedoch nur für sehr wenige Funktionen erfüllt, die zudem meist leicht zu optimieren sind, da man die Teilfunktionen f_i separat optimieren kann.

Auf vielen Funktionenklassen scheint die Building-Block-Hypothese also wenig Sinn zu machen. Hinzu kommt, dass die Building-Block-Hypothese sehr schwammig formuliert ist und durch ihre Vagheit prinzipiell nicht widerlegt werden kann. Daher ist die Building-Block-Hypothese mit einiger Vorsicht und Skepsis zu behandeln.

Dennoch erwähnen wir die Building-Block-Hypothese aus zweierlei Gründen: Zum einen, da sie in der Forschung trotz aller Kritik präsent ist. Zum anderen, da wir in dieser Arbeit Funktionen begegnen werden, bei denen Teile von Suchpunkten besagten „Bausteinen“ ähneln, so dass der Begriff der Bausteine für eine oberflächliche Diskussion angebracht erscheint.

1.2.3 Mutation versus Rekombination

Ein anderes Thema, das in der Forschung kontrovers diskutiert wird, ist die Frage, welche Operatoren den größeren Nutzen bringen: Mutationen oder Rekombinationen?

Begründet ist diese Frage vor allem in den verschiedenen historischen Wurzeln von Evolutionsstrategien auf der einen Seite, die ausschließlich mit Mutationsoperatoren arbeiten und genetischen Algorithmen auf der anderen Seite, deren Hauptaugenmerk auf Rekombination liegt. Daher verwundert es nicht, dass verschiedene Forscher aus dem Bereich evolutionärer Algorithmen oft ganz unterschiedliche Auffassungen über den Nutzen von Mutation und Rekombination haben.

In theoretischen Analysen ist der Effekt von Rekombination oft nur schwer greifbar. Zum einen führt Rekombination im Allgemeinen zu einer Streuung der erzeugten Nachkommen über einen großen Bereich des Suchraums. Mutationen hingegen arbeiten eher lokal begrenzt, so dass sich die Auswirkungen eines Operators bei Mutationen sehr viel besser analysieren lassen als bei Rekombinationen.

Zum anderen setzen Rekombinationen eine entsprechende Population von Suchpunkten voraus, während bei Mutation eine Populationsgröße von 1 ausreicht. Die Selektion zur Reproduktion und die Diversität der Population spielen eine wichtige Rolle bei der Erzeugung von Nachkommen und beim Nutzen von Rekombination, daher müssen diese Effekte in theoretischen Analysen zusätzlich berücksichtigt werden.

Insgesamt führt der Einsatz von Rekombination aufgrund der nötigen Populationsstrukturen und der Interaktion zwischen mehreren Suchpunkten zu einem wesentlich komplexeren System verglichen mit einfachen Algorithmen, die nur mit Mutation arbeiten.

Es war daher lange Zeit eine offene Frage, eine Funktion zu finden, für die man beweisen kann, dass der Einsatz von Rekombination essenziell ist. Mitchell, Forrest und Holland [15] brachten eine Klasse von Funktionen in die Diskussion ein, die sie *Royal Road Funktionen* nannten. Dabei wird der Bitstring in gleich große disjunkte Blöcke zerlegt, die als „Building Blocks“ einer optimalen Lösung fungieren sollen. Ein solcher Block trägt genau dann den Wert 1 zur Fitness bei, wenn er komplett mit Einsen belegt ist, ansonsten trägt er den Wert 0 zur Fitness bei.

Royal Road Funktionen sollten den Nutzen von Rekombination nachweisen und einen „Königsweg“ für genetische Algorithmen bereiten, da nach der Schema Theorie und der Building Block Hypothese sehr leicht verschiedene 1-belegte Blöcke als Bausteine guter Lösungen zusammengesetzt werden können. Allerdings gingen diese Hoffnungen nicht in Erfüllung, da sich in weiteren Experimenten zeigte [16], dass ein mutationsbasierter Algorithmus die Royal Road Funktionen schneller optimiert als ein einfacher genetischer Algorithmus.

Daraufhin entwickelten Jansen und Wegener [13] eine weitere Klasse von Funktionen, die sie *Real Royal Road Funktionen* nannten. Für Real Royal Road Funktionen auf n Bits konnten sie beweisen, dass ein genetischer Algorithmus mit Populationsgröße n eine polynomielle erwartete Optimierungszeit hat, während der (1+1) EA

bei jeder Mutationswahrscheinlichkeit $0 \leq p_m \leq 1/2$ eine exponentielle erwartete Optimierungszeit hat. Auch andere mutationsbasierte Algorithmen tun sich schwer, da die Funktion in einem großen Bereich um das einzige globale Optimum keinerlei Hinweise liefert, wo das Optimum zu finden ist.

Als Kritik an Jansen und Wegener [13] kann man einwenden, dass der genetische Algorithmus mit einer Populationsgröße von n einen unfairen Vorteil gegenüber dem (1+1) EA hat, der nur mit einem Individuum arbeitet. Es blieb die Frage offen, ob es auch Funktionen gibt, bei denen Rekombination selbst dann essenziell ist, wenn wir nur konstante Populationsgrößen erlauben.

Diese Frage beantworten Storch und Wegener [19], indem sie alternative Real Royal Road Funktionen definieren, die für mutationsbasierte Algorithmen schwierig sind, aber von einem genetischen Algorithmus mit Rekombination und der kleinsten möglichen Populationsgröße effizient optimiert werden können, nämlich einer Populationsgröße von 2.

1.3 Das Ising-Modell

Geschichte und Anwendungen des Ising-Modells

Das *Ising-Modell* ist ein Modell aus der Statistischen Physik zur Beschreibung von Systemen mit Wechselwirkung. Es wurde von Ernst Ising [12] entwickelt und beschreibt das Verhalten von Elementen, die die Tendenz haben, sich benachbarten Elementen anzugleichen.

In der traditionellen Formulierung beschreibt das System die magnetischen Beziehungen zwischen Atomen, die zwei verschiedene *Spins* $s_i \in \{-1, +1\}$ annehmen können. Ein System aus mehreren Atomen nimmt einen Zustand kollektiver Ordnung an, wenn alle benachbarten Atome die gleichen Spins aufweisen. Wenn keine Energie von außen zugeführt wird, wenn also die Temperatur $T = 0$ Kelvin beträgt, wird dieser Ordnungszustand mathematisch formuliert durch die Funktion

$$H = -\sum_{i,j} J_{i,j} \cdot s_i \cdot s_j$$

wobei $J_{i,j}$ die Stärke der paarweisen Wechselwirkung zwischen den Atomen i und j angibt.

Auch in vielen anderen Bereichen ist das Ising-Modell anwendbar. Wenn die Wechselwirkungen $J_{i,j}$ nichtnegativ sind, lassen sich mit dem Ising-Modell Imitations- und Kooperationsprozesse beschreiben. So wurde das Ising-Modell beispielsweise verwendet, um das Verhalten von Fischen in Fischschwärmen zu modellieren oder die Kohärenz auf Kapitalmärkten zu beschreiben. Entsprechende Quellen finden sich in [14].

Weiterhin lassen sich auch Meinungsbildungsprozesse modellieren: Die Wechselwirkung $J_{i,j}$ beschreibt dann, welchen Einfluss Person i auf Person j ausübt, um sie

von ihrer Meinung zu überzeugen. Dabei können Wechselwirkungen auch negativ ausfallen, wenn Personen die Tendenz haben, sich in ihrer Meinung voneinander zu distanzieren.

Nicht zuletzt hatten Erkenntnisse über das Ising-Modell auch Auswirkungen auf die Neuroinformatik und die Modellierung neuronaler Netze. Das Hopfield-Modell beispielsweise wurde auf Basis des Ising-Modells formuliert. Es sieht für Neuronen die Zustände „ruhend“ und „feuernd“ vor. Feuernde Neuronen regen benachbarte Neuronen dazu an, ebenfalls zu feuern, so dass auch hier die Tendenz zur gleichen Ausrichtung benachbarter Elemente erkennbar ist. Durch verschieden stark ausgeprägte Wechselwirkungen zwischen den Neuronen kann ein solches neuronales Netz Informationen speichern und wiedergeben.

Das Ising-Modell im Kontext evolutionärer Algorithmen

Aus Sicht der Informatik entspricht das Ising-Modell einem Graphfärbungsproblem: Gegeben ist ein ungerichteter Graph $G = (V, E)$ mit Knoten $V = \{1, \ldots, n\}$ und eine Kantenmenge E. Ein Suchpunkt $x = (x_1, \ldots, x_n)$ entspricht einer *Färbung* dieser Knoten. Die Begriffe „Suchpunkt“ und „Färbung“ werden wir in dieser Arbeit synonym verwenden.

Anstelle der Werte -1 und $+1$ wendet man zur leichteren Handhabung eine affine Transformation an, so dass man Werte $x_i \in \{0, 1\}$ erhält. Ein Knoten kann daher mit den Farben 0 oder 1 gefärbt werden. Eine Färbung codieren wir durch einen Bitstring $x \in \{0, 1\}^n$, wobei jedes Bit durch eine bijektive Abbildung genau einem Knoten zugeordnet ist.

Anstelle von Wechselwirkungen $J_{i,j}$ sprechen wir von Gewichten $w_{ij} \in \mathbb{R}$. Die *Ising-Funktion* auf dem Graphen G ist dann wie folgt definiert:

$$\text{Ising}_G(x) := \sum_{\{i,j\} \in E} w_{ij} \cdot x_i \cdot x_j$$

Die Ising-Funktion wird in der Theorie evolutionärer Algorithmen gern als zu maximierende Fitnessfunktion verwendet, da sie einige interessante Eigenschaften besitzt.

Zum einen ist die Ising-Funktion *bit-flip symmetrisch*, d. h. wenn $\overline{x}$ das bitweise Komplement von x ist, gilt

$$\text{Ising}_G(x) = \text{Ising}_G(\overline{x}).$$

Dies gilt auch für globale Optima: zu jedem globalen Optimum gibt es ein weiteres, komplementäres globales Optimum. Das führt dazu, dass sich auf vielen Graphen verschiedene Teilgraphen in verschiedene Richtungen entwickeln können. In der Sprache genetischer Algorithmen liegen hier also verschiedene gute Bausteine optimaler Lösungen vor.

Ein Problem liegt dann darin, dass durch verschiedenartig gefärbte Bausteine lokale Optima entstehen können, die für viele evolutionäre Algorithmen schwierig sind. Mutationsbasierte evolutionäre Algorithmen müssen typischerweise viele Bits in einem

Schritt flippen, um ein solches lokales Optimum zu verlassen. Evolutionäre Algorithmen mit Rekombination können gute Bausteine zusammen führen und eine gute Gesamtlösung erzeugen, allerdings ist auf allgemeinen Graphen G nicht klar, wie ein vernünftiger allgemeiner Rekombinationsoperator aussehen kann. Ohne den Einsatz von Problemwissen und ohne Einschränkung der betrachteten Graphen scheint die Wahrscheinlichkeit, dass eine Rekombination genau die passenden Knoten eines Bausteins zur Rekombination auswählt, verschwindend gering.

In einer solchen Situation spricht man davon, dass ein Suchpunkt nicht synchronisiert ist; das Problem wird als *Synchronisationsproblem* bezeichnet [10].

Die Fitnesslandschaft der Ising-Funktion ist also multimodal; andererseits enthält die Fitnesslandschaft auf vielen Graphen *Plateaus*, das sind Mengen zusammenhängender Suchpunkte mit gleicher Fitness. Evolutionäre Algorithmen stehen also nicht nur vor der Herausforderung, lokale Optima zu vermeiden bzw. zu verlassen, sondern auch, Plateaus zu durchqueren und einen besseren Suchpunkt zu finden.

Die Komplexität der Ising-Funktion

Die Maximierung der Ising-Funktion mit positiven Gewichten w_{ij} ist trivial, da die einheitlichen Färbungen 0^n und 1^n optimal sind. Falls alle Gewichte den Wert $w_{ij} = -1$ haben, ist das Problem jedoch NP-hart. Die Maximierung der Ising-Funktion entspricht dann dem als NP-hart bekannten Problem MAX-CUT: Gegeben ist ein Graph $G = (V, E)$ und gesucht ist eine Partition der Knotenmenge in zwei Teilmengen $V = V_0 \dot\cup V_1$, die die Zahl der Kanten maximiert, die zwischen V_0 und V_1 verlaufen. Die Menge dieser Kanten bezeichnet man auch als *Schnitt*.

Die Ideen einer entsprechenden Turing-Reduktion sind folgende. Eine Färbung der Knoten des Graphen entspricht einer Partition von V in die Menge V_0 der 0-gefärbten Knoten und die Menge V_1 der 1-gefärbten Knoten und umgekehrt. Eine einheitlich gefärbte Kante trägt zur Ising-Funktion den Wert -1 bei und zur Größe des Schnitts den Wert 0. Eine Kante mit verschieden gefärbten Endpunkten trägt zur Ising-Funktion den Wert 0 bei und zur Größe des Schnitts den Wert 1. Daher unterscheiden sich die Werte möglicher Lösungen beider Probleme nur um den additiven Term $|E|$; eine optimale Lösung für die Ising-Funktion ist auch eine optimale Lösung für MAX-CUT.

Die Maximierung der Ising-Funktion ist daher mit beliebigen Gewichten NP-hart und mit nichtnegativen Gewichten trivial. Die Minimierung der Ising-Funktion entspricht der Maximierung von

$$(-1) \cdot \sum_{\{i,j\} \in E} w_{ij} \cdot x_i \cdot x_j = \sum_{\{i,j\} \in E} (-w_{ij}) \cdot x_i \cdot x_j.$$

Daher gilt analog: Die Minimierung der Ising-Funktion ist mit nichtpositiven Gewichten trivial und mit positiven und beliebigen Gewichten NP-hart.

Bisherige Erkenntnisse über das Ising-Modell

Das eindimensionale Ising-Modell wurde erstmals von Naudts und Naudts [17] in die Diskussion genetischer Algorithmen eingebracht als Beispiel für ein multimodales Optimierungsproblem, bei dem die Bit-Flip-Symmetrie entscheidenden Einfluss auf die Performanz einfacher genetischer Algorithmen hat.

Darauf aufbauend analysieren Goldberg, Van Hoyweghen und Naudts [8, 9] den Nutzen von Niching-Techniken, speziell Fitness-Sharing, zur Optimierung der Ising-Funktion. Ihrer Auffassung nach ist der Einsatz von Niching-Techniken für genetische Algorithmen auf dem eindimensionalen Ising-Modell zwingend notwendig. Allerdings beschränken sie sich in ihren Betrachtungen auf so genannte *selektorekombinative* genetische Algorithmen, die nur mit Rekombination und Selektion arbeiten.

Fischer [5] und Fischer und Wegener [7] analysieren daraufhin den Effekt von Mutationen auf die Optimierung des eindimensionalen Ising-Modells und vergleichen Mutation mit Rekombination. Für einen einfachen genetischen Algorithmus mit Populationsgröße 2, Rekombination und Fitness Sharing zeigen sie eine erwartete Laufzeit $O(n^2)$ und für den (1+1) EA, der ausschließlich mit Mutation arbeitet, zeigen sie eine Schranke für die erwartete Laufzeit von $O(n^3)$, die unter einer begründeten Vermutung scharf ist. Dies widerlegt anfänglich geäußerte Vermutungen innerhalb der Forschung evolutionärer Algorithmen, dass mutationsbasierte evolutionäre Algorithmen auf dem eindimensionalen Ising-Modell exponentielle Rechenzeit benötigen.

Fischer [5, 6] zeigt zudem für das zweidimensionale Ising-Modell, einen quadratischen Torus mit zusätzlichen diagonalen Kanten, dass der (1+1) EA in erwarteter Zeit $O(n^3)$ ein stabiles lokales oder globales Optimum erreicht. Ein einfacher Metropolis-Algorithmus mit niedriger Temperatur findet ein globales Optimum in erwarteter Zeit $O(n^{4,5})$. Dies ist ein überraschendes Resultat, da das zweidimensionale Ising-Modell von den meisten Forschern für schwierig gehalten wurde.

Schließlich wurden von den Teilnehmern der Projektgruppe 427 [1] experimentelle Untersuchungen zum Ising-Modell durchgeführt. Sie untersuchen die randomisierte lokale Suche und den (1+1) EA auf der Ising-Funktion auf verschiedenen Graphen: dem zweidimensionalen Torus, zwei miteinander verbundenen Cliquen und dem booleschen Hypercube.

1.4 Motivation

Das Ising-Modell bietet durch die unterschiedlichen Klassen von Graph-Instanzen eine große Bandbreite an interessanten Fitnesslandschaften. Zwei davon werden wir in dieser Arbeit genauer untersuchen: Cliquengraphen und vollständige binäre Bäume.

Das Ising-Modell auf Cliquengraphen

Eine sehr interessante Klasse von Fitnesslandschaften wird durch so genannte *Cliquengraphen* vorgegeben; das sind Graphen $G = (V, E)$, die sich aus disjunkten

Cliquen zusammensetzen. Als *Clique* bezeichnen wir eine Knotenmenge $C \subseteq V$, innerhalb der jeder Knoten mit jedem anderen verbunden ist:

$$\forall i, j \in C, i \neq j \colon \{i, j\} \in E.$$

Die Cliquen können unabhängige Zusammenhangskomponenten bilden oder durch so genannte *Brückenkanten* miteinander verbunden sein. Das Ising-Modell gibt im Allgemeinen aufgrund der hohen Dichte an Kanten sehr gute Hinweise dahin gehend, die Cliquen jeweils einheitlich zu färben. Falls alle Cliquen unabhängige Zusammenhangskomponenten bilden, wird ein globales Optimum erreicht, wenn alle Cliquen jeweils einheitlich gefärbt sind. Die erwartete Optimierungszeit sollte daher klein sein.

Falls die Cliquen durch wenige Brückenkanten miteinander verbunden sind, gibt es lokale Optima, bei denen verbundene Cliquen jeweils einheitlich, aber mit verschiedenen Farben gefärbt sind. Hier tun sich mutationsbasierte evolutionäre Algorithmen sehr schwer, diese lokalen Optima zu verlassen, da unter Umständen alle Knoten einer Clique in einem Schritt flippen müssen.

Im Fall verbundener Cliquen steht daher die Frage nach der Erfolgswahrscheinlichkeit im Vordergrund: Mit welcher Wahrscheinlichkeit erreichen einfache evolutionäre Algorithmen wie der (1+1) EA ein globales Optimum, ohne in einem schlechten lokalen Optimum hängen zu bleiben?

Ein Paradebeispiel für solche als Synchronisationsprobleme bezeichneten Probleme ist ein Graph, der aus zwei gleich großen disjunkten Cliquen besteht, die durch eine einzige Brückenkante miteinander verbunden sind. Eine Färbung, bei der beide Cliquen einheitlich, aber mit verschiedenen Farben gefärbt sind, entspricht einem lokalen Optimum. Dieses kann nur verlassen werden, wenn mindestens eine Clique in einem Schritt komplett flippt. Wir werden in dieser Arbeit nachweisen, dass der (1+1) EA auf diesem Graphen asymptotisch gesehen eine Worst-Case-Laufzeit hat. Dazu zeigen wir obere und untere Schranken für die erwartete Laufzeit des (1+1) EA, die sich nur um einen konstanten Faktor voneinander unterscheiden.

Die Frage nach der erwarteten Laufzeit des (1+1) EA auf unabhängigen Cliquen erscheint zunächst leicht: Die Fitness kann in suboptimalen Suchpunkten stets durch etliche 1-Bit-Mutationen verbessert werden. Allerdings kann es auch Mutationen mehrerer Bits geben, die den (1+1) EA vom nächsten globalen Optimum fort führen.

Es wird sich zeigen, dass einfache, verbreitete Methoden zur Analyse evolutionärer Algorithmen hier nicht zu einer scharfen oberen Schranke für die erwartete Optimierungszeit führen und dass sich die Aufgabe, eine scharfe Schranke zu beweisen, bei genauerer Betrachtung als unerwartet schwer erweist. Die Analyse des (1+1) EA gelingt schließlich nur, indem wir einen ähnlichen, leicht modifizierten Algorithmus analysieren und dann die dort erzielten Resultate auf den (1+1) EA übertragen. Dazu verwenden wir ausgefeilte Methoden zum Vergleich von Algorithmen, die zum ersten Mal von Fischer und Wegener [7] im Kontext evolutionärer Algorithmen eingesetzt wurden.

Das Ising-Modell auf Bäumen

Eine andere interessante Graphklasse stellen Bäume dar, wobei wir uns hauptsächlich auf die vielleicht typischsten Bäume beschränken, nämlich auf vollständige binäre Bäume. Bei der Optimierung des Ising-Modells auf Bäumen muss ein evolutionärer Algorithmus im Laufe der Zeit an jedem inneren Knoten v zufallsbasierte Entscheidungen treffen, wie der Teilbaum $T(v)$ zu färben ist.

Solche Entscheidungen werden auf verschiedenen Ebenen gefällt, wobei die Entscheidung auf einer höheren Ebene (wir stellen uns Bäume von oben nach unten gerichtet vor, mit der Wurzel ganz oben) von Entscheidungen auf unteren Ebenen beeinflusst wird. Daher können wir die Ising-Funktion auf Bäumen zu so genannten *hierarchischen Funktionen* zählen.

Die Ising-Funktion auf vollständigen binären Bäumen wurde bislang noch nicht untersucht; allerdings gibt es eine bekannte Funktion, die ähnliche Eigenschaften hat und bereits Gegenstand verschiedener Untersuchungen war: Die Funktion *Hierarchical-Iff* oder kurz *H-IFF*. Eine formale Definition findet sich in [2].

Die Funktion H-IFF hat nur indirekt mit Bäumen zu tun; man verwendet lediglich die Vorstellung eines vollständigen binären Baumes, um sich die Fitnessauswertung eines Suchpunkts zu veranschaulichen. Gegeben ist ein Suchpunkt $x = (x_1, \ldots, x_n)$, wobei $n = 2^k$ eine Zweierpotenz ist. Man stelle sich nun vor, dass man auf diesen n Bits einen vollständigen binären Baum konstruiert, so dass die n Bits die Blätter des Baumes bilden. Zusätzlich zu den n Blättern gibt es $n - 1$ gedachte innere Knoten.

Die Fitnessauswertung funktioniert dann wie folgt: Die Fitness ergibt sich als Summe von Beiträgen aller Knoten. Jeder Knoten v trägt genau dann einen von Null verschiedenen Wert zur Fitness bei, wenn alle Blätter im Teilbaum $T(v)$ einheitlich mit 0 oder einheitlich mit 1 belegt sind. In diesem Fall ist der Beitrag von v die Zahl der Blätter in v. Zur besseren Veranschaulichung kann man sich vorstellen, dass auch die inneren Knoten mit 0 oder 1 gefärbt werden können, wenn nämlich alle Blätter in ihrem Teilbaum den Wert 0 bzw. 1 haben (ansonsten bleibt der Knoten ungefärbt).

Der Funktionswert von H-IFF ist also hoch, wenn es große einheitlich gefärbte Teilbäume gibt. Das gleiche gilt für die Ising-Funktion auf vollständigen binären Bäumen; hier verhalten sich beide Fitnessfunktionen ähnlich. Allerdings gibt es auch Unterschiede:

1. Bei H-IFF codieren die Bits nur die Färbung der Blätter (die inneren Knoten sind nur Gedankenkonstrukte zur besseren Veranschaulichung der Fitnessauswertung), bei der Ising-Funktion werden alle Knoten durch Bits repräsentiert.

2. Wenn in einem einheitlich gefärbten Teilbaum ein Blatt die Farbe ändert, sinkt die Fitness bei H-IFF stark ab, da viele Teilbäume nicht mehr einheitlich gefärbt sind und somit nicht mehr zur Fitness beitragen. Die Ising-Funktion sinkt hingegen nur um 1.

3. Die Funktion H-IFF beinhaltet eine stärker ausgeprägte hierarchische Ordnung.

Die Funktion H-IFF gilt als sehr schwierig für einfache mutationsbasierte Algorithmen, da Mutationen i. A. nur lokal arbeiten. Bei hierarchischen Funktionen müssen jedoch oft große Bereiche eines Suchpunkts verändert werden, um eine Fitnessverbesserung zu erreichen. Bei einer geeigneten Codierung kann Rekombination helfen; für H-IFF wurde dies von Dietzfelbinger, Naudts, van Hoyweghen und Wegener [2] untersucht, wobei man einschränkend sagen muss, dass hier ein sehr spezieller Algorithmus betrachtet wurde, der an die Bit-Flip-Symmetrie angepasst ist.

Ähnlich wie beim „großen Bruder“ H-IFF ist auch für das Ising-Modell auf vollständigen binären Bäumen interessant, in wie weit Mutationen die Funktion optimieren können und wie Rekombination helfen kann. Dies knüpft an die Diskussion aus Abschnitt 1.2.3 und an die Betrachtungen von Fischer und Wegener [7] an, die die gleiche Frage für das Ising-Modell auf dem Ring untersuchen. Auf Bäumen ist jedoch zu erwarten, dass die Unterschiede im Nutzen von Mutation und Rekombination wesentlich größer ausfallen als auf dem Ring.

Wir werden in dieser Arbeit zeigen, dass der (1+1) EA eine exponentielle erwartete Optimierungszeit hat, während ein einfacher genetischer Algorithmus mit einer Populationsgröße von 2, Rekombination und Fitness Sharing als diversitätserhaltender Maßnahme mit einer polynomiellen erwarteten Optimierungszeit auskommt. Damit reiht sich die Funktion in die in Abschnitt 1.2.3 vorgestellte Klasse von Funktionen ein, für die Rekombination nachweisbar essenziell ist.

1.5 Methoden zur Analyse evolutionärer Algorithmen

In diesem Abschnitt sollen einige Methoden und Werkzeuge bereit gestellt werden, die wir zur Analyse evolutionärer Algorithmen verwenden werden. Ein Überblick, der viele der vorgestellten Methoden enthält, findet sich in [20].

1.5.1 Wahrscheinlichkeiten von Mutationen

Eine erste interessante Frage bei der Analyse des (1+1) EA ist, wie wahrscheinlich bestimmte Mutationen von k fest gewählten Bits sind. Diese Wahrscheinlichkeiten werden in den folgenden Analysen als bekannt vorausgesetzt und sollen daher hier ausführlich dargestellt werden.

Die Wahrscheinlichkeit, dass der (1+1) EA ein Bit flippt, entspricht der Mutationswahrscheinlichkeit, die wir auf $1/n$ festgelegt haben. Die Wahrscheinlichkeit, dass ein Bit nicht flippt, ist demnach die Gegenwahrscheinlichkeit $1 - 1/n$. Damit genau k fest gewählte Bits flippen, müssen diese Bits flippen und die übrigen $n - k$ Bits dürfen nicht flippen. Damit beträgt die Wahrscheinlichkeit einer bestimmten

Mutation von k Bits

$$\text{Prob(bestimmte } k\text{-Bit-Mutation)} = \frac{1}{n^k} \cdot \left(1 - \frac{1}{n}\right)^{n-k}.$$

Diese Formel können wir weiter auflösen, indem wir folgende Ungleichungen verwenden: Für alle $n \in \mathbb{N}$ gilt

$$\left(1 - \frac{1}{n}\right)^n \leq e^{-1} \leq \left(1 - \frac{1}{n}\right)^{n-1}.$$

Ein Beweis dieser Aussage findet sich in Motwani und Raghavan [18], Anhang B. Die Zahl e steht dabei wie auch im Folgenden für die Eulersche Zahl $e = 2{,}718\ldots$. Die Wahrscheinlichkeit, dass 0 Bits flippen, ist daher höchstens e^{-1} und für $k \geq 1$ gilt

$$\text{Prob(bestimmte } k\text{-Bit-Mutation)} = \frac{1}{n^k} \cdot \left(1 - \frac{1}{n}\right)^{n-k} \geq \frac{1}{en^k}.$$

Insgesamt gibt es $\binom{n}{k}$ verschiedene k-Bit-Mutationen. Die Wahrscheinlichkeit, dass k beliebige Bits flippen, beträgt daher

$$\text{Prob(beliebige } k\text{-Bit-Mutation)} = \binom{n}{k} \cdot \frac{1}{n^k} \cdot \left(1 - \frac{1}{n}\right)^{n-k}$$

Für $k \geq 1$ können wir diese Wahrscheinlichkeit nach oben abschätzen durch

$$\binom{n}{k} \cdot \frac{1}{n^k} \cdot \left(1 - \frac{1}{n}\right)^{n-k} \leq \frac{n(n-1)\ldots(n-k+1)}{k!} \cdot \frac{1}{n^k} \cdot e^{-1} \leq \frac{1}{ek!}.$$

1.5.2 Ungleichungen aus der Wahrscheinlichkeitstheorie

Bei der Analyse randomisierter Suchheuristiken haben wir es oft mit Wahrscheinlichkeiten zu tun, mit denen eine Zufallsvariable $X\colon \Omega \to \mathbb{R}$, deren Erwartungswert bekannt ist, eine gegebene Schranke a über- oder unterschreitet. Um diese Wahrscheinlichkeiten abzuschätzen, können wir einige verbreitete Ungleichungen aus der Wahrscheinlichkeitstheorie verwenden.

Eine sehr einfache Ungleichung ist die *Markoffsche Ungleichung*, deren einzige Voraussetzung ist, dass alle möglichen Werte für X nicht negativ sind.

Theorem 1 (Markoffsche Ungleichung). *Sei $X\colon \Omega \to \mathbb{R}$ eine Zufallsvariable, die nur nicht negative Werte annimmt und $a > 0$. Dann gilt*

$$\text{Prob}(X \geq a) \leq \frac{\text{E}(X)}{a}.$$

Beweis. Der Beweis ist leicht: Wir beziehen uns auf die Definition des Erwartungswertes und schätzen die Wahrscheinlichkeiten ab für die Fälle, in denen X Werte $x \geq a$ annimmt.

$$\begin{aligned} \mathrm{E}(X) &\overset{\text{Def.}}{=} \sum_{x} \mathrm{Prob}(X = x) \cdot x \\ &\geq \sum_{x \geq a} \mathrm{Prob}(X = x) \cdot x \\ &\geq \sum_{x \geq a} \mathrm{Prob}(X = x) \cdot a \\ &= a \cdot \sum_{x \geq a} \mathrm{Prob}(X = x) \end{aligned}$$

Diese Summe entspricht der Wahrscheinlichkeit $\mathrm{Prob}(X \geq a)$. Wir erhalten

$$\mathrm{E}(x) = a \cdot \mathrm{Prob}(X \geq a) \;\Leftrightarrow\; \mathrm{Prob}(X \geq a) \leq \frac{\mathrm{E}(X)}{a}.$$

□

Eine andere Gruppe von Ungleichungen sind so genannte *Chernoff-Schranken*. Diese ermöglichen wesentlich mächtigere Resultate, benötigen dafür aber auch speziellere Voraussetzungen als die Markoffsche Ungleichung. Und zwar betrachten wir bei Chernoff-Schranken eine Zufallsvariable X, die sich als Summe aus n einzelnen unabhängigen Variablen $X_1, \ldots, X_n$ schreiben lässt: $X = \sum_{i=1}^{n} X_i$. Die Variablen X_i stellen dabei *Indikatorvariablen* dar, die nur Werte $X_i \in \{0, 1\}$ annehmen.

Aufgrund der Linearität des Erwartungswertes gilt $\mathrm{E}(X) = \sum_{i=1}^{n} \mathrm{E}(X_i)$. Eine starke Abweichung vom Erwartungswert ist unwahrscheinlich, da dazu viele der unabhängigen Variablen X_i systematisch von ihrem Erwartungswert abweichen müssen. Es müssen also viele der unabhängigen Einzelereignisse für die Variablen X_i einseitig zugunsten kleinerer bzw. größerer Werte ausgehen. Dieses intuitive Argument spiegelt sich in folgenden Schranken wider, die u. a. in Motwani und Raghavan [18] bewiesen werden.

Theorem 2 (Chernoff-Schranken). *Seien $X_1, \ldots, X_n$ unabhängige Zufallsvariablen mit $X_i \in \{0, 1\}$ und $0 < \mathrm{Prob}(X_i = 1) < 1$ für alle $1 \leq i \leq n$ und $X := \sum_{i=1}^{n} X_i$.*

- *Für alle $\delta > 0$ gilt*

$$\mathrm{Prob}(X > (1 + \delta) \cdot \mathrm{E}(X)) < \left(\frac{e^{\delta}}{(1 + \delta)^{1+\delta}} \right)^{\mathrm{E}(X)}$$

- *Für alle δ mit $0 < \delta \leq 1$ gilt*

$$\mathrm{Prob}(X < (1 - \delta) \cdot \mathrm{E}(X)) < e^{-\mathrm{E}(X) \cdot \delta^2 / 2}$$

Wir können Chernoff-Schranken beispielsweise dazu verwenden, um die Zahl der Einsen in einem zufällig gewählten Suchpunkt $x \in \{0,1\}^n$ abzuschätzen, wie dies typischerweise bei der Initialisierung eines evolutionären Algorithmus geschieht. Die Zufallsvariable $X_i \in \{0,1\}$ spiegelt dann die Belegung von Bit x_i wider und $X = \sum_{i=1}^{n} X_i$ ist die Zahl der Einsen in x. Die Variablen X_i sind unabhängig voneinander und der Erwartungswert von X ist $\mathrm{E}(X) = n/2$.

Eine Anwendung der Schranken für ein konstantes δ mit $0 < \delta < 1$ ergibt: Die Wahrscheinlichkeit, dass wir um einen konstanten Faktor vom Erwartungswert abweichen, ist höchstens $e^{-\Omega(n)}$. Man kann sogar zeigen, dass eine Abweichung um einen additiven Term n^{ε} mit $\varepsilon > 1/2$ eine exponentiell kleine Wahrscheinlichkeit hat. Auf diese Aussage werden wir später zurück kommen.

1.5.3 Die Methode der Fitnessschichten

Die Methode der Fitnessschichten ist eine einfache, aber effektive Methode zur Bestimmung oberer Schranken für die erwartete Optimierungszeit einfacher evolutionärer Algorithmen mit Populationsgröße 1. Sie basiert darauf, dass bei evolutionären Algorithmen mit Plusstrategien die Fitness des aktuellen Suchpunkts monoton steigend ist.

Wir partitionieren den Suchraum in so genannte *Fitnessschichten*: disjunkte Teilmengen $A_1, \ldots, A_m$, die sich bezüglich der Fitnesswerte der enthaltenen Suchpunkte ordnen lassen. Alle Suchpunkte einer niedrigeren Schicht sollen dabei eine kleinere Fitness haben als alle Suchpunkte einer höheren Schicht. Eine solche Ordnung notieren wir mit

$$A <_f B \;:\Leftrightarrow\; \forall a \in A, b \in B\colon f(a) < f(b).$$

An die Partition des Suchraums stellen wir zwei Anforderungen. Zum einen soll gelten

$$A_1 <_f A_2 <_f \cdots <_f A_m.$$

Zum anderen soll die höchste Fitnessschicht A_m genau alle global optimalen Suchpunkte enthalten.

Da die Fitness des aktuellen Suchpunkts monoton steigt, werden Suchpunkte aus einer Fitnessschicht A_i nicht mehr betrachtet, sobald die Schicht A_i zu Gunsten einer höheren Schicht verlassen wurde. Genau dann, wenn die höchste Fitnessschicht A_m erreicht wird, haben wir ein globales Optimum erreicht.

Mit $s_i(x)$ für $x \in A_i$ bezeichnen wir die Wahrscheinlichkeit, ausgehend vom Suchpunkt x die Fitnessschicht A_i zu Gunsten einer besseren Schicht zu verlassen. Wir möchten nun vom konkreten Suchpunkt abstrahieren und eine untere Schranke für die Wahrscheinlichkeit, A_i zu verlassen, erhalten, die unabhängig von x ist. Dazu betrachten wir das Minimum über alle Suchpunkte x:

$$s_i := \min\{s_i(x) \mid x \in A_i\}$$

Die erwartete Zeit, bis dieses Ereignis eintritt, ist dann nach oben beschränkt durch den Kehrwert der Wahrscheinlichkeit, s_i^{-1}. Eine hinreichende Bedingung für das

Erreichen eines globalen Optimums ist, dass wir alle nicht optimalen Fitnessschichten $A_1, \ldots, A_{m-1}$ einmal verlassen. Wir erhalten folgende obere Schranke für die erwartete Optimierungszeit $\mathrm{E}(T)$:

$$\mathrm{E}(T) \leq 1 + \sum_{i=1}^{m-1} s_i^{-1}$$

Die 1 steht für die Auswertung des globalen Optimums.

Es sei angemerkt, dass Suchpunkte innerhalb einer Fitnessschicht ähnliche Fitnesswerte haben; die Tatsache, dass zwei Suchpunkte in der gleichen Fitnessschicht liegen, lässt i. A. jedoch keine Rückschlüsse auf die Lage der Suchpunkte im Suchraum zueinander zu. Insbesondere müssen die Mengen A_i nicht zusammenhängend sein. Im Fall, dass eine Fitnessfunktion lokale, nicht globale Optima enthält, kann es passieren, dass die Wahrscheinlichkeit, die Fitnessschicht eines solchen lokalen Optimums zu verlassen, sehr klein ist und die Methode der Fitnessschichten daher eine schlechte obere Schranke liefert. In diesem Fall sind andere Analysemethoden angebracht.

1.5.4 Die Methode des typischen Laufs

Eine alternative Methode, die erwartete Optimierungszeit abzuschätzen, ist die Betrachtung typischer Läufe. Auf manchen Fitnessfunktionen zeigen randomisierte Suchheuristiken „typische“ Verhaltensweisen, so dass viele Läufe nach einem wiederkehrenden Muster ablaufen. Diese Eigenschaft kann man sich zunutze machen und anstelle aller Läufe nur die typischen Läufe betrachten. Wenn die Wahrscheinlichkeit typischer Läufe nicht zu klein ist, lässt sich die erwartete Optimierungszeit durch die erwartete Optimierungszeit in typischen Läufen annähern.

Insbesondere zur Bestimmung unterer Schranken ist diese Methode oft praktisch. Wenn die Wahrscheinlichkeit, dass ein Lauf typisch ist, nach unten durch einen Wert $p(n)$ beschränkt ist, können wir die Optimierungszeit in nicht typischen Läufen durch die triviale untere Schranke 0 abschätzen. Mit bedingten Erwartungswerten erhalten wir eine untere Schranke für die erwartete Optimierungszeit $T(n)$ von

$$\mathrm{E}(T(n)) \geq p(n) \cdot \mathrm{E}(T(n) \mid \text{Lauf ist typisch}).$$

Ereignisse, die dazu führen, dass ein Lauf nicht typisch ist, bezeichnen wir oft als *Fehler* und die Wahrscheinlichkeit solcher Ereignisse als *Fehlerwahrscheinlichkeit*. Unser Ziel ist meist, Fehlerwahrscheinlichkeiten verschiedener Fehler durch exponentiell kleine Werte zu beschränken. In diesem Fall können wir verschiedene Ereignisse von Fehlern durch die Summe aller Fehlerwahrscheinlichkeiten abschätzen und erhalten meist wieder einen exponentiell kleinen Wert für die gesamte Fehlerwahrscheinlichkeit.

1.5.5 Das Coupon-Collector-Theorem

Um das Verhalten evolutionärer Algorithmen zu analysieren, kann es manchmal sinnvoll sein, Analogien zu anderen Problemen zu konstruieren und bereits bekannte Resultate aus anderen Gebieten auf die Analyse evolutionärer Algorithmen zu übertragen.

Eine solches Problem, zu dem wir eine Analogie ziehen können, ist das *Coupon-Collector-Problem*, das sich um das Sammeln von Sammelbildern (Coupons) dreht. Es gibt n verschiedene Typen von Coupons und wir ziehen nun Coupons uniform zufällig aus der Menge aller Typen. Die Frage ist, wie lange es dauert, bis wir von jedem Typ mindestens einen Coupon gesammelt haben.

Sei i die Zahl fehlender Coupon-Typen, dann beträgt die Wahrscheinlichkeit, dass der nächste Coupon unsere Sammlung verschiedener Typen vergrößert, genau i/n. Die erwartete Zahl zu ziehender Coupons, bis wir einen neuen Typ erhalten, ist gleich dem Kehrwert n/i. Sei T die Zeit, bis wir alle n verschiedenen Coupon-Typen beisammen haben. Wenn wir über alle i summieren, erhalten wir

$$\mathrm{E}(T) = \sum_{i=1}^{n} \frac{n}{i} = n \cdot \sum_{i=1}^{n} \frac{1}{i}.$$

Die Summe wird als *harmonische Reihe* bezeichnet; sie lässt sich nach unten durch $\ln n$ und nach oben durch $\ln n + 1$ abschätzen. Daher gilt

$$\mathrm{E}(T) = n \cdot \ln n + O(n).$$

Motwani und Raghaven [18] gehen noch einen Schritt weiter und zeigen, dass diese Abschätzung scharf ist; eine Abweichung nach oben oder unten vom Erwartungswert ist sehr unwahrscheinlich. Wir zitieren ihr Theorem [18] als *Coupon-Collector-Theorem*.

Theorem 3 (Coupon-Collector-Theorem). *Für jedes konstante $c \in \mathbb{R}$ gelten im Grenzübergang $n \to \infty$ die Aussagen*

$$\begin{aligned} \lim_{n\to\infty} \mathrm{Prob}(T > n \ln n + cn) &= 1 - e^{-e^{-c}} \\ \lim_{n\to\infty} \mathrm{Prob}(T \leq n \ln n - cn) &= e^{-e^{c}} \end{aligned}$$

Damit lässt sich die Wahrscheinlichkeit von Abweichungen sehr gut eingrenzen.

Das Coupon-Collector-Theorem (die Aussage über die erwartete Zeit $\mathrm{E}(T)$ mit eingeschlossen) können wir nun verwenden, um eine untere Schranke für die erwartete Optimierungszeit der randomisierten lokalen Suche auf allen pseudobooleschen Funktionen mit eindeutigem globalen Optimum zu erstellen.

Dazu stellen wir folgende Analogie her. Eine notwendige Bedingung, um ein globales Optimum zu erreichen, ist: Alle Bits, die den initialen Suchpunkt x^0 vom eindeutigen

Optimum x^{opt} unterscheiden, müssen irgendwann einmal geflippt werden. Wenn wir diese „falsch belegten“ Bits als Coupons auffassen, besagt die notwendige Bedingung, dass alle Typen von Coupons mindestens einmal gesammelt werden müssen. Dabei nehmen wir optimistischerweise an, dass alle einmal falsch belegten Bits nicht wieder „zurück flippen“, sobald sie einmal geflippt worden sind, so dass wir jeden Coupon-Typen nur einmal ziehen müssen.

Wenn wir nur Schritte der randomisierten lokalen Suche betrachten, in denen eines der falsch belegten Bits getroffen wird, erhalten wir genau das Szenario aus dem Coupon-Collector-Theorem.

Die Zahl verschiedener Coupon-Typen entspicht dem Hammingabstand $H(x^0, x^{\text{opt}})$ zwischen dem initialen Suchpunkt um dem Optimum. Der erwartete Hammingabstand ist $n/2$, da jedes Bit x_i^0 mit Wahrscheinlichkeit $1/2$ bei der Initialisierung die Belegung von x_i^{opt} erhält. Hier können wir die Methode des typischen Laufs anwenden und zeigen, dass man in einem typischen Lauf ein ausreichend großes Coupon-Collector-Szenario erhält.

Mit Chernoff-Schranken kann man zeigen, dass mit einer Wahrscheinlichkeit von $1 - e^{-\Omega(n^{1/3})}$ bei der zufälligen Initialisierung mindestens $n/2 - n^{2/3}$ Bits vom Optimum abweichend belegt sind, dass also mit überwältigender Wahrscheinlichkeit $H(x^0, x^{\text{opt}}) \geq n/2 - n^{2/3}$ ist. Wir erhalten ein Coupon-Collector-Szenario mit $s := n/2 - n^{2/3}$ Coupons. Selbst wenn wir in jedem Schritt ein vom Optimum abweichend belegtes Bit flippen, folgt in einem typischen Lauf eine untere Schranke von $s \ln s$ für die erwartete Zeit, bis alle s Bits mindestens einmal geflippt worden sind. Insgesamt folgt eine untere Schranke von

$$(1 - e^{-\Omega(n^{1/3})}) \cdot s \ln s = \Omega(n \log n)$$

für die erwartete Optimierungszeit der randomisierten lokalen Suche.

Wegener [21] überträgt diesen Ansatz auf den (1+1) EA und zeigt für den (1+1) EA ebenfalls eine untere Schranke $\Omega(n \log n)$ für die erwartete Optimierungszeit auf allen Funktionen mit eindeutigem Optimum. Wir gehen näher auf seinen Ansatz ein, da wir in Abschnitt 2.3 sehr ähnliche Beweise führen werden, um untere Schranken für den (1+1) EA auf dem Ising-Modell zu zeigen.

Beim (1+1) EA kann es vorkommen, dass in einem Schritt kein Bit geflippt wird oder mehr als ein Bit in einem Schritt flippt. Die erwartete Zahl flippender Bits ist jedoch bei einer Mutationswahrscheinlichkeit von $1/n$ gleich 1, so dass sich ähnliche Resultate erzielen lassen.

Allerdings gibt es ein Problem, wenn viele Bits in einem Schritt flippen. Und zwar nehmen wir im Coupon-Collector-Szenario an, dass die Typen von Coupons uniform zufällig gewählt werden. Wenn allerdings mehrere Bits auf einmal flippen, sind die ausgewählten Bits nicht mehr uniform zufällig verteilt, da ein Bit in einem Schritt nicht mehrfach zum Flippen ausgewählt werden kann. Wenn wir die Positionen aller in einem Schritt flippenden Bits nacheinander festlegen, haben beim letzten Bit alle bisher noch nicht gewählten Positionen eine höhere Wahrscheinlichkeit, gewählt

zu werden. Gegenüber dem Coupon-Collector-Szenario werden also einige Typen bevorzugt.

Als Lösung für dieses Problem argumentiert Wegener, dass in einem typischen Lauf mit überwältigender Wahrscheinlichkeit in polynomiell vielen Schritten nur höchstens $n^{1/2}$ Bits in einem Schritt flippen. Wenn wir ein Coupon-Szenario mit s Coupon-Typen betrachten, gilt: Auch beim letzten Bit, für den eine Position festgelegt wird, haben alle Positionen eine Wahrscheinlichkeit von höchstens $1/(s - n^{1/2})$, gewählt zu werden. Daher können wir dem Problem aus dem Weg gehen, indem wir anstelle eines Szenarios mit s Coupon-Typen ein kleineres Szenario mit $s - n^{1/2}$ Coupon-Typen betrachten. In einem typischen Lauf hat das resultierende Szenario immer noch eine Größe von $\Omega(n)$ Coupon-Typen, so dass sich auch hier eine Schranke von $\Omega(n \log n)$ zeigen lässt.

1.5.6 Verwendung von Potenzialfunktionen

Manche Fitnessfunktionen enthalten so genannte *Plateaus*, Mengen zusammenhängender Suchpunkte mit gleicher Fitness. Wenn ein solches Plateau erreicht wird, gibt die Fitnessfunktion im Allgemeinen keinerlei Hinweise, in welche Richtung der Optimierungsprozess laufen soll und wie das Plateau verlassen werden kann.

Dies ist zum einen eine Herausforderung für einen allgemeinen evolutionären Algorithmus, da er ohne Hinweise einen Weg finden muss, das Plateau zu verlassen. Zum anderen ist dies aber auch eine Herausforderung für die Analyse eines solchen Algorithmus, da man den Fortschritt des Algorithmus auf dem Plateau nicht an der Fitness festmachen kann.

Wenn die Fitnessfunktion nicht geeignet ist, um den Fortschritt des betrachteten Algorithmus zu messen, können wir in solchen Situationen *Potenzialfunktionen* verwenden: Anstelle der Fitnessfunktion betrachten wir eine andere Funktion

$$\varphi\colon S \to \mathbb{R},$$

die uns in bestimmten Situationen mehr Informationen liefert oder leichter zu analysieren ist als die Fitnessfunktion. Meist wird eine solche Potenzialfunktion so gewählt, dass die Maximierung (Minimierung) der Potenzialfunktion mit einem positiven Ereignis zusammenfällt. Im einfachsten Fall ist eine Potenzialfunktion monoton in der Fitness und ein Suchpunkt x mit optimalem Potenzial ist gleichzeitig ein Suchpunkt mit optimalem Fitnesswert.

Potenzialfunktionen können auch hilfreich sein, um Rechnungen zu vereinfachen, wenn die Fitnessfunktion zu unhandlichen Formeln führt. In Kapitel 3 werden wir in manchen Beweisen anstelle der Ising-Funktion eine Potenzialfunktion verwenden, die die Zahl der mit einer bestimmten Farbe gefärbten Knoten misst. Es wird sich heraus stellen, dass die Potenzialfunktion in manchen Fällen wesentlich einfacher zu analysieren ist als die echte Fitness.

Wenn die Potenzialfunktion nicht streng monoton in der Fitness ist, kann es passieren, dass sich das Potenzial des aktuellen Suchpunkts – anders als die Fitness

bei Plusstrategien – in einem akzeptierten Schritt verschlechtert. Es sei also explizit darauf hingewiesen, dass der Algorithmus nur mit der gegebenen Fitnessfunktion arbeitet, nicht mit Potenzialfunktionen. Potenzialfunktionen dienen lediglich der theoretischen Analyse und sind nicht Teil der Algorithmen.

1.5.7 Random Walks

Wir haben gesehen, dass es Sinn macht, Potenzialfunktionen zu verwenden, die nicht monoton in der Fitness sind. In so einem Fall kann es passieren, dass das Potenzial $\varphi(x)$ des aktuellen Suchpunkts in akzeptierten Schritten steigt oder sinkt. Das Potenzial vollführt eine so genannte *Irrfahrt* oder einen *Random Walk* auf der eindimensionalen Skala $\mathbb{R}$.

Hier werden wir nun eine Technik kennen lernen, die uns helfen kann, solche Random Walks zu analysieren, sofern der Random Walk die nötigen Voraussetzungen erfüllt. Gegeben sei eine Sequenz von unabhängigen Zufallsvariablen $X_1, \ldots, X_n \in \mathbb{R}$, die aus einer gemeinsamen Verteilung stammen und folgender Random Walk von Werten $S_0, \ldots, S_n$. Der Random Walk beginnt mit dem Startwert $S_0 := 0$ und errechnet sich ansonsten aus der Summe der Zufallsvariablen X_i:

$$S_n := \sum_{0 \leq i < n} X_i.$$

Beim Übergang von S_{n-1} zu S_n wird also einfach der Wert einer neuen Zufallsvariable X_n addiert.

Manche dieser Random Walks haben eine Tendenz in eine bestimmte Richtung. Man spricht hier von einer positiven *Drift*, wenn $\mathrm{E}(X_n) > 0$ ist. Die gemeinsame Verteilung aller X_i ist hier also einseitig zu Gunsten positiver Werte ausgelegt.

In der Folge der S_n bilden sich bei einer positiven Drift Rekordwerte, die größer sind als alle bisherigen S_i-Werte. S_n ist genau dann ein solcher Rekordwert, wenn gilt

$$\forall 0 \leq i < n \colon S_n > S_i.$$

Alte Rekorde werden dabei früher oder später durch neue, bessere Rekorde ersetzt. Die Folge der erreichten Rekordwerte bildet somit anschaulich eine Art Treppe, bei der jeder Rekord eine neue Stufe bildet und die Breite einer solchen Stufe die Differenz zwischen altem und neuem Rekord repräsentiert.

Die erwartete Zeit, bis sich ein neuer Rekordwert ergibt, lässt sich durch die so genannte *Waldsche Identität* errechnen.

Theorem 4 (Waldsche Identität). *Sei T die Zufallsvariable, die die Zeit angibt, bis ein neuer Rekord erreicht wird und H die Zufallsvariable, die die Höhe der dadurch konstruierten Treppenstufe beschreibt, also die Differenz zwischen neuem Rekord und altem Rekord. Falls $\mathrm{E}(X_n) > 0$, gilt*

$$\mathrm{E}(H) = \mathrm{E}(T) \cdot \mathrm{E}(X_n).$$

Dies ist äquivalent zu

$$\mathrm{E}(T) = \frac{\mathrm{E}(H)}{\mathrm{E}(X_n)}.$$

Ein Beweis dieser Aussage findet sich in [4].

Bei fester Drift ist also die erwartete Zeit, bis ein neuer Rekord erreicht wird, proportional zur erwarteten Erhöhung des Rekords. Wenn der Random Walk so gestaltet ist, dass sich der aktuelle Wert S_n maximal um den Wert 1 erhöht, gilt $\mathrm{E}(H) \leq 1$ und somit

$$\mathrm{E}(T) \leq \frac{1}{\mathrm{E}(X_n)}.$$

Droste, Jansen und Wegener [3] analysieren einen solchen Random Walk und präsentieren einen alternativen Beweis für die obige Ungleichung, der auf einer einfachen Rekursionsgleichung beruht. Wir werden in Abschnitt 3.2.2 auf diese Rekursionsgleichung zurück kommen.

Kapitel 2

Allgemeine Erkenntnisse zum Ising-Modell

In diesem Kapitel wollen wir einige neue allgemeine Erkenntnisse über das Ising-Modell heraus arbeiten. Viele dieser Erkenntnisse werden in den Kapiteln 3 und 4 Verwendung finden; wir schaffen uns daher bereits an dieser Stelle einige Grundlagen, durch die unsere Überlegungen in den nachfolgenden Kapiteln in einigen Punkten stark vereinfacht werden.

Wir beginnen in Abschnitt 2.1 mit einigen Überlegungen zu Eigenschaften der Ising-Funktion, um mit den Notationen und den Eigenarten der Ising-Funktion vertraut zu werden. Eine dieser Eigenarten ist, dass eine Mutation eines Knotens v nur lokale Auswirkungen auf die Fitness hat, nämlich nur auf die Beiträge aller zu v inzidenten Kanten. Daher können wir die Fitness eines durch Mutation entstandenen Nachkommens inkrementell aus der Fitness des Elters berechnen können, indem wir die einzelnen lokalen Effekte von Bitflips betrachten. Mit diesem Thema werden wir uns in Abschnitt 2.2 befassen.

In Abschnitt 2.3 zeigen wir eine allgemeine obere Schranke für die erwartete Optimierungszeit des (1+1) EA auf beliebigen Graphen sowie allgemeine untere Schranken für die erwarteten Optimierungszeiten der randomisierten lokalen Suche und des (1+1) EA. Schließlich gehen wir in Abschnitt 2.4 der Frage nach, welche Abhängigkeiten es zwischen den Färbungen einzelner Knoten gibt und wie sich diese Abhängigkeiten während des Laufs eines evolutionären Algorithmus entwickeln.

2.1 Eigenschaften der Ising-Funktion

Notation

Die *Ising-Funktion* Ising_G wurde bereits in Abschnitt 1.3 definiert. Da die Ising-Funktion die einzige Fitnessfunktion sein wird, die wir in dieser Arbeit verwenden werden, schreiben wir im Folgenden auch kurz f_G für Ising_G. Falls der Graph $G =$

(V, E), auf den sich die Ising-Funktion bezieht, aus dem Kontext ersichtlich ist, erlauben wir uns sogar, den Index G wegzulassen und nur von f zu sprechen.

Die Färbung eines Knotens i wird in einem Suchpunkt x eines evolutionären Algorithmus durch ein Bit x_i repräsentiert; die Werte 0 und 1 entsprechen den Farben 0 und 1. Der Einfachheit halber verwenden wir die Begriffe „Knoten“ und „Bit“ synonym; es ist also üblich, dass wir von einem „flippenden Knoten“ i und einem „mit Farbe c gefärbten Bit“ x_i sprechen. Gemeint ist dann natürlich, dass das zum Knoten i gehörige Bit x_i flippt bzw. dass der Knoten i, der zum Bit x_i gehört, mit Farbe c gefärbt ist.

Wir führen nun einige detailliertere Notationen ein, die wir in dieser Arbeit verwenden werden.

Definition 1. *Gegeben ein Graph $G = (V, E)$ mit $n := |V|$ Knoten und ein Suchpunkt $x \in \{0,1\}^n$. Den Beitrag einer Kante $e = \{i, j\}$ zur Fitness $f := \text{Ising}_G$ von x notieren wir mit*

$$f(x, e) := \begin{cases} 0, & \text{falls } x_i \neq x_j \\ 1, & \text{falls } x_i = x_j \end{cases}$$

Dabei nennen wir eine Kante e einfarbig, *falls $x_i = x_j$, und* zweifarbig, *falls $x_i \neq x_j$. Den Beitrag einer ganzen Kantenmenge $E' \subseteq E$ zur Fitness von x notieren wir mit*

$$f(x, E') := \sum_{e \in E'} f(x, e).$$

Die Fitness des Suchpunkts x ist dann

$$f(x) := f(x, E).$$

Klassifizierung

Die Ising-Funktion lässt sich wie folgt als quadratische pseudoboolesche Funktion schreiben. Der Beitrag einer Kante $e = \{i, j\}$ zur Fitness ist genau dann 1, wenn $x_i x_j = 1$ ist oder wenn $(1-x_i)(1-x_j) = 1$ ist. Daher lässt sich $f(x, e)$ auch schreiben als

$$\begin{aligned} f(x, e) &= x_i x_j + (1 - x_i)(1 - x_j) \\ &= 2x_i x_j - x_i - x_j + 1 \end{aligned}$$

Für die gesamte Fitness gilt dann

$$f(x) = \sum_{\{i,j\} \in E} (2x_i x_j - x_i - x_j + 1)$$

Es ist bekannt, dass die Optimierung quadratischer pseudoboolescher Funktionen NP-hart ist. Wegener und Witt [22] analysieren den (1+1) EA auf quadratischen pseudobooleschen Funktionen. Zum einen zeigen sie, dass der (1+1) EA auf quadratische Funktionen mit nichtnegativen Gewichten eine polynomielle erwartete Optimierungszeit hat. Zum anderen zeigen sie mit Hilfe zweier Beispielfunktionen, dass es

quadratische pseudoboolesche Funktionen mit negativen Gewichten gibt, auf denen der (1+1) EA eine exponentielle erwartete Optimierungszeit hat.

Die Ising-Funktion gehört zur zweiten Kategorie, der Klasse quadratischer Funktionen mit negativen Gewichten. Wir werden in den Abschnitten 3.3 und 4.2.2 zeigen, dass die Ising-Funktion auf bestimmten Graphen tatsächlich zu einer exponentiellen erwarteten Optimierungszeit des (1+1) EA führt.

Wertebereich

Der Wertebereich $\mathbb{W}$ der Ising-Funktion auf $G = (V, E)$ umfasst natürliche Zahlen (einschließlich 0) im Intervall $[0, |E|]$, $\mathbb{W} \subseteq (\mathbb{N} \cap [0, |E|])$. Die Ising-Funktion ist i. A. nicht surjektiv, z. B. ist der Wertebereich auf einem Dreieck $\mathbb{W} = \{1, 3\}$; die Werte 0 und 2 sind nicht realisierbar. Genau dann, wenn $0 \in \mathbb{W}$, ist der Graph 2-färbbar und damit bipartit. Jeder Suchpunkt x mit $f(x) = 0$ stellt dann eine entsprechende Färbung bzw. Partition des Graphen dar. Der maximale Fitnesswert ist stets $f(x) = |E|$, der zumindest für die Suchpunkte $x = 0^n$ und $x = 1^n$ angenommen wird.

Dichte Komponenten erhöhen die untere Schranke für die Fitness: enthält G beispielsweise eine Clique mit m Knoten als Subgraphen, ist die Fitness minimal, wenn genau $\lfloor m/2 \rfloor$ Knoten der Clique mit einer Farbe $c \in \{0, 1\}$ gefärbt sind und die anderen $\lceil m/2 \rceil$ Knoten mit der anderen Farbe $1 - c$ gefärbt sind. Daher beträgt in diesem Fall die Fitness $f(x)$ für alle x mindestens

$$f(x) \geq \binom{\lfloor m/2 \rfloor}{2} + \binom{\lceil m/2 \rceil}{2}.$$

Bei mehreren kantendisjunkten Subgraphen lassen sich entsprechende untere Schranken addieren.

Lokalität von Farbänderungen

Die Ising-Funktion hat die schöne Eigenschaft, dass Änderungen eines Knotens nur lokale Auswirkungen haben: Wenn sich an einem Knoten v die Farbe ändert, ändern sich die Beiträge zur Fitness nur in den zu v inzidenten Kanten. Alle zu v inzidenten Kanten, die vorher einfarbig waren, werden nun zweifarbig und umgekehrt; alle anderen Kanten behalten ihren alten Beitrag zur Fitness.

Mit dieser Erkenntnis lässt sich die Fitness von x' aus der Fitness von x inkrementell berechnen. Dieser Idee werden wir im folgenden Abschnitt nachgehen.

2.2 Die Betrachtung lokaler Fitnessveränderungen

Wir zeigen, dass sich die Fitnessänderung zwischen einem Suchpunkt x und seinem Nachkommen x' inkrementell berechnen lässt. Dies hilft nicht nur bei der Implementation eines Algorithmus, sondern auch bei der theoretischen Analyse, da wir

mit folgenden Techniken leichter entscheiden können, ob die Fitnessveränderung $f(x') - f(x) \geq 0$ ist, sprich ob x' als neuer Suchpunkt akzeptiert wird oder nicht.

Zuerst definieren wir lokale Fitnessveränderungen.

Definition 2. *Wir bezeichnen mit x^i für $1 \leq i \leq n$ den Suchpunkt, der aus $x = (x_1, \ldots, x_n)$ dadurch entsteht, dass x_i flippt: $x^i := (x_1, \ldots, x_{i-1}, \overline{x_i}, x_{i+1}, \ldots, x_n)$. Dann sei $\ell(x, x_i)$ für $1 \leq i \leq n$ die lokale Veränderung der Fitness, wenn nur das Bit x_i flippt:*

$$\ell(x, x_i) := f(x^i) - f(x).$$

Die Differenz $f(x^i) - f(x)$ lässt sich sehr leicht ausrechnen: Sei $\deg(v)$ der Grad eines Knotens v. Mit $\deg_x^+(v)$ notieren wir die Zahl der zum Knoten v inzidenten Kanten, die in x einfarbig sind; analog ist $\deg_x^-(v)$ die Zahl der zu v inzidenten und in x zweifarbigen Kanten.

Die zu i inzidenten Kanten, die in x zweifarbig sind, sind in x^i einfarbig, was gegenüber x einen Fitnessgewinn von $\deg_x^-(i)$ bringt. Die zu i inzidenten Kanten, die in x einfarbig sind, sind in x^i zweifarbig, was gegenüber x zu einem Fitnessverlust von $\deg_x^+(i)$ führt. Die Fitness ändert sich daher insgesamt um

$$\ell(x, x_i) \;=\; f(x^i) - f(x) \;=\; \deg_x^-(i) - \deg_x^+(i).$$

Die Analyse der randomisierten lokalen Suche auf dem Ising-Modell fällt gegenüber dem (1+1) EA oft sehr viel leichter, da sich in jedem Schritt nur an einem Knoten i die Farbe ändert und sich diese Fitnessveränderung sehr leicht berechnen lässt. Anhand des Vorzeichens kann man schnell ablesen, ob der erzeugte Nachkomme vom Algorithmus akzeptiert wird oder nicht.

Der (1+1) EA hingegen kann in einem Schritt die Farbe mehrerer Knoten ändern, so dass wir mehrere lokale Fitnessveränderungen berücksichtigen müssen. Um die gesamte Fitnessveränderung zu erhalten, kann man mehrere lokale Veränderungen aufaddieren, wenn sie verschiedene Zusammenhangskomponenten oder verschiedene Bereiche einer Zusammenhangskomponente betreffen. Dieser einfache Ansatz schlägt jedoch fehl, wenn die lokalen Änderungen sich überschneiden, d. h. wenn es eine Kante gibt, bei der beide Endpunkte flippen. In diesem Fall können die lokalen Veränderungen nicht mehr unabhängig voneinander betrachtet werden, da sie sich gegenseitig beeinflussen.

Als Beispiel nehmen wir an, dass beide Endpunkte einer in x einfarbigen Kante $e = \{i, j\}$ geflippt werden. Da beide Endpunkte flippen, ist e im Nachkommen wieder einfarbig gefärbt. Wenn wir aber nun die lokale Fitnessveränderung an i auswerten, betrachten wir ein Szenario, in dem nur i geflippt wird; die Kante e erscheint hier zweifarbig und geht dementsprechend negativ in $\ell(x, x_i)$ sein. Das gleiche gilt, wenn wir mit $\ell(x, x_j)$ ein Szenario betrachten, in dem nur j geflippt wird. Die lokalen Fitnessveränderungen schätzen also den Gesamteffekt beider Bitflips falsch ein.

Es gibt jedoch zwei Möglichkeiten, wie man mehrere Bitflips auf die Betrachtung einzelner lokaler Änderungen zurück führen kann.

1. Wir führen die Bitflips nacheinander aus und summieren die lokalen Fitnessveränderungen auf.
2. Wir führen die Bitflips parallel und unabhängig voneinander aus und korrigieren die Fehler, die durch Abhängigkeiten entstehen.

Beim ersten Ansatz betrachtet man eine Folge verschiedener Färbungen, die durch die nacheinander ausgeführten Bitflips entstehen. Hingegen betrachtet man beim zweiten Ansatz nur die Färbung des Elters, so dass man die lokalen Fitnessveränderungen $\ell(x, \cdot)$ nur einmal berechnen muss.

Wir wollen im Folgenden nachweisen, dass beide Ansätze korrekt sind.

Nacheinander ausgeführte Bitflips

Lemma 1. *Gegeben ein Suchpunkt $x \in \{0,1\}^n$ und ein Suchpunkt x', der durch Mutation aus x entsteht; $i_1, \ldots, i_k$ sei eine beliebige Reihenfolge der Indizes der Bits, die durch diese Mutation geflippt werden. Sei x^j für $0 \leq j \leq k$ der Suchpunkt, in dem gegenüber x die Bits $i_1, \ldots, i_j$ geflippt sind:*

$$x_l^j = \begin{cases} \overline{x_l}, & \text{falls } l \in \{i_1, \ldots, i_j\} \\ x_l, & \text{falls } l \notin \{i_1, \ldots, i_j\} \end{cases}$$

Dann lässt sich die Fitness von x' wie folgt aus x inkrementell berechnen:

$$f(x') = f(x) + \sum_{j=1}^{k} \ell(x^{j-1}, x_{i_j}).$$

Beweis. Nach Definition entsteht x^j aus x^{j-1} dadurch, dass das Bit i_j flippt. Aus der Definition von $\ell(x^{j-1}, x_{i_j})$ folgt

$$\ell(x^{j-1}, x_{i_j}) = f(x^j) - f(x^{j-1}).$$

Zweitens gilt $x = x^0$ und $x^k = x'$. Insgesamt gilt daher

$$\begin{aligned} & f(x) + \sum_{j=1}^{k} \ell(x^{j-1}, x_{i_j}) \\ = \; & f(x) + \sum_{j=1}^{k} (f(x^j) - f(x^{j-1})) \\ = \; & f(x) - f(x^0) + f(x^k) = f(x'). \end{aligned}$$

□

Parallel ausgeführte Bitflips

Wir haben bereits an unserem kleinen Beispiel gesehen, dass der Fitnessbeitrag einer in x einfarbigen Kante $e = \{i, j\}$ mit zwei flippenden Endpunkten durch lokale Fitnessveränderungen falsch abgeschätzt wird. Dies gilt auch in dem Fall, dass e zweifarbig gefärbt ist; hier erscheint e in den Szenarien der lokalen Fitnessveränderungen als einfarbig und geht positiv in $\ell(x, x_i)$ und $\ell(x, x_j)$ ein. Der Fehler kann daher in zwei Richtungen auftreten: Ist e einfarbig, wird der Fitnessbeitrag $f(x, e)$ um 2 unterschätzt, ist e hingegen zweifarbig, wird der Fitnessbeitrag $f(x, e)$ um 2 überschätzt. Der Faktor 2 rührt daher, dass wir den Beitrag von e sowohl in $\ell(x, x_i)$ als auch in $\ell(x, x_j)$ falsch anrechnen.

Um den Fehler für $f(x, e)$ zu korrigieren, werden wir zu den lokalen Fitnessveränderungen einen Korrekturterm addieren, der für $f(x, e) = 1$ den Wert $+2$ annimmt und für $f(x, e) = 0$ den Wert -2. Geeignet hierfür ist der Term

$$4f(x, e) - 2.$$

Einen solchen Term müssen wir für jede Kante anrechnen, bei der beide Endpunkte flippen. Nach diesen vorbereitenden Worten sollte die Aussage des folgenden Lemmas nicht überraschen.

Lemma 2. *Gegeben ein Graph $G = (V, E)$ mit $n := |V|$ Knoten, ein Suchpunkt $x \in \{0, 1\}^n$ und ein Suchpunkt x', der durch Mutation aus x entsteht. Sei $V' \subseteq V$ die Menge der Knoten, deren Bits in dieser Mutation flippen und E' die Kantenmenge, die durch V' induziert wird, $E' := E \cap \{\{i, j\} \mid i \in V', j \in V'\}$. Dann gilt für $f := \text{Ising}_G$:*

$$f(x') - f(x) = \sum_{i \in V'} \ell(x, x_i) + 4f(x, E') - 2|E'|$$

Beweis. Sei x^i der Suchpunkt aus der Definition von $\ell(x, x_i)$, der aus x dadurch entsteht, dass genau das Bit x_i flippt. Als erstes führen wir $\ell(x, x_i)$ für $i \in V'$ auf einen Ausdruck zurück, der nicht mehr von x^i, sondern nur noch von x und x' abhängt. Dazu nutzen wir zwei Eigenschaften von x^i gegenüber x und x' aus:

- Für Kanten $\{i, j\} \in E$ mit $i \in V', j \notin V'$ (sprich: nur der Knoten i flippt) nutzen wir aus, dass i und j in x' genau so gefärbt sind wie in x^i:

$$f(x^i, \{i, j\}) = f(x', \{i, j\}).$$

- Für Kanten $\{i, j\} \in E$ mit $i, j \in V'$ nutzen wir aus, dass i in x anders gefärbt ist als in x^i, während j in x und x^i gleich gefärbt ist:

$$f(x^i, \{i, j\}) = 1 - f(x, \{i, j\}).$$

Wir werden nun den Term $\ell(x, x_i)$ umformen. Zunächst einmal gilt nach Definition von $\ell(x, x_i)$, wenn wir die Fitness in die Beiträge aller Kanten zerlegen:

$$\begin{aligned}\ell(x, x_i) &= f(x^i) - f(x) \\ &= \sum_{\{i,j\} \in E} \big(f(x^i, \{i,j\}) - f(x, \{i,j\})\big) .\end{aligned}$$

Wegen der Lokalität der Farbänderungen unterschieden sich x^i und x nur in den Beiträgen der zu i inzidenten Kanten. In der obigen Summe summieren sich die Beiträge aller anderen Kanten zu 0. Damit können wir $\ell(x, x_i)$ für $i \in V'$ schreiben als

$$\ell(x, x_i) = \sum_{j \in \mathrm{Adj}(i)} \big(f(x^i, \{i,j\}) - f(x, \{i,j\})\big) .$$

Nun spalten wir die Summe auf in Kanten, bei denen nur i flippt ($j \in \mathrm{Adj}(i) \setminus V'$), und Kanten, bei denen beide Endpunkte flippen ($j \in \mathrm{Adj}(i) \cap V'$). Anschließend wenden wir auf die Teilsummen die beiden oben erwähnten Eigenschaften von x^i an.

$$\begin{aligned}&\ell(x, x_i) \\ = &\sum_{j \in \mathrm{Adj}(i)} \big(f(x^i, \{i,j\}) - f(x, \{i,j\})\big) \\ = &\sum_{j \in \mathrm{Adj}(i) \setminus V'} \big(f(x^i, \{i,j\}) - f(x, \{i,j\})\big) + \sum_{j \in \mathrm{Adj}(i) \cap V'} \big(f(x^i, \{i,j\}) - f(x, \{i,j\})\big) \\ = &\sum_{j \in \mathrm{Adj}(i) \setminus V'} \big(f(x', \{i,j\}) - f(x, \{i,j\})\big) + \sum_{j \in \mathrm{Adj}(i) \cap V'} (1 - 2f(x, \{i,j\})) \\ = &\sum_{j \in \mathrm{Adj}(i) \setminus V'} \big(f(x', \{i,j\}) - f(x, \{i,j\})\big) - \sum_{j \in \mathrm{Adj}(i) \cap V'} (2f(x, \{i,j\}) - 1)\end{aligned}$$

Wenn wir die zweite Summe zu beiden Seiten addieren, erhalten wir für $i \in V'$ die Gleichung

$$\ell(x, x_i) + \sum_{j \in \mathrm{Adj}(i) \cap V'} (2f(x, \{i,j\}) - 1) = \sum_{j \in \mathrm{Adj}(i) \setminus V'} \big(f(x', \{i,j\}) - f(x, \{i,j\})\big) \quad (*)$$

Diese Gleichung lässt sich wie folgt interpretieren. Kanten mit zwei flippenden Knoten $i, j \in V'$ sind in x' genau so gefärbt wie in x. Die Summe auf der rechten Seite der Gleichung beinhaltet alle zu i benachbarten Knoten, die nicht flippen. Also beschreibt die rechte Seite der Gleichung die Veränderung der Fitness zwischen x und x' am Knoten i.

Die linke Seite der Gleichung lässt sich wie folgt interpretieren. Wenn wir die lokale Fitnessveränderung $\ell(x, x_i)$ betrachten, erhalten wir nicht immer die Fitnessveränderung am Knoten i, sondern wir machen für jede zu i inzidente Kante mit zwei flippenden Endpunkten einen Fehler von ± 1, der durch einen Korrekturterm $2f(x, e) - 1$ ausgeglichen wird. (Gegenüber dem Korrekturterm $4f(x, e) - 2$ aus den

Vorbetrachtungen ist dies nur die halbe Miete, da wir nur die lokale Fitnessveränderung $\ell(x, x_i)$ an einem statt zwei Endpunkten betrachten.)

Diese Erkenntnis für einen flippenden Knoten $i \in V'$ lässt sich nun leicht auf alle flippenden Knoten übertragen. Wir nutzen aus, dass die Fitnesswerte von x und x' sich nur durch die Beiträge von Kanten unterscheiden, die zu genau einem Knoten in V' inzident sind. In der Differenz $f(x') - f(x)$ summieren sich die Beiträge aller anderen Kanten zu 0. Es folgt

$$\begin{aligned} f(x') - f(x) &= \sum_{e \in E} \big(f(x', e) - f(x, e)\big) \\ &= \sum_{i \in V'} \sum_{j \in \mathrm{Adj}(i) \setminus V'} \big(f(x', \{i, j\}) - f(x, \{i, j\})\big) \,. \end{aligned}$$

Nun setzen wir Gleichung $(*)$ ein und erhalten

$$\begin{aligned} f(x') - f(x) &= \sum_{i \in V'} \left(\ell(x, x_i) + \sum_{j \in \mathrm{Adj}(i) \cap V'} (2f(x, \{i, j\}) - 1) \right) \\ &= \sum_{i \in V'} \ell(x, x_i) + \sum_{i \in V'} \sum_{j \in \mathrm{Adj}(i) \cap V'} (2f(x, \{i, j\}) - 1). \end{aligned}$$

Die Doppelsumme zählt alle Kanten in E' genau zwei Mal. Daher können wir den Ausdruck weiter vereinfachen zu

$$\begin{aligned} f(x') - f(x) &= \sum_{i \in V'} \ell(x, x_i) + 2 \sum_{\{i,j\} \in E'} (2f(x, \{i, j\}) - 1) \\ &= \sum_{i \in V'} \ell(x, x_i) + 4f(x, E') - 2|E'|. \end{aligned}$$

□

In den meisten Schritten flippen nur wenige Knoten und die Zahl der falsch angerechneten Kanten ist gering. Wenn wir daher zu $f(x)$ alle lokalen Fitnessveränderungen addieren, erhalten wir eine gute Näherung für die Fitness $f(x')$ des Nachkommens. Dies halten wir in folgendem Korollar fest.

Korollar 1. *Aus Lemma 2 folgt direkt*

$$-2|E'| \;\leq\; f(x') - \left(f(x) + \sum_{i \in V'} \ell(x, x_i) \right) \;\leq\; 2|E'|$$

Der Fehler ist also höchstens quadratisch in der Zahl flippender Bits, auf spärlich besetzten Graphen sogar noch geringer.

2.3 Allgemeine Schranken für die erwartete Optimierungszeit

Auch ohne Kenntnis des Graphen lassen sich einfache allgemeine obere und untere Schranken zeigen. Dieser Aufgabe wollen wir uns nun widmen.

Zuerst zeigen wir eine obere Schranke, die darauf basiert, dass der (1+1) EA in jedem Schritt mit einer kleinen Wahrscheinlichkeit direkt zu einem globalen Optimum springt. Dabei hilft uns die Bit-Flip-Symmetrie weiter, da es stets mehrere globale Optima gibt und der Abstand zum nächsten globalen Optimum nicht allzu groß werden kann.

Theorem 5. *Gegeben ein beliebiger Graph $G = (V, E)$ mit $n := |V|$ Knoten. Die erwartete Optimierungszeit des (1+1) EA auf $f := \text{Ising}_G$ ist nach oben beschränkt durch $O(n^{n/2})$.*

Beweis. Die Suchpunkte 0^n und 1^n sind auf jedem Graphen globale Optima. Sei x der aktuelle Suchpunkt des (1+1) EA. Da $H(x, 0^n) + H(x, 1^n) = n$ ist, ist das Minimum aus beiden Hammingabständen höchstens $n/2$. Daher ist für jeden Suchpunkt x der Hammingabstand zum nächsten globalen Optimum höchstens $n/2$.

Die Wahrscheinlichkeit, dass aus dem jeweils aktuellen Suchpunkt durch Mutation direkt ein globales Optimum erzeugt wird, ist daher mindestens

$$\frac{1}{n^{n/2}} \cdot \left(1 - \frac{1}{n}\right)^{n/2} \geq \frac{1}{n^{n/2}} \cdot \left(1 - \frac{1}{n}\right)^{n-1} \geq \frac{1}{e \cdot n^{n/2}}$$

und die erwartete Zeit, bis dieses Ereignis eintritt, ist $O(n^{n/2})$. □

Für allgemeine pseudobooleschen Funktionen mit n Variablen kann man analog eine allgemeine Schranke n^n zeigen [3]. Die Ising-Funktion enthält mindestens zwei zueinander komplementäre Optima, was also bei dieser sehr allgemeinen Aussage einen greifbaren Vorteil bringt.

Als nächstes untersuchen wir allgemeine untere Schranken. Hier zeigt sich, dass das zweite Optimum keinen wesentlichen Vorteil bietet: Verglichen mit den allgemeinen unteren Schranke $\Omega(n \log n)$ für die randomisierte lokale Suche und den (1+1) EA auf pseudobooleschen Funktionen mit n Bits und einem eindeutigem globalen Optimum sind die folgenden unteren Schranken für beide Algorithmen nur um jeweils einen konstanten Faktor größer.

Theorem 6. *Gegeben ein beliebiger zusammenhängender Graph $G = (V, E)$ mit $n := |V|$ Knoten. Die erwartete Optimierungszeit der randomisierten lokalen Suche und des (1+1) EA auf $f := \text{Ising}_G$ ist nach unten beschränkt durch $\Omega(n \log n)$.*

Beweis. Zuerst zeigen wir, dass 0^n und 1^n die einzigen globalen Optima sind. Angenommen, im aktuellen Suchpunkt x gibt es einen Knoten i mit $x_i = 0$ und einen

Knoten j mit $x_j = 1$. Da der Graph zusammenhängend ist, gibt es einen Pfad zwischen i und j. Auf diesem Pfad gibt es mindestens eine Kante, deren Endpunkte nicht gleich gefärbt sind, da ansonsten $x_i = x_j$ wäre. Also ist jeder Suchpunkt $x \notin \{0^n, 1^n\}$ nicht optimal.

Die untere Schranke $\Omega(n \log n)$ für die erwartete Optimierungszeit folgt dann aus einer Anwendung des Coupon-Collector-Theorems. Wir haben in Abschnitt 1.5.5 den Beweis von Wegener [21] vorgestellt, der das Coupon-Collector-Theorem und die Methode des typischen Laufs anwendet, um für den (1+1) EA auf Funktionen mit eindeutigem Optimum eine untere Schranke $\Omega(n \log n)$ zu zeigen.

Sein Beweis lässt sich direkt auf unser Szenario mit zwei komplementären Optima übertragen: Bei Funktionen mit eindeutigem Optimum hat der bei der Initialisierung ausgewürfelte Suchpunkt x mit einer Wahrscheinlichkeit von $1 - e^{-\Omega(n^{1/3})}$ nur höchstens $n/2 - n^{2/3}$ Bitpositionen mit dem Optimum gemeinsam. Dies gilt genau so für die Ising-Funktion; hier können wir Chernoff-Schranken zwei Mal anwenden, um die Zahl der Nullen und die Zahl der Einsen durch $n/2 - n^{2/3}$ zu beschränken. Wir erhalten das Resultat, dass der initiale Suchpunkt ebenfalls mit einer Wahrscheinlichkeit von $1 - e^{-\Omega(n^{1/3})}$ nur höchstens $n/2 - n^{2/3}$ Bitpositionen mit beiden Optima gemeinsam hat.

Wie in Abschnitt 1.5.5 erhalten wir ein Coupon-Collector-Szenario mit $s := n/2 - n^{2/3}$ Coupons. Gegenüber der Analyse mit einem eindeutigen Optimum gibt es jedoch einen kleinen Unterschied: Beim Ising-Modell mit den Optima 0^n und 1^n kann sich der Algorithmus „aussuchen“, ob er Nullen oder Einsen sammeln möchte. Sei T_0 die Zeit, bis der Algorithmus mindestens $n/2 - n^{2/3}$ Nullen gesammelt hat und T_1 die Zeit, bis der Algorithmus mindestens $n/2 - n^{2/3}$ Einsen gesammelt hat. Dann gilt für die Optimierungszeit T und alle $t \in \mathbb{N}$:

$$\text{Prob}(T \leq t) \;\leq\; \text{Prob}(T_0 \leq t) + \text{Prob}(T_1 \leq t) \;\leq\; 2 \cdot \text{Prob}(T_1 \leq t).$$

Dabei gilt die zweite Ungleichung aus Symmetriegründen. Die Wahrscheinlichkeit $\text{Prob}(T_1 \leq t)$ lässt sich mit dem Coupon-Collector-Theorem (Theorem 3) beliebig scharf abgrenzen, so dass der zusätzliche Faktor 2 nicht ins Gewicht fällt.

Insgesamt lassen sich daher für beide Algorithmen Schranken von $\Omega(n \log n)$ zeigen. □

Die im Folgenden Matching-Methode genannte Methode wurde von Fischer [5] erdacht und ist auch bei nicht zusammenhängenden Graphen anwendbar.

Ein *Matching* auf einem Graphen $G = (V, E)$ ist eine Teilmenge der Kantenmenge $M \subseteq E$, die die Eigenschaft hat, dass jeder Knoten $v \in V$ nur zu höchstens einer Matching-Kante, einer Kante in M, inzident ist. Diese Eigenschaft führt dazu, dass wir Operationen auf den Matching-Kanten unabhängig voneinander betrachten können.

Fischer [5] verwendet diese Idee, um eine untere Schranke für eine spezielle Graphklasse zu zeigen. Wir wollen seine Ideen verwenden, um eine allgemeinere Aussage herzuleiten. Der folgende Beweis basiert anders als der Beweis von Fischer auf dem Coupon-Collector-Theorem.

Theorem 7. *Gegeben ein beliebiger Graph $G = (V, E)$ und ein Matching M der Größe $m := |M|$ auf G. Dann ist die erwartete Optimierungszeit der randomisierten lokalen Suche und des (1+1) EA auf $f := \text{Ising}_G$ nach unten beschränkt durch $\Omega(n \log m)$.*

Beweis. Zuerst behandeln wir zwei Sonderfälle. Falls $m = 1$, ist $\log m = 0$ und es ist nichts zu zeigen. Falls $m \geq 2$ und $m = O(1)$, folgt die untere Schranke $\Omega(n \log m) = \Omega(n)$ trivialerweise, da beide Algorithmen mit Wahrscheinlichkeit $1 - 2 \cdot 2^{-n}$ mit einem suboptimalen Suchpunkt starten und dann mindestens eines von $m = O(1)$ Bits flippen müssen, um ein Optimum zu erreichen. Die erwartete Wartezeit auf ein solches Ereignis ist $\Omega(n)$. Wir nehmen daher im Folgenden an, dass $m = \omega(1)$.

Die Wahrscheinlichkeit, dass bei der zufälligen Initialisierung eine Matching-Kante zweifarbig gefärbt wird, ist 1/2. Aufgrund der Matching-Eigenschaft sind die Wahrscheinlichkeiten für alle Matchingkanten unabhängig; die erwartete Zahl zweifarbiger Matching-Kanten ist $m/2$. Daher können wir mit Chernoff-Schranken zeigen: Die Wahrscheinlichkeit, dass weniger als $m/3$ Kanten des Matchings zweifarbig sind, ist beschränkt durch $e^{-(m/36)} = o(1)$.

Wir verwenden die Methode des typischen Laufs und nehmen im Folgenden an, dass mindestens $m/3$ Kanten des Matchings zweifarbig sind. Eine notwendige Bedingung, um ein globales Optimum zu erreichen, ist, dass alle zweifarbigen Matching-Kanten einfarbig gefärbt werden. Hierfür ist wiederum notwendig, dass alle zweifarbigen Matching-Kanten mindestens einmal von Bitflips getroffen werden.

Wir können aus diesem Szenario ein Szenario für das Coupon-Collector-Theorem erstellen, indem wir $m/3$ zweifarbige Matching-Kanten auswählen und sie uns als Coupons vorstellen. Ein Coupon wird vom Algorithmus eingesammelt, wenn er einen Endpunkt einer Kante flippt. (Wir nehmen optimistischerweise an, dass die Kante danach nie wieder zweifarbig wird.) Die Matching-Eigenschaft sorgt dafür, dass die Coupons unabhängig voneinander sind, da jeder Knoten nur zu höchstens einem Coupon gehören kann. Damit erhalten wir ein Szenario für das Coupon-Collector-Theorem mit $m/3$ Coupons.

Der (1+1) EA kann in einem Schritt mehrere Coupons ziehen; dabei tritt wieder das Problem auf, dass die Wahl der in einem Schritt flippenden Bits nicht mehr unabhängig ist. Wir unterscheiden die folgenden zwei Fälle.

Fall 1: $m = \omega(1)$ und $m < n^{1/4}$.
Die Wahrscheinlichkeit, dass in einem Schritt mindestens zwei Bits des Matchings flippen, ist $O(n^{-3/2})$. Die Wahrscheinlichkeit, dass dies mindestens einmal in $o(n^{3/2})$ Schritten passiert, lässt sich durch $o(1)$ abschätzen. Wir nehmen an, dass dies nicht passiert und verbleiben somit in einem Coupon-Collector-Szenario mit $m/3$ Coupons.

Fall 2: $m \geq n^{1/4}$.
Hier verwenden wir das Argument von Wegener [21] mit leicht abgewandelten Werten: Mit einer Wahrscheinlichkeit von $o(1)$ kommt es in polynomiell vielen

Schritten nicht vor, dass mindestens $m/12 \geq n^{1/4}/12$ Bits in einem Schritt flippen. Wenn wir annehmen, dass dies nicht eintritt, reicht es, wenn wir uns auf ein Coupon-Collector-Szenario mit $m/3 - m/12 = m/4$ Coupons beschränken.

Wir gehen also davon aus, dass wir $m/4$ Coupon-Typen sammeln müssen. Sei T die Zahl gezogener Coupons, bis wir alle Typen von Coupons gesammelt haben. Der Erwartungswert $\mathrm{E}(T)$ ist, wie in Abschnitt 1.5.5 gezeigt, in der Größenordnung $\Theta(m \log m)$. Wir wählen nun $c := (\ln(m/4))/2$ und nach dem Coupon-Collector-Theorem (Theorem 3) folgt: Die Wahrscheinlichkeit, dass $T \leq (m/4) \cdot \ln(m/4) - cm/4 = (m/8) \cdot \ln(m/4)$ ist, konvergiert für $n \to \infty$ gegen $e^{-e^c} = e^{-m^{1/2}} = o(1)$. Für ausreichend große n können wir die Wahrscheinlichkeit nach oben durch $o(1)$ abschätzen. In einem typischen Lauf benötigen wir also mehr als $(m/8) \cdot \ln(m/4)$ Ziehungen von Coupons.

Um einen der $m/4$ Coupons zu ziehen, muss ein flippendes Bit einen Endpunkt einer zweifarbigen Matching-Kante treffen. Da die zu Coupons gehörigen Kanten ein Matching bilden, kommen hier genau $m/2$ Knoten in Frage.

Die erwartete Zahl der in einem Schritt gezogenen Coupons ist für beide Algorithmen durch $m/(2n)$ nach oben beschränkt. Die erwartete Zahl von Ziehungen nach $(n/8) \cdot \ln(m/4)$ Schritten ist daher höchstens $(m/16) \cdot \ln(m/4)$. Mit Chernoff-Schranken kann man zeigen: Die Wahrscheinlichkeit, dass in $(n/8) \cdot \ln(m/4)$ Schritten mindestens $(m/8) \cdot \ln(m/4)$ der Coupon-Bits flippen, ist exponentiell klein.

Insgesamt erhalten wir für dieses Szenario: Die Wahrscheinlichkeit, in $(n/8) \cdot \ln(m/4)$ Schritten alle Coupons zusammen zu haben, ist höchstens $o(1)$. Daraus folgt die untere Schranke von $(1 - o(1)) \cdot (n/8) \cdot \ln(m/4) = \Omega(n \log m)$ für die erwartete Optimierungszeit. □

Wenn also ein Graph G ein Matching der Größe $\Omega(n^\varepsilon)$ enthält für eine Konstante $\varepsilon > 0$, folgt aus Theorem 7 ebenfalls eine untere Schranke $\Omega(n \log n)$.

Es gibt jedoch auch Graphen, bei denen dies nicht der Fall ist, und auf denen Theorem 6 bessere Resultate liefert. Als Beispiel sei hier ein sternförmiger Graph $G = (V, E)$ genannt mit Knotenmenge $V = \{1, \ldots, n\}$, bei dem die Knoten $2, \ldots, n$ nur zum Knoten 1 adjazent sind. Ein maximales Matching besteht nur aus einer Kante, so dass die Matching-Methode nur eine triviale untere Schranke von $\Omega(n \log 1) = 0$ liefert. Da G zusammenhängend ist, gilt nach Theorem 6 jedoch eine untere Schranke von $\Omega(n \log n)$.

2.4 Abhängigkeiten zwischen Teilfärbungen

Die Ising-Funktion gibt einem evolutionären Algorithmus starke Hinweise dahin gehend, benachbarte Knoten mit der gleichen Farbe zu färben. Diese lokalen Effekte wirken sich mit der Zeit auch auf weiter entfernte Knoten aus, wenn ganze Teilgraphen die Tendenz entwickeln, sich mit der gleichen Farbe zu färben. Eine interessante

Frage ist daher, welchen Einfluss die Färbung eines Knotens u auf die Färbung eines anderen Knotens v über die Zeit hat.

Es ist intuitiv klar, dass zwei Knoten u und v sich nicht gegenseitig beeinflussen, wenn sie in verschiedenen Zusammenhangskomponenten $V_u \dot{\cup} V_v$ liegen. Aufgrund der Bit-Flip-Symmetrie ist eine Teilfärbung von V_u, in der u mit 1 gefärbt, genau so wahrscheinlich wie eine komplementäre Teilfärbung von V_u, die u mit 0 färbt. (Dies werden wir noch formal beweisen.) Aber auch wenn sie in der gleichen Zusammenhangskomponente liegen, sind die Färbungen am Anfang des Optimierungsprozesses unabhängig, da die Initialisierung alle Knoten unabhängig voneinander färbt.

Wenn u und v weit auseinander liegen, kann es eine Zeit lang dauern, bis sich der Einfluss der Färbung von u auf die Färbung von v auswirkt, insbesondere wenn es nur wenige Pfade zwischen u und v gibt. Hier spielt also auch die Dichte der Pfade zwischen u und v eine Rolle.

Wir werden im Folgenden zeigen, dass die Färbungen einzelner ausgezeichneter Knoten in verschiedenen Zusammenhangskomponenten unabhängig voneinander sind. Für jede Zusammenhangskomponente betrachten wir dabei nur einen Knoten, der die Zusammenhangskomponente repräsentiert und den wir als *Pivot-Knoten* bezeichnen wollen. Wir werden mit Hilfe der Bit-Flip-Symmetrie zeigen, dass die Färbung dieser Pivot-Knoten stochastisch unabhängig sind. Darauf aufbauend definieren wir dann Mengen, die für jeden Zeitpunkt t mögliche Abhängigkeiten angeben und aus denen sich ablesen lässt, welche Pivot-Knoten stochastisch unabhängig voneinander gefärbt sind.

Es sei darauf hingewiesen, dass es bei unseren Betrachtungen für den (1+1) EA zwingend notwendig ist, sich auf einzelne Pivot-Knoten zu beschränken. Die Färbung zweier Knoten u_1, u_2 innerhalb einer Zusammenhangskomponente V_u ist im Allgemeinen nicht unabhängig von der Färbung zweier Knoten v_1, v_2 innerhalb einer anderen Zusammenhangskomponente V_v. Wenn in V_v kein Fitnessgewinn möglich ist, haben u_1 und u_2 meist eine gute Tendenz, die gleiche Farbe anzunehmen. Wenn es jedoch in V_v hohe Fitnessgewinne gibt, kann dies einen störenden Einfluss auf den Färbungsprozess in V_u haben: Schritte des (1+1) EA, die die Färbung innerhalb von V_u verschlechtern, können trotzdem akzeptiert werden, wenn gleichzeitig in der Komponente von v_1, v_2 die Fitness ein ausreichend großer Fitnessgewinn auftritt.

Das einfachste Beispiel für eine solche Konstellation ist ein Graph $G = (V, E)$, der nur aus den Knoten u_1, u_2 und v_1, v_2 und den Kanten $E = \{\{u_1, u_2\}, \{v_1, v_2\}\}$ besteht. Wenn in einem Suchpunkt x gilt: $x_{u_1} = x_{u_2}$, kann der (1+1) EA diese Färbung in einem später erreichten Suchpunkt x' nur dann noch ändern, wenn $x_{v_1} \neq x_{v_2}$ ist. Die Färbungen der vier Knoten sind also in x' nicht stochastisch unabhängig.

Unabhängige Zusammenhangskomponenten

Als Basis für kompliziertere Überlegungen führen wir zunächst einen formalen Beweis für die Aussage, dass die Färbungen von Pivot-Knoten in verschiedenen Zusammenhangskomponenten unabhängig sind. Dabei betrachten wir die *Trajektorie*

von Suchpunkten, sprich die Folge der durchlaufenen Suchpunkte des betrachteten Algorithmus über die Zeit.

Lemma 3. *Gegeben ein Graph $G = (V, E)$, der aus den Zusammenhangskomponenten $V_1, \ldots, V_k$ besteht und ein Algorithmus $\mathcal{A} \in \{RLS, (1+1)\ EA\}$. Sei x der aktuelle Suchpunkt von $\mathcal{A}$, $v_1, \ldots, v_k$ mit $v_i \in V_i$ beliebige Pivot-Knoten aus den k Zusammenhangskomponenten und $c_1, \ldots, c_k$ mit $c_i \in \{0, 1\}$ beliebige Farben der Pivot-Knoten. Dann sind die Ereignisse $x_{v_i} = c_i$ stochastisch unabhängig:*

$$\forall I \subseteq \{1, \ldots, k\}\colon \text{Prob}\left(\bigcap_{i \in I} \{x_{v_i} = c_i\}\right) = \prod_{i \in I} \text{Prob}(x_{v_i} = c_i)$$

Beweis. Wir zerlegen die Suchpunkte in Teilfärbungen: Für x bezeichnen wir die Teilfärbungen mit $x = (x_{(1)}, \ldots, x_{(k)})$, wobei $x_{(i)} \in \{0, 1\}^{|V_i|}$ eine Färbung der Knoten in der Zusammenhangskomponente V_i ist.

Mit dieser Notation definieren wir zwei Folgen von Suchpunkten $a^0, \ldots, a^t$ und $b^0, \ldots, b^t$ mit Teilfärbungen $a^j = (a^j_{(1)}, \ldots, a^j_{(k)})$ und $b_j = (b^j_{(1)}, \ldots, b^j_{(k)})$. Die Folge $a^0, \ldots, a^t$ sei beliebig, die Folge $b^0, \ldots, b^t$ definieren wir dann wie folgt: b^j entstehe aus a^j dadurch, dass die Teilfärbungen einer beliebigen Menge von Zusammenhangskomponenten invertiert wird. Sei $I \subseteq \{1, \ldots, k\}$ eine beliebige Indexmenge, dann gilt für alle $0 \leq j \leq t$ und $1 \leq i \leq k$:

$$b^j_{(i)} := \begin{cases} \overline{a^j_{(i)}}, & \text{falls } i \in I \\ a^j_{(i)}, & \text{falls } i \notin I \end{cases}$$

Wir zeigen nun per Induktion über t, dass beide Folgen von Suchpunkten mit gleicher Wahrscheinlichkeit durchlaufen werden. Sei x^t der Suchpunkt von $\mathcal{A}$ in Generation t, dann zeigen wir

$$\text{Prob}\left(\bigcap_{j=0}^{t} \{x^j = a^j\}\right) = \text{Prob}\left(\bigcap_{j=0}^{t} \{x^j = b^j\}\right).$$

Induktionsanfang: Der Suchpunkt x^0 entspricht einem bei der Initialisierung zufällig gewählten Suchpunkt. Daher gilt

$$\text{Prob}(x^0 = a^0) = \text{Prob}(x^0 = b^0).$$

Induktionsschluss: Angenommen, die Behauptung gilt für $t-1$. Die Behauptung für t lässt sich mit folgender Formel auf die Behauptung für $t-1$ zurück führen:

$$\begin{aligned} & \text{Prob}\left(\bigcap_{j=0}^{t} \{x^j = a^j\}\right) \\ = \ & \text{Prob}\left(\{x^t = a^t\} \,\middle|\, \bigcap_{j=0}^{t-1} \{x^j = a^j\}\right) \cdot \text{Prob}\left(\bigcap_{j=0}^{t-1} \{x^j = a^j\}\right) \quad (*) \end{aligned}$$

Auf den rechten Faktor können wir nachher die Induktionsvoraussetzung anwenden. Zuerst aber kümmern wir uns um die bedingte Wahrscheinlichkeit, den Übergang zu a^t durchzuführen. Nach Definition von Algorithmus $\mathcal{A}$ hängt ein Übergang nur vom letzten aktuellen Suchpunkt ab, hier also a^{t-1}.

Nach Konstruktion der beiden Folgen müssen für die Übergänge von a^{t-1} zu a^t und von b^{t-1} zu b^t die gleichen Bits geflippt werden. Die Wahrscheinlichkeiten, dass eine Mutation aus a^{t-1} den Mutanten a^t oder aus b^{t-1} den Mutanten b^t erzeugt, sind also gleich. Zweitens gilt aufgrund der Bit-Flip-Symmetrie der Zusammenhangskomponenten $f(a^t) = f(b^t)$. Damit wird a^t als Nachkomme von a^{t-1} genau dann akzeptiert, wenn b^t als Nachkomme von b^{t-1} akzeptiert wird. Es gilt also

$$\text{Prob}\left(\{x^t = a^t\} \;\middle|\; \bigcap_{j=0}^{t-1}\{x^j = a^j\}\right) = \text{Prob}\left(\{x^t = b^t\} \;\middle|\; \bigcap_{j=0}^{t-1}\{x^j = b^j\}\right) \qquad (**)$$

Wenn wir dies in die Formel $(*)$ einsetzen und die Induktionsvoraussetzung für $t-1$ anwenden, erhalten wir

$$\begin{aligned}
& \text{Prob}\left(\bigcap_{j=0}^{t}\{x^j = a^j\}\right) \\
\overset{(*)}{=}\; & \text{Prob}\left(\{x^t = a^t\} \;\middle|\; \bigcap_{j=0}^{t-1}\{x^j = a^j\}\right) \cdot \text{Prob}\left(\bigcap_{j=0}^{t-1}\{x^j = a^j\}\right) \\
\overset{(**)}{=}\; & \text{Prob}\left(\{x^t = b^t\} \;\middle|\; \bigcap_{j=0}^{t-1}\{x^j = b^j\}\right) \cdot \text{Prob}\left(\bigcap_{j=0}^{t-1}\{x^j = a^j\}\right) \\
\overset{\text{I.V.}}{=}\; & \text{Prob}\left(\{x^t = b^t\} \;\middle|\; \bigcap_{j=0}^{t-1}\{x^j = b^j\}\right) \cdot \text{Prob}\left(\bigcap_{j=0}^{t-1}\{x^j = b^j\}\right) \\
=\; & \text{Prob}\left(\bigcap_{j=0}^{t}\{x^j = b^j\}\right).
\end{aligned}$$

Da dies für alle Folgen von Suchpunkten gilt, die zu a^t und b^t führen, folgt daraus, dass a^t und b^t mit der gleichen Wahrscheinlichkeit erreicht werden:

$$\text{Prob}(x^t = a^t) = \text{Prob}(x^t = b^t).$$

Da wir die Indexmenge I, die die in b gegenüber a geflippten Teilfärbungen beschreibt, beliebig wählen können, haben wir sogar gezeigt, dass alle Suchpunkte mit gleicher Wahrscheinlichkeit erreicht werden, die durch das Flippen ganzer Teilfärbungen aus a^t entstehen.

Es folgt: Da es $2^{|I|}$ verschiedene Färbungen der $|I|$ ausgewählten Pivot-Knoten gibt, hat eine spezielle Färbung dieser Pivot-Knoten eine Wahrscheinlichkeit von $2^{-|I|}$,

erreicht zu werden. Also gilt

$$\text{Prob}\left(\bigcap_{i\in I}\{x_{v_i} = c_i\}\right) \quad = \quad 2^{-|I|}$$

Zweitens gelten aus Symmetriegründen für alle v_i und alle c_i die unbedingten Wahrscheinlichkeiten

$$\text{Prob}(x_{v_i} = c_i) = \frac{1}{2}.$$

Zusammen gilt für alle Indexmengen $I \subseteq \{1, \ldots, k\}$:

$$\text{Prob}\left(\bigcap_{i\in I}\{x_{v_i} = c_i\}\right) = 2^{-|I|} = \prod_{i\in I} \text{Prob}(x_{v_i} = c_i).$$

□

Unabhängige Knotenmengen

Nun wollen wir ausgefeiltere Methoden entwickeln, um die Unabhängigkeit von Pivot-Knoten über die Zeit zu bestimmen. Zu Anfang eines Laufs, nach der zufälligen Initialisierung, sind alle Färbungen unabhängig; Abhängigkeiten entwickeln sich erst nach und nach. Diese Abhängigkeiten können wir durch eine Partition der Knotenmenge sichtbar machen, wobei jede Menge genau die Knoten enthält, zwischen denen es Abhängigkeiten geben kann; die Färbungen von Pivot-Knoten aus verschiedenen Mengen der Partition sollen unabhängig sein.

Wenn wir die Abhängigkeiten über die Zeit betrachten, erhalten wir eine Folge von Partitionen. Wir starten mit einer Partition, die nur einelementige Teilmengen enthält, da bei der zufälligen Initialisierung alle Knoten unabhängig voneinander gefärbt werden. Mit der Zeit entwickeln sich Abhängigkeiten innerhalb einer Zusammenhangskomponente, so dass die Partition sich nach und nach einer Partition in Zusammenhangskomponenten annähert.

Welche neuen Abhängigkeiten können in einem Schritt entstehen? Wenn im Mutationsschritt ein Bit x_i geflippt wird, hängt die Akzeptanz des Nachkommens u. a. davon ab, wie die zu i adjazenten Knoten gefärbt sind. Daher ergeben sich möglicherweise zusätzliche Abhängigkeiten zwischen i und den Nachbarn von i. (Abhängigkeiten entstehen also insbesondere auch in Schritten, in denen der Nachkomme verworfen wird!)

Diese Abhängigkeiten können sich weiter fortpflanzen: Falls i zu einem Knoten i' abhängig ist (also i und i' in der gleichen Menge der Partition liegen) und ein Nachbar $j \in \text{Adj}(i)$ zu einem Knoten j' abhängig ist (also j und j' in der gleichen Menge der Partition liegen), sind nach diesem Schritt i' und j' ebenfalls abhängig voneinander. (Genauer: Wir können nicht mehr garantieren, dass i' und j' stochastisch unabhängig sind, also nehmen wir pessimistischerweise an, dass sie abhängig voneinander sind.)

Eine Kante $\{i, j\}$, die zu einem flippenden Knoten i inzident ist, schlägt also eine Brücke zwischen den Mengen von i und j; alle Knoten in der einen Menge können nun zu allen Knoten der anderen Menge abhängig sein. Ein Beispiel für ein solches Szenario ist in Abbildung 2.1 skizziert. Um die Partition für die nächste Generation zu erstellen, müssen wir daher die Mengen von i und j vereinigen (sofern nicht i und j bereits in der gleichen Menge liegen). Insgesamt werden daher in der neuen Partition die Menge von i und alle Mengen von Nachbarn von i miteinander verschmolzen.

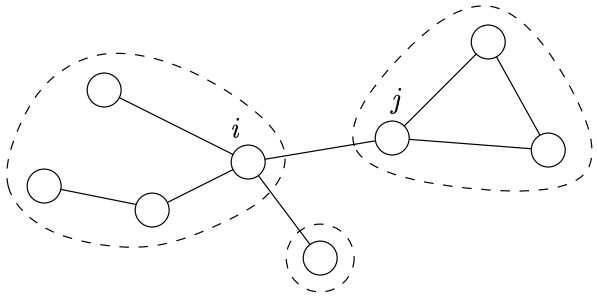


Abbildung 2.1: Ein Beispiel für eine Partition in Teilmengen (gestrichelt), die mögliche Abhängigkeiten repräsentieren. Wenn in einem Mutationsschritt Knoten i flippt, müssen alle drei Mengen zu einer Gesamtmenge vereinigt werden.

Mit diesen Ideen können wir nun die angesprochenen Partitionen formal definieren.

Definition 3. *Gegeben ein Graph $G = (V, E)$ mit $n := |V|$ Knoten und ein Algorithmus $\mathcal{A} \in \{RLS, (1+1)\ EA\}$. Wir definieren eine zufällige Folge von Partitionen $\mathcal{P}^t = \mathcal{P}^t(\mathcal{A})$ der Knotenmenge. Es sei*

$$\mathcal{P}^0 := (\{1\}, \ldots, \{n\})$$

und für $t > 0$ ergibt sich $\mathcal{P}^t$ aus $\mathcal{P}^{t-1}$ mit folgendem Algorithmus. Sei S_k^t die Menge der Partition $\mathcal{P}^t$, die den Knoten k enthält. Der Befehl UNION$(A, B, \mathcal{P})$ *vereinigt die Mengen A und B in der Partition $\mathcal{P}$: Die Mengen A und B werden aus der Partition $\mathcal{P}$ entfernt und die Vereinigung $A \cup B$ wird $\mathcal{P}$ hinzugefügt.*

1. $\mathcal{P}^t := \mathcal{P}^{t-1}$.
2. *Für alle flippenden Bits x_i:*

   Für alle $j \in Adj(i)$:

   UNION$(S_i^t, S_j^t, \mathcal{P}^t)$.

Wir behaupten nun, dass die Färbungen von Pivot-Knoten zum Zeitpunkt t unabhängig sind, wenn die Pivot-Knoten in paarweise verschiedenen Mengen von $\mathcal{P}^t$ liegen. Aus den gleichen Gründen wie zuvor bei der Betrachtung von Zusammenhangskomponenten darf aus jeder Menge jeweils nur ein Pivot-Knoten betrachtet werden.

Lemma 4. *Gegeben ein Graph $G = (V, E)$, ein Algorithmus $\mathcal{A} \in \{RLS, (1+1)\ EA\}$ auf $f_G := \text{Ising}_G$ und der Suchpunkt $x = x^t$ von $\mathcal{A}$ zum Zeitpunkt t. Seien $v_1, \ldots, v_k$ mit $v_i \in S_i^t$ beliebige Pivot-Knoten aus paarweise verschiedenen Mengen S_i^t der Partition $\mathcal{P}^t = \mathcal{P}^t(\mathcal{A})$ und $c_1, \ldots, c_k$ mit $c_i \in \{0, 1\}$ beliebige Farben der Pivot-Knoten. Dann sind die Ereignisse $x_{v_i} = c_i$ stochastisch unabhängig:*

$$\forall I \subseteq \{1, \ldots, k\}\colon \text{Prob}\left(\bigcap_{i \in I} \{x_{v_i} = c_i\}\right) = \prod_{i \in I} \text{Prob}(x_{v_i} = c_i)$$

Beweis. Sei $E_{\mathcal{P}}^t$ die Menge der Kanten zwischen den Mengen der Partition $\mathcal{P}^t$:

$$E_{\mathcal{P}}^t := \{\{u, v\} \in E \mid \exists i, j, i \neq j\colon u \in S_i^t, v \in S_j^t\}$$

Sei $G^* = (V, E \backslash E_{\mathcal{P}}^t)$ der Graph ohne Kanten zwischen verschiedenen Mengen aus $\mathcal{P}^t$. Damit besteht G^* aus Zusammenhangskomponenten, die genau die Mengen aus $\mathcal{P}^t$ umfassen; jede Zusammenhangskomponente enthält genau einen Pivot-Knoten v_i.

Wir spalten nun die Fitness auf in die Beiträge der Kanten in G^* und die Beiträge der Kanten in $E_{\mathcal{P}}^t$:

$$f_G(x') = f_{G^*}(x') + f_G(x', E_{\mathcal{P}}^t).$$

Das zentrale Argument ist nun: Nach Konstruktion der Partitionen $\mathcal{P}^0, \ldots, \mathcal{P}^t$ sind die Kanten in $E_{\mathcal{P}}^t$ in allen Fitnessauswertungen der Generationen $0, \ldots, t$ mit den gleichen Farben gefärbt. Wenn nämlich mindestens ein Endpunkt einer Kante $e \in E_{\mathcal{P}}^t$ in einem Mutationsschritt geflippt wäre, wären beide Endpunkte von e in die gleiche Menge aufgenommen worden, was im Widerspruch zu $e \in E_{\mathcal{P}}^t$ steht.

Daher können wir die Fitness für alle bisherigen Fitnessauswertungen auch schreiben als

$$f_G(x) = f_{G^*}(x) + c$$

wobei c eine Konstante ist. Algorithmus $\mathcal{A}$ arbeitet nur mit den bisher durchgeführten Fitnessauswertungen und verhält sich auf der Funktion $f_{G^*} + c$ wie auf der Funktion f_{G^*}. Daher können wir anstelle von G den Graphen G^* betrachten und aus Lemma 3 folgt die Behauptung. □

Kern des Beweises ist also, dass die Färbung der Kanten zwischen den Mengen der Partitionen in allen bisherigen Fitnessauswertungen gleich ist. Der Algorithmus bemerkt quasi nicht, dass diese Kanten überhaupt vorhanden sind und verhält sich wie auf einem Graphen ohne diese Kanten.

Wenn der betrachtete Graph nicht zusammenhängend sind, bilden die Zusammenhangskomponenten zu jedem Zeitpunkt t verschiedene Teilmengen in $\mathcal{P}^t$. Die Aussage von Lemma 4 ist also eine stärkere Aussage als die von Lemma 3 über verschiedene Zusammenhangskomponenten.

Lemma 4 zeigt, dass in den ersten Schritten eines Laufs viele Färbungen unabhängig sind. Daher lässt sich das Lemma gut einsetzen, um eine kurze Anfangsphase eines Laufs zu analysieren und zu garantieren, dass ausgewählte Pivot-Knoten bestimmter

Teilgraphen unabhängig voneinander gefärbt sind, solange bestimmte Knoten, die diese Teilgraphen miteinander verbinden, nicht flippen. Wir werden das Lemma in Abschnitt 3.3 zu genau diesem Zweck einsetzen, um Graphen mit zwei Cliquen, die durch eine Kante verbunden sind, zu analysieren.

Andererseits ist Lemma 4 für eine Anwendung auf zusammenhängende Graphen über einen längeren Zeitraum ungeeignet. Mit Hilfe des Coupon-Collector-Theorems [18] kann man zeigen, dass sowohl die randomisierte lokale Suche als auch der (1+1) EA mit sehr hoher Wahrscheinlichkeit in $O(n \log n)$ Schritten jedes Bit mindestens einmal flippen. (Für die randomisierte lokale Suche folgt dies direkt aus dem Coupon-Collector-Theorem. Für den (1+1) EA folgt eine nur um einen Faktor e höhere Schranke, wenn wir uns mit der Betrachtung von 1-Bit-Mutationen begnügen.)

Wenn alle Bits mindestens einmal geflippt sind, besteht die Partition $\mathcal{P}$ für zusammenhängende Graphen nur noch aus der Gesamtmenge V, so dass keinerlei Unabhängigkeiten mehr garantiert werden können.

Kapitel 3

Das Ising-Modell auf Cliquengraphen

Die erste von zwei Graphklassen, mit der wir uns in dieser Arbeit beschäftigen wollen, sind so genannte *Cliquengraphen*. In einem Graphen $G = (V, E)$ ist eine *Clique* eine Knotenmenge $C \subseteq V$, in der jeder Knoten mit jedem anderen Knoten verbunden ist:

$$\forall i, j \in C, i \neq j \colon \{i, j\} \in E.$$

Ein Cliquengraph mit k Cliquen ist dann wie folgt definiert.

Definition 4. *Ein Graph $G = (V, E)$ mit $n := |V|$ Knoten ist ein Cliquengraph mit k Cliquen, $1 \leq k \leq n$, wenn sich die Knotenmenge V in k nicht leere Teilmengen $C_1, \ldots, C_k$ partitionieren lässt, so dass $C_1, \ldots, C_k$ jeweils Cliquen sind.*

Die Cliquen können unabhängige Zusammenhangskomponenten bilden oder durch Kanten verbunden sein, die wir als *Brückenkanten* bezeichnen.

Wenn die Cliquen unabhängige Zusammenhangskomponenten bilden, ist die Ising-Funktion auf solchen Graphen leicht zu optimieren. Eine Clique enthält viele Kanten und gibt daher auch starke Hinweise darauf, diese Kanten einheitlich zu färben, indem die Knoten der Clique die gleiche Farbe annehmen. Wenn der Graph nur aus einer Clique besteht, sprich wenn ein vollständiger Graph mit n Knoten vorliegt, kann man mit Fitnessschichten und der unteren Schranke aus Lemma 6 leicht zeigen, dass die erwartete Optimierungszeit der randomisierten lokalen Suche und des (1+1) EA in der Größenordnung $\Theta(n \log n)$ liegt.

Die Ising-Funktion auf mehreren unabhängigen Cliquen ist separabel in den Knotenmengen $C_1, \ldots, C_k$, die die Cliquen bilden, d. h. die Fitness entspricht der Summe von Fitnessbeiträgen aller Cliquen. Es ist daher möglich, die Cliquen unabhängig voneinander zu optimieren und so das Gesamtproblem zu lösen.

Leider ist eine unabhängige Optimierung bei vielen evolutionären Algorithmen wie z. B. dem (1+1) EA nicht ohne Weiteres möglich: Es kann Schritte geben, bei denen

die „Fitness“ einer Clique (genauer: der Beitrag einer Clique zur Fitness) erhöht wird, dafür aber die Fitness einer anderen Clique gesenkt wird. Dadurch wird eine Analyse trotz der Separabilität der Funktion erschwert.

In Abschnitt 3.2.1 werden wir zeigen, dass die üblichen einfachen Analysemethoden nicht zum gewünschten Erfolg führen, nämlich eine scharfe obere Schranke $O(n \log n)$ für die erwartete Optimierungszeit des (1+1) EA zu zeigen. In Abschnitt 3.2.2 werden wir einen alternativen Ansatz kennen lernen und eine Methode erarbeiten, die uns bei der Analyse des (1+1) EA auf unabhängigen Cliquen helfen wird.

In Abschnitt 3.2.3 werden wir dann die erarbeitete Methode verwenden, um für den (1+1) EA auf k gleich großen unabhängigen Cliquen eine Schranke $O(k \cdot n \log n)$ zu zeigen, die für eine konstante Zahl von Cliquen scharf ist. Auf diesem Beweis aufbauend zeigen wir die angestrebte obere Schranke $O(n \log n)$ für eine beliebige Zahl gleich großer unabhängiger Cliquen, indem wir in Abschnitt 3.2.4 einen modifizierten Algorithmus analysieren und dann in Abschnitt 3.2.5 die Resultate auf den (1+1) EA übertragen.

Im Fall verbundener Cliquen stehen evolutionäre Algorithmen vor dem Problem, dass die Cliquen mit verschiedenen Farben gefärbt werden können; ein Problem, das von Goldberg, van Hoyweghen und Naudts als *Synchronisationsproblem* bezeichnet wird [10]. Der Algorithmus hängt dann im Allgemeinen in einem lokalen Optimum fest, das nur sehr schwer zu verlassen ist.

Der worst case ist hier ein Graph mit zwei gleich großen Cliquen, die durch eine Brückenkante miteinander verbunden sind. Unter allen Möglichkeiten, zwei Cliquen miteinander zu verbinden, gibt eine einzelne Brückenkante die wenigsten Hinweise darauf, die Cliquen mit der gleichen Farbe zu färben. Während bei unabhängigen Cliquen beide Cliquen mit Wahrscheinlichkeit 1/2 mit verschiedenen Farben gefärbt werden, sollte dieser Fall bei Cliquen mit einer Brückenkante daher mit einer Wahrscheinlichkeit von $1/2 - o(1)$ eintreten; eine Wahrscheinlichkeit, die also nur unwesentlich kleiner ist als 1/2. Diese Aussage werden wir in Abschnitt 3.3 beweisen und dadurch zeigen, dass der genannte Graph ein asymptotisches Worst-Case-Beispiel für die erwartete Optimierungszeit des (1+1) EA auf der Ising-Funktion ist.

3.1 Einführung und Grundlagen

In diesem Abschnitt wollen wir nun einige auf Cliquengraphen bezogene Grundlagen erarbeiten, die wir später verwenden werden. Manche dieser Grundlagen sind auf Graphen mit unabhängigen Cliquen bezogen; sie gelten aber meist in ähnlicher Form auch für Graphen mit wenigen Brückenkanten, da die Fitness durch die wenigen Brückenkanten gegenüber Graphen ohne Brückenkanten nur leicht „verrauscht“ wird.

3.1.1 Die Betrachtung von Mehrheiten

Definitionen

Bei der Färbung eines Cliquengraphens spielt eine große Rolle, wie groß die *Mehrheit* der vorherrschenden Farbe in einer Clique ist. Damit meinen wir die Zahl der Knoten, die mit der Farbe gefärbt sind, die in der Clique am häufigsten vorkommt.

Definition 5. *Gegeben ein Cliquengraph $G = (V, E)$ mit $n := |V|$ Knoten und Cliquen $C_1, \ldots, C_k$. Sei C eine dieser Cliquen, $x \in \{0,1\}^n$ ein Suchpunkt und $||x_C||_i$ die Zahl der Knoten in C, die durch x mit Farbe i gefärbt werden.*

Die Mehrheit in C bezüglich der Färbung x ist definiert als

$$\mathrm{maj}(x, C) := \max\{||x_C||_0, ||x_C||_1\}.$$

Die Mehrheit wird also immer auf eine Färbung bezogen. Falls die Färbung aus dem Kontext ersichtlich ist, erlauben wir es uns, im laufenden Text die Färbung auszulassen und nur von Mehrheiten zu sprechen.

Die Farbe i, für die das Maximum aus $||x_C||_0$ und $||x_C||_1$ angenommen wird, bezeichnen wir als *Mehrheitsfarbe* in C, die andere Farbe als *Minderheitsfarbe* in C. In dem Sonderfall, dass beide Farben in C gleich oft vorkommen, bezeichnen wir eine beliebige Farbe $c \in \{0,1\}$ als Mehrheitsfarbe und die andere Farbe $1 - c$ als Minderheitsfarbe.

Ein Bit, das in einer Färbung x die Mehrheitsfarbe seiner Clique annimmt, bezeichnen wir als *Mehrheitsbit* in x; ein Bit, dass in x die Minderheitsfarbe seiner Clique annimmt, bezeichnen wir als *Minderheitsbit* in x. Auch hier erlauben wir es uns, die Färbung x gegebenenfalls auszulassen, wenn der Bezug eindeutig ist.

Das Maximum zweier Werte ist stets mindestens so groß wie der Durchschnitt. Die Mehrheit einer Clique ist daher minimal, wenn genau $\lceil |C|/2 \rceil$ Knoten mit der Mehrheitsfarbe gefärbt sind; sie ist maximal, wenn die Clique einheitlich gefärbt ist, d. h. wenn alle Knoten der Clique mit der gleichen Farbe gefärbt sind. Daher gilt für die Mehrheit in C immer

$$\left\lceil \frac{|C|}{2} \right\rceil \;\leq\; \mathrm{maj}(x, C) \;\leq\; |C|.$$

Mehrheiten und Fitness

Bei unabhängigen Cliquen $C_1, \ldots, C_k$ lässt sich die Fitness aus alle Mehrheiten $\mathrm{maj}(x, \cdot)$ vollständig errechnen: In einer Clique C bilden die $\mathrm{maj}(x, C)$ Knoten einen einheitlich gefärbten Subgraphen mit $\binom{\mathrm{maj}(x,C)}{2}$ Kanten; das Gleiche gilt für die $|C| - \mathrm{maj}(x, C)$ mit der Minderheitsfarbe gefärbten Knoten. Alle anderen Kanten in C verlaufen zwischen diesen beiden Subgraphen und sind somit zweifarbig. Daher gilt für $f := \mathrm{Ising}_G$:

$$f(x) = \sum_{i=1}^{k} \left(\binom{\mathrm{maj}(x, C_i)}{2} + \binom{|C_i| - \mathrm{maj}(x, C_i)}{2} \right)$$

Wenn Brückenkanten ins Spiel kommen, muss lediglich der Fitnessbeitrag aller Brückenkanten zur rechten Seite addiert werden, um die Fitness zu errechnen.

Von der Mehrheit in C hängt es auch ab, wie hoch der Fitnessgewinn bei einer Mutation ist. Dies ist am leichtesten einzusehen, wenn wir Cliquengraphen ohne Brückenkanten betrachten und uns auf 1-Bit-Mutationen konzentrieren. In Abschnitt 2.2 haben wir bereits gesehen, dass sich die Fitness bei einer 1-Bit-Mutation eines Knotens i um $\ell(x, x_i) = \deg_x^-(i) - \deg_x^+(i)$ ändert. Wenn x_i ein Minderheitsbit ist, gibt es in x genau $\deg_x^-(i) = \mathrm{maj}(x, C)$ zu i inzidente zweifarbige Kanten, die bei einem Bitflip von x_i einfarbig werden und positiv in die Fitnessveränderung eingehen. Weiter gibt es in x genau $\deg_x^+(i) = |C| - \mathrm{maj}(x, C) - 1$ zu i inzidente einfarbige Kanten (wir zählen alle anderen Minderheitsbits in C außer i selbst). Insgesamt ist die lokale Fitnessveränderung, wenn ein Mehrheitsbit x_i flippt, also

$$\begin{aligned} \ell(x, x_i) &= \mathrm{maj}(x, C) - (|C| - \mathrm{maj}(x, C) - 1) \\ &= 2\,\mathrm{maj}(x, C) - |C| + 1. \end{aligned}$$

Der Fitnessgewinn ist also bei fester Cliquengröße $|C|$ proportional zur Mehrheit in C. Die Spanne der möglichen Fitnessverbesserungen durch 1-Bit-Mutationen in C reicht von $\ell(x, x_i) = 1$, falls $\mathrm{maj}(x, C) = |C|/2$ und $|C|$ gerade ist, bis hin zu $\ell(x, x_i) = |C| - 1$, falls $\mathrm{maj}(x, C) = |C| - 1$ ist, sprich: das letzte Minderheitsbit in C flippt.

Wenn x_i hingegen ein Mehrheitsbit ist, gibt es $\deg_x^-(i) = |C| - \mathrm{maj}(x, C)$ zu i inzidente zweifarbige Kanten, nämlich alle Kanten zu Minderheitsbits. Weiter gibt es in x genau $\deg_x^+(i) = \mathrm{maj}(x, C) - 1$ zu i inzidente einfarbige Kanten, nämlich die Kanten zu allen anderen Mehrheitsbits außer i in C. Die Fitnessveränderung beträgt dann

$$\begin{aligned} \ell(x, x_i) &= |C| - \mathrm{maj}(x, C) - (\mathrm{maj}(x, C) - 1) \\ &= |C| - 2\,\mathrm{maj}(x, C) + 1. \end{aligned}$$

Anschaulich haben Bits in Cliquen mit unterschiedlichen Mehrheiten unterschiedliche „Gewichte“, sprich unterschiedlich lokale Fitnessveränderungen und unterschiedlich starke Einflüsse auf die Fitness. Je größer die Mehrheit $\mathrm{maj}(x, C)$, um so größer das Gewicht der Bits von Knoten in C.

Kippende Mehrheiten

Die Mehrheit in einer Clique C bezüglich einer Färbung x kann durch Bitflips von Minderheitsbits und Mehrheitsbits verändert werden. Dabei kann es jedoch auch vorkommen, dass in einem Schritt die Mehrheitsfarbe wechselt und die aktuelle Minderheitsfarbe in x im neuen Suchpunkt x' die Mehrheitsfarbe darstellt. In einem solche Fall sprechen wir davon, dass die Mehrheit in x *kippt*.

Kippende Mehrheiten müssen in unseren Betrachtungen berücksichtigt werden, treten in der Praxis aber nur selten auf. Angenommen, die Farbe $c \in \{0, 1\}$ ist Mehrheitsfarbe in C bezüglich x. Damit in einem aus x erzeugten Nachkommen x' mindestens $\lceil |C|/2 \rceil$ Knoten mit Farbe $1 - c$ gefärbt sind, müssen in x mindestens

$\lceil \text{maj}(x, C) - |C|/2 \rceil$ Mehrheitsbits flippen. Wenn die Cliquengröße $|C|$ ungerade ist, ist in x' eine andere Farbe in der Mehrheit als in x; ist die Cliquengröße $|C|$ gerade, hängt die Mehrheit davon ab, welche Farbe wir bei Gleichstand als Mehrheitsfarbe bezeichnen. Eventuell sind daher mindestens $\text{maj}(x, C) - |C|/2 + 1$ flippende Mehrheitsbits nötig, um die Mehrheit zu kippen.

3.1.2 Die Summe der Mehrheiten als Potenzialfunktion φ

Wir haben gesehen, dass Mehrheiten entscheidend für die Fitness eines Suchpunkts sind. Zweitens gibt eine Mehrheit $\text{maj}(x, C)$ an, wie viele Bits von x mit einem nächstgelegenen Suchpunkt im Raum $\{0,1\}^n$ überein stimmen, bei dem die Clique C einheitlich gefärbt ist. Wenn wir die Mehrheiten aller Cliquen aufsummieren, erhalten wir ein sinnvolles Maß für den Fortschritt des Algorithmus und die Nähe zu einer einheitlichen Färbung aller Cliquen. Nach den Ideen aus Abschnitt 1.5.6 können wir daher die Summe der Mehrheiten als *Potenzialfunktion* φ verwenden.

Definition 6. *Gegeben ein beliebiger Cliquengraph $G = (V, E)$ mit $n := |V|$ Knoten und Cliquen $C_1, \ldots, C_k$. Die Potenzialfunktion $\varphi\colon \{0,1\}^n \to \mathbb{N}$ ist definiert als*

$$\varphi(x) := \sum_{i=1}^{k} \text{maj}(x, C_i).$$

Die Potenzialfunktion ist eng mit der Fitness verbunden. Wenn alle Cliquen eine gerade Größe haben, ist der minimale Wert für φ der Wert $\varphi(x) = n/2$, der genau dann erreicht wird, wenn in allen Cliquen zwischen beiden Farben „Gleichstand“ herrscht, wenn also in jeder Clique genau so viele Knoten 1-gefärbt sind wie 0-gefärbt. Auf unabhängigen Cliquen hat ein solcher Suchpunkt eine minimale Fitness unter allen Suchpunkten des Suchraums $\{0,1\}^n$. Ähnliches gilt auf unabhängigen Cliquen für Suchpunkte mit maximalem φ-Wert. Das Maximum ist $\varphi(x) = n$ und es wird für alle Suchpunkte angenommen, bei denen alle Cliquen einheitlich gefärbt sind. Bei unabhängigen Cliquen entspricht dies genau allen globalen Optima.

Bei unabhängigen Cliquen stimmen also die Minima und Maxima der Potenzialfunktion mit den Minima und Maxima der Fitnessfunktion überein. Bei Cliquengraphen mit wenigen Brückenkanten gelten ähnliche Aussagen, obwohl die Fitness durch die Brückenkanten gegenüber Cliquengraphen ohne Brückenkanten leicht „verrauscht“ wird. Ein Suchpunkt mit $\varphi(x) = n$ charakterisiert auf Cliquengraphen mit verbundenen Cliquen im Allgemeinen ein globales oder lokales Optimum.

Die Potenzialfunktion hat vielfältige Anwendungsgebiete bei der Analyse von Algorithmen. Der Wert $\varphi(x)$ zählt genau die Zahl der Mehrheitsbits in x; der Term $n - \varphi(x)$ gibt daher die Zahl der Minderheitsbits in x an. Dies entspricht dem Hammingabstand zu der nächsten Färbung, bei der alle Cliquen einheitlich mit den jeweiligen Mehrheitsfarben gefärbt sind. Bei unabhängigen Cliquen gibt es immer $n - \varphi(x)$ Fitness erhöhende 1-Bit-Mutationen, daher findet die Potenzialfunktion unter anderem bei einfachen Laufzeitabschätzungen Verwendung.

In manchen Situationen ist die Potenzialfunktion deutlich handlicher als die Fitness, da das Potenzial nur die Zahl der Mehrheitsbits erfasst und von der konkreten Verteilung der Mehrheitsbits abstrahiert. Dadurch lassen sich lokale Veränderungen flippender Bits wesentlich leichter behandeln. Ein flippendes Bit verändert die Fitness lokal um einen Wert zwischen $-(|C|-1)$ und $|C|-1$, abhängig von der Mehrheit der Clique C, in der das flippende Bit liegt. Das Potenzial wird lokal nur um -1 oder $+1$ verändert; je nachdem, ob ein Mehrheitsbit oder ein Minderheitsbit geflippt wird.

Der Wertebereich der Fitness nimmt ein Intervall der Länge $\Theta(\sum_{i=1}^{k} |C_i|^2)$ ein, was bei k Cliquen mit gleicher Größe n/k einer Menge mit $\Theta(n^2/k)$ Werten entspricht (nicht alle Werte sind möglich, die Zahl der Werte beträgt aber trotz fehlender Surjektivität $\Theta(n^2/k)$). Der Wertebereich der Potenzialfunktion umfasst jedoch nur höchstens $n/2+1$ Werte. Aufgrund der kleineren Skala eignet sich φ auch gut als Fortschrittsmaß für den Algorithmus. Anstatt uns zu fragen, wie lange es dauert, bis die Fitness erhöht wird, können wir uns fragen, wie lange es dauert, bis das Potenzial erhöht wird. Entsprechende Ansätze werden wir in Abschnitt 3.2.2 entwickeln.

Das Potenzial hat allerdings den Nachteil gegenüber der Fitness, dass es nicht monoton über die Zeit ist; wenn eine Mutation eine große Mehrheit erhöht und mehrere kleine Mehrheiten senkt, kann ein solcher Schritt trotzdem akzeptiert werden, da große Mehrheiten sich stärker auf die Fitness auswirken als kleine Mehrheiten. In Abschnitt 3.2.2 werden wir sehen, wie man mit diesem Problem umgehen kann.

3.1.3 Relevante und erfolgreiche Schritte

Bei unabhängigen Cliquen sind alle Knoten einer Clique isomorph. Es kann daher Schritte geben, bei denen beispielsweise genau ein 1-gefärbter Knoten und ein 0-gefärbter Knoten der gleichen Clique flippen. In diesem Fall ändert sich an der aktuellen Färbung nichts Wesentliches; es werden lediglich Farben innerhalb einer Clique vertauscht. Die Fitness bleibt konstant, so dass ein solcher Schritt keinerlei Bewegung in den Optimierungsprozess bringt.

Daher macht es Sinn, sich auf die Betrachtung von Schritten zu konzentrieren, in denen Mehrheiten tatsächlich verändert werden. Solche Schritte bezeichnen wir als *relevante Schritte*.

Definition 7. *Gegeben ein beliebiger Cliquengraph $G = (V, E)$ mit $n := |V|$ Knoten und ein Suchpunkt $x \in \{0,1\}^n$. Ein Schritt, der aus x einen Nachkommen x' erzeugt, heißt* relevant, *wenn x' als neuer Suchpunkt akzeptiert wird und wenn sich die Mehrheit einer Clique C^* ändert:* $\mathrm{maj}(x', C^*) \neq \mathrm{maj}(x, C^*)$.

Im Fall unabhängiger Cliquen lässt sich diese Definition noch vereinfachen. Falls in einem relevanten Schritt die Mehrheiten in allen Cliquen nur sinken oder gleich bleiben, verschlechtert sich die Fitness und der Nachkomme wird nicht akzeptiert, Widerspruch zur Definition eines relevanten Schritts. Daher ist eine auf unabhängigen Cliquen eine notwendige Bedingung für einen relevanten Schritt, dass in einer

Clique die Mehrheit steigt. Einen solchen Schritt bezeichnen wir als *erfolgreichen Schritt*.

Definition 8. *Gegeben ein beliebiger Cliquengraph $G = (V, E)$ mit $n := |V|$ Knoten und ein Suchpunkt $x \in \{0,1\}^n$. Ein Schritt, der aus x einen Nachkommen x' erzeugt, heißt* erfolgreich, *wenn x' als neuer Suchpunkt akzeptiert wird und wenn sich die Mehrheit einer Clique C^* erhöht:* $\mathrm{maj}(x', C^*) > \mathrm{maj}(x, C^*)$.

Jeder erfolgreiche Schritt ist auch ein relevanter Schritt. In umgekehrter Richtung kann ein relevanter Schritt nur dann nicht erfolgreich sein, wenn Brückenkanten im Spiel sind. Mit Brückenkanten sind die Knoten einer Clique im Allgemeinen nicht mehr isomorph; wenn Farben innerhalb einer Clique vertauscht werden, kann es sein, dass dadurch die Färbung von Brückenkanten verbessert wird. Ist der daraus resultierende Fitnessgewinn groß genug, kann es sogar sein, dass in einigen Cliquen Mehrheiten sinken, keine Mehrheit erhöht wird und der Schritt trotzdem akzeptiert wird.

Trotzdem macht es auch auf Cliquen mit Brückenkanten Sinn, sich bei einigen Fragestellungen auf die Effekte relevanter und erfolgreicher Schritte zu konzentrieren, da diese Schritte normalerweise einen wesentlich höheren Einfluss auf das Verhalten der betrachteten Algorithmen haben als Schritte, die lediglich Farben innerhalb von Cliquen vertauschen.

Die Wahrscheinlichkeiten relevanter und erfolgreicher Schritte werden uns in diesem Kapitel häufiger über den Weg laufen. Daher wollen wir gleich zu Anfang obere und untere Schranken für die Wahrscheinlichkeit erfolgreicher Schritte herleiten. Die untere Schranke für die Wahrscheinlichkeit erfolgreicher Schritte ist gleichzeitig eine untere Schranke für die Wahrscheinlichkeit relevanter Schritte, da jeder erfolgreiche Schritt auch ein relevanter Schritt ist.

Die folgenden Schranken lassen sich unabhängig von den eventuell vorhandenen Brückenkanten zeigen; daher gilt der Beweis für alle Cliquengraphen mit k gleich großen Cliquen. Der Beweis hat zudem den interessanten Nebeneffekt, dass wir aus den Rechnungen Aussagen darüber entnehmen können, mit welcher Wahrscheinlichkeit die Mehrheit in einer Clique erhöht wird. Auf diese Aussagen werden wir später in den Abschnitten 3.2 und 3.3 zurück kommen.

Lemma 5. *Gegeben ein beliebiger Cliquengraph $G = (V, E)$ mit $n := |V|$ Knoten und k Cliquen $C_1, \ldots, C_k$ der Größe n/k. Sei $x \in \{0,1\}^n$ der aktuelle Suchpunkt des (1+1) EA auf $f := \mathrm{Ising}_G$ und A das Ereignis, dass ausgehend von x ein erfolgreicher Schritt stattfindet, d. h. dass sich eine Mehrheit erhöht. Dann gilt*

$$\frac{n - \varphi(x)}{en} \;\leq\; \mathrm{Prob}(A) \;\leq\; 2 \cdot \frac{n - \varphi(x)}{n}.$$

Beweis. Die untere Schranke ist leicht zu zeigen, da eine 1-Bit-Mutation, die eines von $n - \varphi(x)$ Minderheitsbits trifft, für einen erfolgreichen Schritt sorgt.

Die obere Schranke ist aufwändiger. Sei A_i für $1 \leq i \leq k$ das Ereignis, dass sich in C_i die Mehrheit erhöht. Dann gilt

$$A \subseteq \bigcup_{i=1}^{k} A_i \quad \text{und} \quad \text{Prob}(A) \leq \sum_{i=1}^{k} \text{Prob}(A_i).$$

Wir können uns also auf einzelne Cliquen konzentrieren, indem wir $\text{Prob}(A_i)$ nach oben abschätzen. Wir müssen nun zwei Fälle unterscheiden, wie sich in einer Clique C die Mehrheit erhöhen kann:

Fall 1: Die aktuelle Mehrheit wird vergrößert. (Falls die Clique mehrheitlich mit 1 gefärbt ist, flippen mehr Nullen als Einsen.)

Fall 2: Die aktuelle Mehrheit kippt und die neue Mehrheit ist größer als die alte Mehrheit. (Die Zahl der Nullen im Nachkommen x' und der Clique C ist größer als die Zahl der Einsen in x und der Clique C.)

Sei $A_i^1 \subseteq A$ das Ereignis, dass sich in Clique C_i die Mehrheit gemäß Fall 1 erhöht und $A_i^2 \subseteq A$ das Ereignis, dass sich in Clique C_i die Mehrheit gemäß Fall 2 erhöht. Wir zeigen, dass die Wahrscheinlichkeit, dass sich in einer Clique C_i die Mehrheit erhöht, in beiden Fällen zusammen beschränkt ist durch

$$\text{Prob}(A_i) = \text{Prob}(A_i^1) + \text{Prob}(A_i^2) \leq 2 \cdot \frac{n/k - \text{maj}(x, C_i)}{n}.$$

Über alle Cliquen summiert folgt dann die obere Schranke $2(n - \varphi(x))/n$.

In Fall 1 ist eine notwendige Bedingung für eine Erhöhung der Mehrheit, dass mindestens eines von $n/k - \text{maj}(x, C_i)$ Minderheitsbits flippt. Dies führt zur Ungleichung

$$\text{Prob}(A_i^1) \leq \frac{n/k - \text{maj}(x, C_i)}{n}.$$

Die gleiche Schranke zeigen wir nun für A_i^2. Damit die neue Mehrheit größer ist als die alte Mehrheit, müssen mindestens $m := 2(\text{maj}(x, C_i) - n/(2k)) + 1$ Mehrheitsbits flippen. Damit erhalten wir

$$\begin{aligned}
\text{Prob}(A_i^2) &\leq \binom{\text{maj}(x, C_i)}{m} \cdot n^{-m} \\
&= \frac{\text{maj}(x, C_i) \cdot \ldots \cdot (\text{maj}(x, C_i) - m + 1)}{m!} \cdot n^{-m} \\
&\leq \text{maj}(x, C_i) \cdot \ldots \cdot (\text{maj}(x, C_i) - m + 1) \cdot n^{-m} \\
&= \frac{\text{maj}(x, C_i)}{n} \cdot \ldots \cdot \frac{\text{maj}(x, C_i) - m + 1}{n}
\end{aligned}$$

Wir setzen nun $m := 2(\text{maj}(x, C_i) - n/(2k)) + 1$ ein und schätzen alle Faktoren bis

auf den letzten nach oben durch $\mathrm{maj}(x, C_1)/n \leq 1/k$ ab. Wir erhalten

$$\begin{aligned}\mathrm{Prob}(A_i^2) &\leq \left(\frac{1}{k}\right)^{m-1} \cdot \frac{\mathrm{maj}(x, C_i) - (2\,\mathrm{maj}(x, C_i) - n/k + 1) + 1}{n} \\ &= \left(\frac{1}{k}\right)^{m-1} \cdot \frac{n/k - \mathrm{maj}(x, C_i)}{n} \\ &\leq \frac{n/k - \mathrm{maj}(x, C_i)}{n}\end{aligned}$$

Es sei angemerkt, dass im Fall $\mathrm{maj}(x, C_i) = n/(2k)$ (falls die Cliquengröße n/k gerade ist) $\mathrm{Prob}(A_i^1) = \mathrm{Prob}(A_i^2)$ ist, da es genau so viele Mehrheitsbits wie Minderheitsbits gibt und die Mehrheit kippt, wenn mehr Mehrheitsbits flippen als Minderheitsbits. Die Schranke für $\mathrm{Prob}(A_i^2)$ ist hier also genau so scharf wie die Schranke für $\mathrm{Prob}(A_i^1)$. Erst mit wachsender Mehrheit $\mathrm{maj}(x, C_i)$ wird die Schranke für $\mathrm{Prob}(A_i^2)$ schlechter, da die Wahrscheinlichkeit von A_i^2 exponentiell fällt.

Insgesamt gilt

$$\mathrm{Prob}(A) \leq \sum_{i=1}^{k} \mathrm{Prob}(A_i) \leq \sum_{i=1}^{k} 2 \cdot \frac{n/k - \mathrm{maj}(x, C_i)}{n} = 2 \cdot \frac{n - \varphi(x)}{n}.$$

□

3.2 Die Analyse unabhängiger Cliquen

In diesem Abschnitt wollen wir Cliquengraphen mit unabhängigen Cliquen betrachten und die erwartete Optimierungszeit des (1+1) EA nach oben abschätzen. Dabei beschränken wir uns auf Graphen mit Cliquen gleicher Größe, da bei verschieden großen Cliquen nicht nur die Mehrheitsverhältnisse, sondern auch die Größenverhältnisse der Cliquen eine Rolle spielen und das Problem dadurch zusätzlich erschwert wird.

Wir haben bereits argumentiert, dass die Ising-Funktion auf einem Cliquengraphen G mit nur einer Clique (sprich: dem vollständigen Graphen) zu einer erwarteten Optimierungszeit von $\Theta(n \log n)$ führt, wenn n die Knotenzahl ist. Die Funktion Ising_G ist für den (1+1) EA relativ einfach, da es in jedem Schritt $n - \varphi(x)$ Minderheitsbits gibt und somit $n - \varphi(x)$ verschiedene 1-Bit-Mutationen die Fitness erhöhen können. Auch die Analyse des (1+1) EA auf Ising_G ist mit der Methode der Fitnessschichten leicht, da es nur $O(n)$ verschiedene Fitnesswerte gibt und $O(n)$ Fitnessschichten ausreichen.

Wenn wir die Zahl der Cliquen erhöhen, erhöht sich zum einen die Zahl möglicher Fitnesswerte, zum anderen aber auch der Grad der Separabilität der Fitnessfunktion. Unsere Intuition sagt uns, dass das Problem dadurch nicht wesentlich schwieriger werden sollte; die Zahl hilfreicher 1-Bit-Mutationen jedenfalls bleibt dabei konstant. Auch Experimente bestätigen diese Intuition; die erwartete Optimierungszeit scheint

für alle nicht trivialen Zahlen von Cliquen gleich groß zu sein. Daher sollte es möglich sein, für Cliquengraphen mit einer beliebigen Zahl von unabhängigen, gleich großen Cliquen eine Schranke $O(n \log n)$ für die erwartete Optimierungszeit zu zeigen.

3.2.1 Fitnessbasierte Ansätze

Als erste Annäherung an das Problem wollen wir versuchen, die Methode der Fitnessschichten anzuwenden.

Theorem 8. *Gegeben ein Cliquengraph $G = (V, E)$ mit $n := |V|$ Knoten und $1 \leq k \leq n/2$ unabhängigen Cliquen der Größe n/k. Die erwartete Optimierungszeit des (1+1) EA auf $f := \text{Ising}_G$ ist beschränkt durch $O(n/k \cdot n \log n)$.*

Beweis. Wir verwenden die Methode der Fitnessschichten mit der kanonischen Einteilung in Schichten $A_0 <_f \cdots <_f A_{|E|}$ mit

$$A_i := \{x \mid f(x) = i\}.$$

Wie viele Minderheitsbits gibt es in Fitnessschicht A_i? Jedes flippende Bit erhöht die Fitness um maximal $n/k-1 < n/k$, da jeder Knoten nur Grad $n/k-1$ hat. Wenn der Fitnesswert $f(x) = i$ beträgt, ist die Zahl der Minderheitsbits in x mindestens

$$\frac{|E| - i}{n/k},$$

da man mit weniger Minderheitsbits keine Fitness von $|E|$ erreichen kann. Die erwartete Zeit, bis eine solche 1-Bit-Mutation auftritt und die Fitness erhöht wird, ist höchstens

$$\frac{n/k}{|E| - i} \cdot en.$$

Die erwartete Zeit, bis wir eine Fitness von $|E|$ erreichen, ist damit höchstens

$$1 + \sum_{i=0}^{|E|-1} \frac{n/k}{|E| - i} \cdot en \;=\; 1 + n/k \cdot en \sum_{i=1}^{|E|} \frac{1}{i} \;=\; O(n/k \cdot n \log n).$$

□

Für eine konstante Zahl von Cliquen liefert Theorem 8 nur eine obere Schranke von $O(n^2 \log n)$, die also um einen Faktor n von unserem angestrebten Ziel entfernt ist.

Warum liefert dieser einfache Ansatz keine überzeugenden Resultate?

Der Erfolg von Fitnessschichten basiert auf einer geschickt gewählten Einteilung in Schichten und auf einer ausreichend großen Wahrscheinlichkeit, die aktuelle Schicht nach oben zu verlassen.

Die Wahrscheinlichkeit einer Fitnesserhöhung bei einer Fitness von $f(x) = i$ schätzen wir ab, indem wir zeigen, dass es mindestens $(|E| - i)/(n/k)$ Minderheitsbits gibt.

Diese Abschätzung scheint zunächst grob; wir wollen jedoch im Folgenden zeigen, dass die Abschätzung bis auf einen konstanten Faktor scharf ist.

Dazu verwenden wir folgendes Gedankenspiel, in dem wir den Optimierungsprozess quasi rückwärts verfolgen. Gegeben ein Suchpunkt x. Wir gehen von einem globalen Optimum x^{opt} aus, das x am nächsten liegt und betrachten eine kürzeste Folge von 1-Bit-Mutationen, die aus x^{opt} den Suchpunkt x erzeugt. Und zwar flippen wir in jedem Schritt ein Bit, das komplementär zu x belegt ist, wodurch jeweils genau ein neues Minderheitsbit erzeugt wird.

Der gesuchte Wert, die Zahl von Minderheitsbits in x, ist gleich dem Hammingabstand $H(x^{\text{opt}}, x)$ und der Länge der Folge von 1-Bit-Mutationen. Wir fragen uns daher: Wie viele Minderheitsbits kann man ausgehend von x^{opt} höchstens erzeugen, bevor $x \in A_i$ erreicht wird?

Diese Frage wollen wir mit Hilfe der Fitness beantworten. Auf der Fitness-Skala gilt $f(x^{\text{opt}}) - f(x) = |E| - i$, wir dürfen also nur so viele Minderheitsbits erzeugen, dass die Fitness um höchstens $|E| - i$ absinkt.

Wir wissen bereits, dass $(|E| - i)/(n/k)$ eine *untere* Schranke für die Zahl erzeugbarer Minderheitsbits ist: Um die Fitness ausgehend von x^{opt} um mindestens $|E| - i$ abzusenken, müssen wir mindestens $(|E| - i)/(n/k)$ Minderheitsbits erzeugen, da der Fitnessverlust pro erzeugtem Minderheitsbit höchstens $n/k - 1 < n/k$ ist. (Dies ist das Argument aus Theorem 8.) Wie sieht aber eine *obere* Schranke für die Zahl erzeugbarer Minderheitsbits aus?

In einer einheitlich gefärbten Clique C beträgt der Fitnessbeitrag $\binom{n/k}{2}$. Wenn in einer Clique die maximale Anzahl von $\lfloor n/(2k) \rfloor$ Minderheitsbits erzeugt wird, senkt dies den Fitnessbeitrag in C um einen Term

$$\binom{n/k}{2} - \left(\binom{\lfloor n/(2k) \rfloor}{2} + \binom{\lceil n/(2k) \rceil}{2} \right) = \Theta((n/k)^2).$$

Der durchschnittliche Fitnessverlust pro erzeugtem Minderheitsbit ist daher in der Größenordnung $\Theta(n/k)$; dies gilt auch, falls in C weniger Minderheitsbits erzeugt werden, da gerade die ersten erzeugten Minderheitsbits die Fitness am stärksten senken. Also können wir höchstens $O((|E| - i)/(n/k))$ Minderheitsbits erzeugen; die Zahl der Minderheitsbits in x ist in der Größenordnung $\Theta((|E| - i)/(n/k))$.

Die im Beweis von Theorem 8 verwendete Abschätzung für die Zahl der Minderheitsbits ist also bis auf einen konstanten Faktor scharf. Auch die Abschätzung der erwarteten Zeit, bis die Fitness erhöht wird, ist bis auf einen konstanten Faktor scharf (vergleiche Lemma 5 zur Wahrscheinlichkeit erfolgreicher Schritte). Die Ursache für das unbefriedigende Resultat muss also woanders liegen.

In der Tat ist der Schwachpunkt des Beweises die Wahl der Fitnessschichten. Eine Fitnessschicht umfasst nur Suchpunkte mit einem einzigen Fitnesswert. Da wir in der Abschätzung die Zeiten aufaddieren, um alle Fitnessschichten einmal zu verlassen, gehen wir implizit und pessimistischerweise davon aus, dass wir stets nur einen Fitnessgewinn von 1 haben. (Genauer: ... dass wir die nächsthöhere Fitnessschicht

erreichen. Einige Mengen A_i sind nämlich leer, da die Ising-Funktion nicht surjektiv ist.) Dies trifft in einem typischen Lauf zu Anfang zu, wenn die Mehrheiten nur knapp über dem Minimum liegen, z. B. $\mathrm{maj}(x, C) = n/(2k) + O(1)$ und der Fitnessgewinn bei einer Erhöhung der Mehrheit in der Clique C nur

$$2\,\mathrm{maj}(x, C) - n/k + 1 = O(1)$$

beträgt. Wenn die Mehrheit in C aber eine Höhe von $\mathrm{maj}(x, C) = n/(2k) + \Omega(n/k)$ erreicht hat, ist der Fitnessgewinn in der Größenordnung $\Omega(n/k)$, so dass wir $\Omega(n/k)$ Fitnessschichten in einem Rutsch überspringen.

Experimente zeigen, dass in typischen Läufen nach einer ersten Phase die meisten Fitnesserhöhungen in der Größenordnung $\Theta(n/k)$ liegen. Genau um diesen Faktor ist die Aussage von Theorem 8 schlechter als die angepeilte Schranke $O(n \log n)$. Dennoch ist es schwierig, einen solchen Fitnessgewinn zu garantieren. Es gibt nämlich Suchpunkte, die einen worst case für Fitnesserhöhungen darstellen. Diese lassen sich wie folgt charakterisieren: In einigen Cliquen ist die Mehrheit mit Werten von $\mathrm{maj}(x, \cdot) = \lceil n/(2k) \rceil$ minimal und alle anderen Cliquen sind einheitlich gefärbt mit Mehrheiten von $\mathrm{maj}(x, \cdot) = n/k$.

In einem Suchpunkt mit einer solchen Worst-Case-Verteilung ist in einheitlich gefärbten Cliquen kein Fitnessgewinn möglich. Jede akzeptierte 1-Bit-Mutation erhöht die Fitness nur um 1 oder 2; je nachdem, ob die Cliquengröße gerade oder ungerade ist. Mit Korollar 1 zu lokalen Fitnessveränderungen lässt sich leicht zeigen, dass damit auch der erwartete Fitnessgewinn in einem Schritt durch $O(1)$ beschränkt ist.

Eine Worst-Case-Verteilung der Mehrheitsbits wird in der Praxis wohl nur mit einer überwältigend kleinen Wahrscheinlichkeit erreicht. Dennoch können wir mit der Methode der Fitnessschichten solche (oder ähnliche) Suchpunkte nicht umgehen, da wir innerhalb einer Fitnessschicht implizit immer vom Worst-Case-Suchpunkt der Schicht ausgehen. Zur Veranschaulichung kann man sich auch einen teuflischen Gegenspieler vorstellen, der beim Erreichen einer neuen Fitnessschicht den aktuellen Suchpunkt durch einen Worst-Case-Suchpunkt der gleichen Fitnessschicht ersetzt.

Es ist somit klar, dass wir mit solch einfachen Methoden wie der Methode der Fitnessschichten für einen Großteil eines Laufs stets nur einen konstanten Fitnessgewinn garantieren können. Da der Wertebereich der Ising-Funktion eine Spannweite von $\Theta((n/k)^2)$ Werten umfasst (und wir mit hoher Wahrscheinlichkeit mit kleinen Werten starten), ist von vorn herein klar, dass wir mit konstantem Fitnessgewinn niemals eine obere Schranke von $o((n/k)^2)$ zeigen können, selbst wenn wir unrealistischerweise annehmen, dass jeder Schritt die Fitness erhöht.

Es bleibt uns daher nichts anderes übrig, als andere Analysemethoden zu verwenden.

3.2.2 Alternativer Ansatz: Ein Drift-Argument für das Potenzial φ

Ein anderer Ansatz ist, den Fortschritt des Algorithmus mit der Potenzialfunktion φ zu messen. Dies könnte zum Beweis einer kleineren oberen Schranke hilfreich sein,

da die Funktion φ einen um Faktor $\Theta(n/k)$ kleineren Wertebereich als die Ising-Funktion auf Cliquengraphen mit mehreren unabhängigen Cliquen hat. Wir müssen also weniger oft einen höheren φ-Wert erreichen als eine höhere Fitnessschicht. Anstelle auf der Fitness-Skala die erwartete Zeit abzuschätzen, bis ein Suchpunkt mit maximaler Fitness erreicht wird, können wir analog auf der φ-Skala die erwartete Zeit abschätzen, bis ein Suchpunkt mit maximalem φ-Wert erreicht wird, da genau alle Suchpunkte x mit $\varphi(x) = n$ global optimal sind.

Der φ-Wert des jeweils aktuellen Suchpunkts ist im Allgemeinen nicht monoton über die Zeit, sondern er vollführt einen Random Walk auf einer Skala aus natürlichen Zahlen zwischen dem minimalen φ-Wert ($n/2$, falls die Cliquengröße n/k gerade ist) und dem maximalen Wert n. Jeder dieser Werte entspricht einem möglichen Zustand des Random Walks.

Da sich der φ-Wert in nicht relevanten Schritte nicht verändert, beschränken wir unsere Betrachtungen auf relevante Schritte. Würden wir alle Schritte betrachten, würde der Random Walk mit höheren φ-Werten immer langsamer werden, da relevante Schritte mit sinkender Zahl von Minderheitsbits immer unwahrscheinlicher werden.

Wenn wir den Random Walk auf relevante Schritte beschränken, abstrahieren wir von der Wahrscheinlichkeit relevanter Schritte und den daraus resultierenden Effekten und analysieren statt dessen einen „bereinigten“ Prozess, der nur noch die wesentlichen Übergänge widerspiegelt. Der φ-Wert kann zwar so in manchen Schritten immer noch konstant bleiben, z. B. wenn eine Mehrheit um 1 erhöht und eine andere Mehrheit um 1 gesenkt wird; solche Schritte haben hier jedoch die gleiche Berechtigung wie Schritte, die φ tatsächlich verändern.

Wir können in manchen Szenarien nachweisen, dass der Random Walk von φ in relevanten Schritten eine erwartete Drift in Richtung höherer Werte hat, dass also Erhöhungen gegenüber Senkungen systematisch bevorteilt werden. Wir haben bereits in Abschnitt 1.5.7 gesehen, wie man diese Eigenschaft mit Hilfe der Waldschen Identität (Theorem 4) ausnutzen kann, um die erwartete Zeit, bis ein höherer φ-Wert erreicht wird, nach oben abzuschätzen. Eine gleichwertige Aussage wollen wir nun auch für den Random Walk des (1+1) EA auf der φ-Skala herleiten.

Herleitung und Motivation des Drift-Arguments

Der erste Gedanke ist, die Waldsche Identität auf unseren Random Walk anzuwenden. Dies ist jedoch nicht ohne Weiteres möglich, da die Waldsche Identität verlangt, dass der Random Walk *ortshomogen* ist. Ortshomogenität bedeutet: Die Wahrscheinlichkeit, von Zustand i in den Zustand j überzugehen, hängt nur ab von der Differenz $j - i$, nicht aber von i. Wir haben als in jedem Zustand gleiche Wahrscheinlichkeiten, um eine feste Strecke nach links oder rechts zu springen.

Diese Eigenschaft ist für den in Abschnitt 1.5.7 beschriebenen Random Walk erfüllt, da zum aktuellen Zustand stets Variablen aus der gleichen Wahrscheinlichkeitsverteilung addiert werden. Beim Random Walk auf φ ist die Ortshomogenität aber

nicht gegeben: Während es bei φ-Werten in der Mitte der φ-Skala im Allgemeinen Übergänge nach links und nach rechts gibt, sind für die minimalen und maximalen φ-Werte nur Übergänge in eine Richtung möglich. Der Random Walk ist also nicht ortshomogen.

Es könnte jedoch möglich sein, die Waldsche Identität zu modifizieren, an unser Szenario anzupassen und so trotzdem zum Ziel zu kommen. Droste, Jansen und Wegener [3] präsentieren einen alternativen Beweis für die Aussage der Waldschen Identität, wenn der betrachtete Random Walk ortshomogen ist und der aktuelle Zustand des Random Walks in einem Schritt nur um 1 wachsen kann. Um die Einschränkung, dass der aktuelle Zustand nur um 1 wachsen kann, werden wir uns später kümmern. Zunächst beschreiben wir den Beweis von Droste, Jansen und Wegener [3].

Der Beweis verwendet eine Rekursionsgleichung, um die erwartete Zeit für eine Erhöhung des aktuellen Zustands abzuschätzen. Sei T die Zeit, bis ein um 1 höherer Zustand erreicht wird. Der Algorithmus führt in jedem Fall eine Fitnessauswertung durch; falls sich der aktuelle Zustand um $d = 1$ verändert, haben wir das Ziel in einem Schritt direkt erreicht. Falls sich der aktuelle Zustand um $d \leq 0$ verändert, müssen wir auf insgesamt $(1 - d)$ Erhöhungen des jeweils aktuellen Zustands warten; in einem ortshomogenen Random Walk können wir diese Zeit nach oben durch $(1-d) \cdot \mathrm{E}(T)$ abschätzen. Sei D die Zufallsvariable, die die Veränderung des aktuellen Zustands in einem Schritt angibt, dann erhalten wir folgende Rekursionsgleichung:

$$\mathrm{E}(T) \;=\; 1 + \sum_{d=-\infty}^{1} \mathrm{Prob}(D = d) \cdot (1 - d) \cdot \mathrm{E}(T)$$

Mit einer einfachen Rechnung lässt sich daraus die Aussage der Waldschen Identität (Theorem 4) im Spezialfall $H = 1$ herleiten (siehe [3]).

Was passiert nun, wenn der Random Walk nicht ortshomogen ist? Wir diskutieren folgenden Ansatz. Anstelle der Variablen T und D definieren wir entsprechende Variablen T_i und D_i ausgehend von Zustand i. Wenn der Algorithmus nun ausgehend vom Zustand i in einen Zustand $j < i$ übergeht und $i - j = d$ ist, müssen wir die erwarteten Zeiten aufsummieren, bis der Algorithmus die Zustände $j, \ldots, i$ jeweils erhöht hat. Wir erhalten

$$\mathrm{E}(T_i) \;=\; 1 + \sum_{d=-\infty}^{1} \mathrm{Prob}(D_i = d) \cdot (\mathrm{E}(T_{i-d}) + \cdots + \mathrm{E}(T_i))$$

Es scheint plausibel, dass die Erhöhung von φ auf kleineren Zuständen nicht schwerer sein kann, da sie mehr Übergänge nach rechts zulassen als größere Zustände. Angenommen, es gilt $\mathrm{E}(T_j) \leq \mathrm{E}(T_i)$ für $j < i$. Wenn wir dies in unsere Gleichung einsetzen, erhalten wir die Gleichung von Droste, Jansen und Wegener [3] als Ungleichung:

$$\mathrm{E}(T_i) \;\leq\; 1 + \sum_{d=-\infty}^{1} \mathrm{Prob}(D_i = d) \cdot (1 - d) \cdot \mathrm{E}(T_i)$$

Hieraus ließe sich wieder die Aussage der Waldschen Identität für $H = 1$ herleiten.

Die Verwendung des Konjunktivs lässt erahnen, dass hier etwas nicht stimmt. Der Ansatz hat einen gravierenden Haken, und zwar die Annahme, dass mit $\mathrm{E}(T_j) \leq \mathrm{E}(T_i)$ für $j < i$ eine Erhöhung von φ auf kleineren Zuständen generell nicht schwieriger sein kann. Es kann nämlich sein, dass es einen Suchpunkt x gibt mit $\varphi(x) = i-1$ und einen Suchpunkt y mit $\varphi(y) = i$, aber die Verteilungen der Mehrheitsbits unterschiedlich aussehen.

In x können die Mehrheiten so verteilt sein, dass Sprünge nach links ermöglicht werden, z. B. indem es einige Cliquen mit mittelgroßen Mehrheiten gibt, in denen der erwartete Fitnessgewinn durch 1-Bit-Mutationen hoch ist, und andere Cliquen mit kleineren Mehrheiten, die aufgrund des kleineren „Gewichts“ weiter verkleinert werden können. In y hingegen können die Mehrheiten ungefähr gleich groß sein, so dass alle Bits ähnliches Gewicht haben und Sprünge nach links unwahrscheinlich sind, da die Fitness schnell sinken kann, wenn mehr Mehrheitsbits flippen als Minderheitsbits. Hier hätten wir also ein Szenario, in dem eine Erhöhung des φ-Werts ausgehend von x länger dauern kann als eine Erhöhung des φ-Werts ausgehend von y, die Ungleichung $\mathrm{E}(T_j) \leq \mathrm{E}(T_i)$ für $j < i$ ist also im Allgemeinen falsch.

Die beiden Beispiel-Verteilungen zeigen auch, dass der aktuelle φ-Wert nicht immer ein zuverlässiges Maß für den Fortschritt des Algorithmus ist. Wenn ein Suchpunkt y mit $\varphi(y) = i$ eine ungünstige Verteilung von Mehrheitsbits hat, sind wir eventuell sogar in einer schlechteren Lage als mit einem Suchpunkt x mit $\varphi(x) = i - 1$ und einer vorteilhafteren Verteilung der Mehrheitsbits.

Die wahre Ursache dieser überraschenden Phänomene liegt tiefer: Wenn wir uns nur auf den φ-Wert eines Suchpunkts beschränken, ignorieren wir die Verteilung der Mehrheitsbits und verlieren dadurch wertvolle Informationen. Es ist also nicht ohne Weiteres möglich, den Suchprozess des (1+1) EA auf einen Random Walk in einem eindimensionalen Raum zu reduzieren. Oder anders ausgedrückt: Die Potenzialfunktion $\varphi(x)$ ist keine *suffiziente Statistik*.

Die Gründe warum wir näher auf die genannten Probleme eingehen, sind folgende. Zum einen soll aus der Diskussion klar werden, dass die Analyse des (1+1) EA auf Cliquengraphen mit unabhängigen Cliquen schwieriger ist, als es auf den ersten Blick aussieht. Einfache Lösungen schlagen fehl; dies gilt sowohl für fitnessbasierte Ansätze als auch für einfache Drift-Argumente. Zweitens soll die Vorgehensweise im Rest dieses Kapitels motiviert werden, da sie dem Leser ohne diese Erläuterungen möglicherweise abstrus und unnötig komplex erscheinen würde.

Ein rigoroser Beweis für das Drift-Argument auf φ

Aus der Diskussion einfacher Drift-Ansätze können wir folgende Schlüsse ziehen:

1. Mehrere Suchpunkte mit gleichem φ-Wert können unterschiedliche erwartete Zeiten haben, bis φ erhöht wird. Wenn wir der Einfachheit halber in Abschätzungen alle Suchpunkte mit gleichem φ-Wert gleich behandeln wollen, sollten wir unter all diesen Suchpunkten die *worst-case erwartete Zeit* betrachten.

2. Wir sollten ständig im Hinterkopf behalten, dass der Suchprozess im Suchraum $\{0,1\}^n$ stattfindet und dass die Potenzialfunktion lediglich ein Maß ist, das den Suchraum (unter Verlust von Informationen) auf eine eindimensionale Skala projiziert.

Letztere Vorstellung spiegelt sich in der Notation wider, die wir nun einführen wollen. Das Drift-Argument wollen wir nicht nur auf die Funktion φ anwenden, sondern in Abschnitt 3.3 beispielsweise auch auf einzelne Mehrheiten $\mathrm{maj}(x,\cdot)$. Daher abstrahieren wir von φ und sprechen von einer allgemeinen Projektion α, die ein Maß für den Fortschritt des Algorithmus darstellt.

Die Funktion α spezifizieren wir als eine Abbildung von $\{0,1\}^n$ in ein Intervall $[a,b]$ ganzer Zahlen, $a,b \in \mathbb{Z}$:

$$\alpha\colon \{0,1\}^n \to \mathbb{Z} \cap [a,b]$$

Als nächstes partitionieren wir den Suchraum analog zu Fitnessschichten in so genannte α-*Schichten*: Die i-te α-Schicht umfasst dabei genau alle Suchpunkte x mit $\alpha(x) = i$. Genau wie man bei Fitnessschichten implizit unter allen Suchpunkten der aktuellen Fitnessschicht die worst-case erwartete Zeit betrachtet, bis eine höhere Fitnessschicht erreicht wird, betrachten wir nun unter allen Suchpunkten der aktuellen α-Schicht die worst-case erwartete Zeit, bis eine höhere α-Schicht erreicht wird. Der einzige gravierende Unterschied zwischen Fitnessschichten und α-Schichten ist, dass Fitnessschichten nicht zugunsten schlechterer Fitnessschichten verlassen werden können.

Folgendes Lemma beweist das angestrebte Drift-Argument. Als Voraussetzung verwenden wir die bedingte erwartete Drift $\mathrm{E}(D_x \mid D_x \leq 1)$ unter der Bedingung $D_x \leq 1$. Die Gründe hierfür lassen sich leicht motivieren. Angenommen, die Bedingung $D_x \leq 1$ wäre nicht gegeben. Dann stellen wir uns einen Random Walk auf $\{0,1\}^n$ mit α-Werten $1,\ldots,2^n$ vor, der mit dem minimalen α-Wert 1 startet und der nur einen einzigen Übergang durchführt: Und zwar springt er mit einer Wahrscheinlichkeit 2^{-n} direkt zu einem Suchpunkt mit α-Wert 2^n und bleibt ansonsten beim aktuellen Suchpunkt mit α-Wert 1. Die erwartete Drift ist 1, trotzdem beträgt die erwartete Zeit, bis ein höherer α-Wert erreicht wird, 2^n.

Dieses Phänomen wird als *overstepping* bezeichnet. Wir warten lediglich auf eine kleine Erhöhung des α-Wertes; bei der einzigen Erhöhung schießen wir jedoch weit über das Ziel hinaus. Solche Ausreißer neigen dazu, die erwartete Drift zu dominieren und zu verfälschen. Die Bedingung $D_x \leq 1$ verhindert das overstepping, da wir nur Schritte in die Drift einbeziehen, die den α-Wert um höchstens 1 erhöhen. Dadurch wird die Aussagekraft der erwarteten Drift wesentlich erhöht. Wenn die erwartete Drift hoch ist, selbst wenn wir nur Erhöhungen von höchstens 1 in Betracht ziehen, ist dies eine stärkere Aussage als eine hohe unbedingte erwartete Drift.

Wir werden im Folgenden stets kurz von „erwarteter Drift“ sprechen. Damit ist aber implizit stets die bedingte erwartete Drift gemeint unter der Bedingung, dass der aktuelle Wert nur um höchstens 1 wächst.

Lemma 6. *Mit T_x für $x \in \{0,1\}^n$ bezeichnen wir die zufällige Zahl der Schritte, bis der (1+1) EA ausgehend von x einen Suchpunkt z mit $\alpha(z) > \alpha(x)$ erreicht. Sei y der Suchpunkt, der ausgehend von x im nächsten Schritt erreicht wird und $D_x := \alpha(y) - \alpha(x)$ eine Zufallsvariable, die abhängig von x die Veränderung des α-Werts im nächsten Schritt angibt. Den Wert r definieren wir als*

$$r := \min_{x,\alpha(x)<b} \mathrm{E}(D_x \mid D_x \leq 1).$$

Falls $r > 0$, gilt für alle $x \in \{0,1\}^n$ mit $\alpha(x) < b$:

$$\mathrm{E}(T_x) \leq \frac{1}{r}.$$

Dies gilt auch dann, wenn wir zwischen zwei Schritten den aktuellen Suchpunkt durch einen anderen Suchpunkt mit gleichem α-Wert ersetzen.

Beweis. Für den Parameter r gelten folgende Ungleichungen, die wir später benötigen werden. Für alle $x \in \{0,1\}^n$ mit $\alpha(x) < b$ gilt

$$r^{-1} \geq \frac{1}{\mathrm{E}(D_x \mid D_x \leq 1)} \geq 1,$$

wobei letztere Ungleichung gilt, da $0 < \mathrm{E}(D_x \mid D_x \leq 1) \leq 1$.

Wir zerlegen nun den Suchraum in Schichten gemäß der Funktion α. Für $a \leq i \leq b$ definieren wir

$$A_i := \{x \in \{0,1\}^n \mid \alpha(x) = i\}$$

und die worst-case erwartete Zeit in Schicht A_i als

$$E_i := \max_x \{\mathrm{E}(T_x) \mid x \in A_i\}.$$

Damit genügt es, die Schranke r^{-1} für alle Werte E_i nachzuweisen.

Wir zeigen die Behauptung $E_i \leq r^{-1}$ per Induktion über i, sprich per Induktion über die α-Schichten. Für den Induktionsanfang nutzen wir aus, dass der Wertebereich der Projektion α endlich ist, denn bei einer entsprechenden erwarteten Drift wird die kleinste α-Schicht schnell nach „oben“ verlassen. Im Induktionsschluss verwenden wir eine abgewandelte Form der Rekursionsgleichung von Droste, Jansen und Wegener [3], die wir bereits erläutert haben.

Induktionsanfang: In der kleinsten α-Schicht A_a können wir, wenn wir A_a verlassen, nur Suchpunkte mit höherem α-Wert erreichen. Mit einer Wahrscheinlichkeit von $\mathrm{Prob}(D_x \geq 1)$ erreichen wir direkt von x aus unser Ziel, ansonsten erreichen wir einen Suchpunkt $y \in A_a$. Wir wollen nun die Wahrscheinlichkeit $\mathrm{Prob}(D_x \geq 1)$ nach unten abschätzen. Da wir eine positive erwartete Drift haben, lässt sich eine solche Schranke direkt aus der erwarteten Drift folgern.

Wenn wir die erwartete Drift unter $D_x \leq 1$ in die (bedingten) disjunkten Ereignisse $D_x = 1$ und $D_x \leq 0$ (gegeben $D_x \leq 1$) aufspalten, gilt für alle $x \in A_a$ nach Definition bedingter Erwartungswerte

$$\begin{aligned} \mathrm{E}(D_x \mid D_x \leq 1) &= \mathrm{E}(D_x \mid D_x = 1) \cdot \mathrm{Prob}(D_x = 1 \mid D_x \leq 1) \\ &\quad + \mathrm{E}(D_x \mid D_x = 0) \cdot \mathrm{Prob}(D_x = 0 \mid D_x \leq 1) \\ &= 1 \cdot \mathrm{Prob}(D_x = 1 \mid D_x \leq 1) + 0 \cdot \mathrm{Prob}(D_x = 0 \mid D_x \leq 1) \\ &= \mathrm{Prob}(D_x = 1 \mid D_x \leq 1) \end{aligned}$$

Dies ist noch nicht die gewünschte Schranke für $\mathrm{Prob}(D_x \geq 1)$; wir können jedoch mit folgender Rechnung zeigen, dass $\mathrm{Prob}(D_x \geq 1) \geq \mathrm{Prob}(D_x = 1 \mid D_x \leq 1)$ ist.

$$\begin{aligned} \mathrm{Prob}(D_x \geq 1) &= \mathrm{Prob}(D_x = 1) + \mathrm{Prob}(D_x > 1) \\ &= \mathrm{Prob}(D_x = 1 \mid D_x \leq 1) \cdot \mathrm{Prob}(D_x \leq 1) + \mathrm{Prob}(D_x > 1) \\ &\overset{!}{\geq} \mathrm{Prob}(D_x = 1 \mid D_x \leq 1) \cdot \mathrm{Prob}(D_x \leq 1) \\ &\quad + \mathrm{Prob}(D_x = 1 \mid D_x \leq 1) \cdot \mathrm{Prob}(D_x > 1) \end{aligned}$$

Die Ungleichung hierbei gilt, da $\mathrm{Prob}(D_x = 1 \mid D_x \leq 1) \leq 1$. Wenn wir nun $\mathrm{Prob}(D_x > 1)$ durch $1 - \mathrm{Prob}(D_x \leq 1)$ ersetzen, erhalten wir

$$\begin{aligned} \mathrm{Prob}(D_x \geq 1) &\geq \mathrm{Prob}(D_x = 1 \mid D_x \leq 1) \cdot \mathrm{Prob}(D_x \leq 1) \\ &\quad + \mathrm{Prob}(D_x = 1 \mid D_x \leq 1) \cdot (1 - \mathrm{Prob}(D_x \leq 1)) \\ &= \mathrm{Prob}(D_x = 1 \mid D_x \leq 1). \end{aligned}$$

Insgesamt haben wir also gezeigt:

$$\mathrm{Prob}(D_x \geq 1) \geq \mathrm{E}(D_x \mid D_x \leq 1)$$

Mit dieser Erkenntnis können wir nun die worst-case erwartete Zeit abschätzen, bis wir die kleinste α-Schicht verlassen. Es gilt

$$\begin{aligned} E_a &\leq \max_{x \in A_a} \left\{ \frac{1}{\mathrm{Prob}(D_x \geq 1)} \right\} \\ &\leq \max_{x \in A_a} \left\{ \frac{1}{\mathrm{E}(D_x \mid D_x \leq 1)} \right\} \overset{\text{Def.}}{\leq} \frac{1}{r}. \end{aligned}$$

Induktionsvoraussetzung: Angenommen, $E_j \leq r^{-1}$ gilt für alle j mit $a \leq j < i$.

Induktionsschluss: Wir zeigen nun $E_i \leq r^{-1}$. Dazu leiten wir eine Rekursionsgleichung her, die der bereits behandelten Gleichung von Droste, Jansen und Wegener [3] ähnelt.

Ausgehend von x führen wir in jedem Fall einen Schritt durch und erreichen einen Suchpunkt y. Falls $D_x := \alpha(y) - \alpha(x) > 0$, haben wir unser Ziel direkt erreicht. Falls $D_x < 0$, müssen wir ausgehend von y warten, bis sich der α-Wert des jeweils

aktuellen Suchpunkts auf mehr als i erhöht hat. Die erwartete Zeit ist in diesem Fall beschränkt durch $E_j + \cdots + E_i$, wobei wir in jeder erreichten Schicht die worst-case erwartete Zeit anrechnen.

Für E_i gilt daher folgende Rekursionsgleichung:

$$E_i \leq 1 + \sum_{d=-\infty}^{0} \text{Prob}(D_x = d) \cdot (E_{i-d} + \cdots + E_i)$$

Auf die Summanden E_j mit $j < i$ können wir die Induktionsvoraussetzung anwenden; falls wir von α-Schicht A_i aus auf der α-Skala nach „unten“ springen, wissen wir, dass wir relativ schnell wieder nach „oben“ zurückkehren. Allerdings bleibt der Summand E_i erhalten, denn eventuell erreichen wir wieder einen Suchpunkt in A_i und finden uns in der Ausgangsposition wieder.

Die Rekursionsgleichung können wir mit der Induktionsvoraussetzung vereinfachen zu

$$E_i \leq 1 + \sum_{d=-\infty}^{0} \text{Prob}(D_x = d) \cdot ((-d) \cdot r^{-1} + E_i)$$

Diese Gleichung wollen wir im Folgenden auflösen. Es gilt

$$\begin{aligned} E_i &\leq 1 + \sum_{d=-\infty}^{0} \text{Prob}(D_x = d) \cdot ((-d) \cdot r^{-1} + E_i) \\ &= 1 - \sum_{d=-\infty}^{0} \text{Prob}(D_x = d) \cdot d \cdot r^{-1} + \sum_{d=-\infty}^{0} \text{Prob}(D_x = d) \cdot E_i \\ &= 1 - \sum_{d=-\infty}^{0} \text{Prob}(D_x = d) \cdot d \cdot r^{-1} + \text{Prob}(D_x \leq 0) \cdot E_i \end{aligned}$$

Wenn wir nun von beiden Seiten $\text{Prob}(D_x \leq 0) \cdot E_i$ subtrahieren, erhalten wir auf der linken Seite den Term $E_i \cdot (1 - \text{Prob}(D_x \leq 0))$. Wir teilen beide Seiten durch den Faktor $1 - \text{Prob}(D_x \leq 0)$ und erhalten

$$\begin{aligned} E_i &\leq \frac{1}{1 - \text{Prob}(D_x \leq 0)} \cdot \left(1 - \sum_{d=-\infty}^{0} \text{Prob}(D_x = d) \cdot d \cdot r^{-1}\right) \\ &= \frac{1}{\text{Prob}(D_x \geq 1)} \cdot \left(1 - r^{-1} \sum_{d=-\infty}^{0} \text{Prob}(D_x = d) \cdot d\right) \quad (*) \end{aligned}$$

Die Summe über d werden wir nun in einer Nebenrechnung umformen. Zunächst

benötigen wir dafür folgende Hilfsaussage.

$$\begin{aligned}
\mathrm{E}(D_x \mid D_x \leq 1) &= \sum_{d=-\infty}^{\infty} \mathrm{Prob}(D_x = d \mid D_x \leq 1) \cdot d \\
&= \sum_{d=-\infty}^{\infty} \frac{\mathrm{Prob}(D_x = d, D_x \leq 1)}{\mathrm{Prob}(D_x \leq 1)} \cdot d \\
&= \sum_{d=-\infty}^{1} \frac{\mathrm{Prob}(D_x = d)}{\mathrm{Prob}(D_x \leq 1)} \cdot d
\end{aligned}$$

Multiplikation der Gleichung mit $\mathrm{Prob}(D_x \leq 1)$ ergibt

$$\sum_{d=-\infty}^{1} \mathrm{Prob}(D_x = d) \cdot d \;=\; \mathrm{E}(D_x \mid D_x \leq 1) \cdot \mathrm{Prob}(D_x \leq 1) \quad (**).$$

Damit können wir die Summe über d aus Gleichung $(*)$ nun umformen. Im ersten Schritt addieren wir eine „intelligente Null“, um einen zusätzlichen Summanden für $d = 1$ zu erzeugen.

$$\begin{aligned}
& \sum_{d=-\infty}^{0} \mathrm{Prob}(D_x = d) \cdot d \\
\overset{+0}{=} & \sum_{d=-\infty}^{1} \mathrm{Prob}(D_x = d) \cdot d - \sum_{d=1}^{1} \mathrm{Prob}(D_x = d) \cdot d \\
= & \sum_{d=-\infty}^{1} \mathrm{Prob}(D_x = d) \cdot d - \mathrm{Prob}(D_x = 1) \\
\overset{(**)}{=} & \mathrm{E}(D_x \mid D_x \leq 1) \cdot \mathrm{Prob}(D_x \leq 1) - \mathrm{Prob}(D_x = 1) \quad (***)
\end{aligned}$$

Damit ist unsere Nebenrechnung beendet. Wenn wir $(***)$ in $(*)$ einsetzen, erhalten wir

$$E_i \;\leq\; \frac{1}{\mathrm{Prob}(D_x \geq 1)} \cdot \left(1 - \frac{\mathrm{E}(D_x \mid D_x \leq 1) \cdot \mathrm{Prob}(D_x \leq 1)}{r} + \frac{\mathrm{Prob}(D_x = 1)}{r}\right)$$

Nun müssen wir nur noch diesen Ausdruck weiter auflösen. Da nach Voraussetzung $r^{-1} \geq 1/\,\mathrm{E}(D_x \mid D_x \leq 1)$, folgt $\mathrm{E}(D_x \mid D_x \leq 1) \cdot r^{-1} \geq 1$ und

$$\begin{aligned}
E_i &\leq \frac{1}{\mathrm{Prob}(D_x \geq 1)} \cdot \left(1 - \mathrm{Prob}(D_x \leq 1) + \mathrm{Prob}(D_x = 1) \cdot r^{-1}\right) \\
&= \frac{1}{\mathrm{Prob}(D_x \geq 1)} \cdot \left(\mathrm{Prob}(D_x > 1) + \mathrm{Prob}(D_x = 1) \cdot r^{-1}\right)
\end{aligned}$$

Da zudem $r^{-1} \geq 1$, gilt $\mathrm{Prob}(D_x > 1) \leq \mathrm{Prob}(D_x > 1) \cdot r^{-1}$ und somit

$$\begin{aligned}
E_i &\leq \frac{1}{\mathrm{Prob}(D_x \geq 1)} \cdot \left(\mathrm{Prob}(D_x > 1) \cdot r^{-1} + \mathrm{Prob}(D_x = 1) \cdot r^{-1}\right) \\
&= \frac{1}{\mathrm{Prob}(D_x \geq 1)} \cdot \mathrm{Prob}(D_x \geq 1) \cdot r^{-1} = r^{-1}.
\end{aligned}$$

Da wir stets die worst-case erwarteten Zeiten in einer Schicht A_i betrachten, gilt die Aussage auch dann, wenn wir zwischen zwei Schritten den aktuellen Suchpunkt durch einen anderen Suchpunkt der gleichen α-Schicht ersetzen. □

Die Aussage, dass wir einen Suchpunkt zwischen zwei Schritten durch andere Suchpunkte der gleichen α-Schicht ersetzen können, ist für unsere weiteren Betrachtungen wichtig. Und zwar betrachten wir unter anderem den Random Walk von $\alpha := \varphi$, eingeschränkt auf relevante Schritte. In nicht relevanten Schritten können beispielsweise Nullen und Einsen innerhalb einer Clique ausgetauscht werden. Wenn der Schritt nicht relevant ist, bleibt der φ-Wert bei dieser Operation konstant. Trotzdem liegt beim nächsten relevanten Schritt ein anderer Suchpunkt vor, so dass ein solcher Austausch zwischen relevanten Schritten berücksichtigt werden muss.

3.2.3 Eine Schranke $O(k \cdot n \log n)$ für den (1+1) EA

Das erarbeitete Drift-Argument wollen wir nun anwenden, um eine einfache obere Schranke für die erwartete Optimierungszeit des (1+1) EA zu zeigen, die zumindest für eine konstante Zahl von Cliquen scharf ist.

Wie bereits motiviert, betrachten wir nun den Random Walk des (1+1) EA auf der Skala von φ in erfolgreichen Schritten. Um das Drift-Argument aus Lemma 6 anzuwenden, zeigen wir, dass dieser Random Walk in erfolgreichen Schritten eine erwartete Drift von mindestens $1/k$ hat, selbst wenn sich der aktuelle Zustand nur um höchstens 1 erhöht.

Theorem 9. *Gegeben ein Cliquengraph $G = (V, E)$ mit $n := |V|$ Knoten und $1 \leq k \leq n/2$ unabhängigen Cliquen der Größe n/k. Die erwartete Optimierungszeit des (1+1) EA auf $f := \text{Ising}_G$ ist beschränkt durch $O(k \cdot n \log n)$.*

Beweis. Mit A bezeichnen wir das Ereignis, dass ein Schritt des (1+1) EA erfolgreich ist. Sei x der aktuelle Suchpunkt des (1+1) EA und x' der Suchpunkt, der in einem erfolgreichen Schritt aus x erzeugt wird. Zu zeigen ist nun eine gute untere Schranke für die erwartete Drift $\mathrm{E}(D_x \mid D_x \leq 1, A)$ in erfolgreichen Schritten, selbst wenn φ nur um höchstens 1 wächst.

Nach Definition erfolgreicher Schritte erhöht sich in einer Clique C^* die Mehrheit $\text{maj}(x, C^*)$ um mindestens 1. Es gibt also Bitflips, die einen positiven Effekt auf die erwartete Drift haben. Wie viele Bitflips können einen negativen Effekt auf die erwartete Drift haben?

In der Menge $V \setminus C^*$ gibt es $n - n/k$ andere Bits, die flippen können und damit im Extremfall $\varphi(x)$ um jeweils 1 senken können. Die unbedingte erwartete Zahl flippender Bits aus $n - n/k$ Bits ist $1/n \cdot (n - n/k) = 1 - 1/k$. Allerdings betrachten wir nur erfolgreiche Schritte, so dass wir immer bedingte Erwartungswerte unter der Bedingung A betrachten.

Welche Auswirkungen hat die Bedingung A auf die erwartete Zahl negativ flippender Bits? Es ist plausibel, dass i. A. in erfolgreichen Schritten nicht viele Bitflips

Mehrheiten senken können, da ansonsten die Fitness zu stark sinkt und der Nachkomme nicht akzeptiert wird. Dies zeigt sich auch in Experimenten; es ist allerdings schwierig, diesen Effekt zu berücksichtigen, da die Mehrheiten sehr unterschiedlich aussehen können.

Wir wählen deshalb einen anderen Weg und betrachten ein pessimistischeres Szenario. Und zwar gehen wir davon aus, dass die positiven Effekte in C^* so groß sind, dass sie alle negativen Effekte in anderen Cliquen dominieren. Auf diese Weise erreichen wir, dass die Effekte negativ flippender Bits in $V \setminus C^*$ unabhängig von Ereignis A und der Akzeptanz des Nachkommens sind.

Mit diesem Hintergrund nehmen wir pessimistischerweise an, dass alle flippenden Bits in $V \setminus C^*$ das Potenzial φ um jeweils 1 senken. Diese Annahme ist in drei Punkten pessimistisch:

1. Wir ignorieren, dass es unter den flippenden Bits Minderheitsbits geben kann.
2. Wir ignorieren kippende Mehrheiten (wodurch die Mehrheit einer Clique C effektiv um weniger als die Zahl flippender Bits in C sinkt).
3. Wir ignorieren negative Einflüsse flippender Bits auf die Fitness.

Wenn wir eine untere Schranke für die erwartete Drift $\mathrm{E}(D_x \mid D_x \leq 1, A)$ in diesem pessimistischen Szenario zeigen, gilt diese untere Schranke auch ohne diese pessimistischen Annahmen.

Sei $D_x := \varphi(x') - \varphi(x)$ die Erhöhung von φ. Die erwartete Zahl flippender Bits in $V \setminus C^*$ ist $|V \setminus C^*|/n = (n - n/k)/n = 1 - 1/k$. Selbst in dem Fall, dass sich die Mehrheit in C^* nur um 1 erhöht, ist erwartete Drift von φ mindestens

$$\mathrm{E}(D_x \mid D_x \leq 1, A) \;\geq\; 1 - \left(1 - \frac{1}{k}\right) = \frac{1}{k}.$$

Diese Schranke gilt für alle nicht optimalen Suchpunkte x, sprich für $\varphi(x) < n$. Wir können daher das Drift-Argument aus Lemma 6 mit $\alpha := \varphi$ anwenden, wenn wir den Random Walk auf erfolgreiche Schritte beschränken. Wir haben bereits angesprochen, dass es zwischen zwei erfolgreichen Schritten nicht erfolgreiche Schritte geben kann, in dem ein Suchpunkt x durch einen anderen Suchpunkt $x' \neq x$ ersetzt wird. Da sich der φ-Wert dabei nach Definition erfolgreicher Schritte aber nicht ändern kann, lässt sich Lemma 6 auch in dieser Situation anwenden.

Aus Lemma 6 folgt, dass die erwartete Zahl erfolgreicher Schritte, bis der (1+1) EA auf der φ-Skala einen höheren Rekordwert erreicht, durch k beschränkt ist.

Wie lange warten wir auf einen erfolgreichen Schritt? Eine hinreichende Bedingung für einen erfolgreichen Schritt ist eine 1-Bit-Mutation, die ein Minderheitsbit flippt. Da es in x genau $n - \varphi(x)$ Minderheitsbits gibt, hat ein erfolgreicher Schritt eine Wahrscheinlichkeit von mindestens $(n - \varphi(x))/(en)$.

Eine hinreichende Bedingung zum Erreichen eines globalen Optimums ist, dass alle φ-Werte von $\lceil n/2 \rceil$ bis $n-1$ durch einen höheren Rekordwert übertroffen werden. Wenn i der aktuelle Rekordwert für φ ist, ist $en/(n-i)$ eine obere Schranke für die erwartete Zeit auf einen erfolgreichen Schritt; diese obere Schranke gilt auch dann, wenn zwischendurch Suchpunkte mit kleinerem φ-Wert als i erreicht werden. Die erwartete Optimierungszeit ist daher nach oben beschränkt durch

$$1 + \sum_{i=\lceil n/2 \rceil}^{n-1} \frac{e \cdot n}{n-i} \cdot k = O(k \cdot n \log n).$$

Die „1“ steht wie bei Fitnessschichten für die Auswertung eines globalen Optimums. □

Das pessimistische Szenario, dass jedes weitere flippende Bit in $V \setminus C^*$ das Potenzial um jeweils 1 senkt, wird uns noch öfter begegnen. Mit Hinblick auf die Analyse von zwei Cliquen mit einer Brückenkante in Abschnitt 3.3 sei zudem gesagt, dass das sich die Abschätzung der erwarteten Drift in erfolgreichen Schritten auch auf beliebige Cliquengraphen mit Brückenkanten übertragen lässt. Dadurch, dass wir den positiven Effekt auf die Fitness in der Clique C^* (deren Existenz auch mit Brückenkanten aus der Definition erfolgreicher Schritte folgt) überschätzen und die negativen Einflüsse ignorieren, abstrahieren wir vollständig von den konkreten Fitnessveränderungen und somit auch von den Effekten möglicher Brückenkanten auf die Fitness.

Wir haben also auch auf beliebigen Cliquengraphen mit k gleich großen Cliquen eine erwartete Drift von mindestens $1/k$ in erfolgreichen Schritten. Dieses Ergebnis halten wir für später in einem Korollar fest.

Korollar 2. *Gegeben ein beliebiger Cliquengraph $G = (V, E)$ mit $n := |V|$ Knoten und $1 \leq k \leq n/2$ disjunkten Cliquen der Größe n/k. Sei x der aktuelle Suchpunkt des (1+1) EA, x' der aus x erzeugte Nachkomme, $D_x := \varphi(x') - \varphi(x)$ und A das Ereignis, dass ein Schritt des (1+1) EA erfolgreich ist. Dann gilt*

$$\mathrm{E}(D_x \mid D_x \leq 1, A) \quad \geq \quad \frac{1}{k}.$$

Um Missverständnissen vorzubeugen sei darauf hingewiesen, dass wir in Korollar 2 nur erfolgreiche Schritte betrachten. Bei Cliquengraphen mit Brückenkanten kann es auch relevante Schritte geben, die nicht erfolgreich sind, d. h. mit hohen Fitnessgewinnen durch Brückenkanten kann φ eventuell in nicht erfolgreichen Schritten sinken. Es ist also nicht korrekt, sich allein auf den Random Walk in erfolgreichen Schritten zu beschränken; diesem Problem werden wir uns aber erst in Abschnitt 3.3 stellen.

3.2.4 Eine obere Schranke für einen modifizierten Algorithmus

Aus dem Beweis von Theorem 9 wird deutlich, wo die nächste Schwierigkeit bei der Analyse unabhängiger Cliquen liegt: Wir müssen eine ausreichend große untere

Schranke für die erwartete Drift nachweisen. In Theorem 9 können wir nur eine untere Schranke von $1/k$ für die erwartete Drift zeigen, was gemäß unserem Drift-Argument aus Lemma 6 zu einem Faktor k in der Schranke $O(k \cdot n \log n)$ für die erwartete Optimierungszeit führt.

Unser nächstes Ziel ist daher, eine bessere untere Schranke für die erwartete Drift zu entwickeln.

Bei der Einführung des pessimistischen Szenarios („jedes weitere flippende Bit senkt φ um jeweils 1“) haben wir bereits angeführt, dass in typischen Schritten meist nur wenige Bits negativ für das Potenzial φ flippen dürfen, da ansonsten die Fitness typischerweise zu stark absinkt und der Nachkomme nicht akzeptiert wird. Wegen der großen Bandbreite von Mehrheiten und lokaler Fitnessveränderungen (zwischen ± 1 und $\pm(n/k - 1)$) ist dieser Effekt aber nur schwer kontrollierbar.

Die Argumentation wird einfacher, wenn alle Cliquen große Mehrheiten haben, so dass alle lokalen Fitnessveränderungen durch flippende Minderheitsbits in der gleichen Größenordnung $\Theta(n/k)$ liegen. Für solch einen Suchpunkt lässt sich zeigen, dass pro steigender Mehrheit nicht allzu viele Mehrheiten sinken können, da ansonsten die Fitnessveränderung negativ wird. Auf diese Art und Weise kann man die negativen Effekte von Bitflips eingrenzen und mit passenden Rechnungen die erwartete Drift durch eine positive Konstante nach unten beschränken.

Dieser Ansatz klingt viel versprechend, allerdings sind mehrere Hürden zu überwinden:

1. Zu Anfang eines Laufs sind typischerweise alle Mehrheiten klein und das Potenzial φ ist niedrig.

   Dieses Problem lässt sich aber leicht lösen: So lange $\varphi = n - \Theta(n)$ gilt, gibt es $\Theta(n)$ Minderheitsbits. Flippende Minderheitsbits können φ aber i. A. nicht senken, daher lässt sich in diesem Fall mit einem anderen abgewandelten pessimistischen Szenario eine erwartete Drift von $\Omega(1)$ nachweisen.

2. Auch wenn $\varphi = n - o(n)$ ist, kann es bei sehr asymmetrischen Verteilungen von Mehrheitsbits immer noch Cliquen mit kleinen Mehrheiten geben.

   Hier müssen wir von unserer Vorstellung absehen, dass nämlich alle Cliquen große Mehrheiten haben; wir können allerdings zeigen, dass die meisten Cliquen große Mehrheiten haben, wenn das Potenzial ausreichend groß ist.

3. Es kann mehrere Cliquen geben, in denen die Mehrheiten steigen und in denen es daher Fitnessverbesserungen gibt. Jede Fitnessverbesserung birgt die Gefahr, dass dadurch woanders an mehreren Stellen Fitnessverluste durch sinkende Mehrheiten ermöglicht werden.

Eine Lösung für das dritte Problem ist, dass wir uns auf Schritte konzentrieren, in denen nur eine Mehrheit um höchstens 1 wachsen kann. Dadurch wird der mögliche Fitnessgewinn künstlich beschnitten; wenn alle positiven Effekte zusammen nur noch zu einem Fitnessgewinn von höchstens s führen, können keine negativen Effekte mehr

auftreten, die zusammen die Fitness um mehr als s senken, da dann die gesamte Fitnessveränderung negativ ist und der Nachkomme nicht mehr akzeptiert wird.

Vielen sehr schlechten Operationen, die das Potenzial stark senken, wird so der Nährboden entzogen. Insgesamt werden die Auswirkungen von Schritten des Algorithmus kontrollierbarer.

Wir definieren nun einen modifizierten Algorithmus, der nur noch Schritte akzeptiert, in denen der mögliche Fitnessgewinn begrenzt ist, da nur noch eine Mehrheit um höchstens 1 wachsen kann. Diesen modifizierten (1+1) EA bezeichnen wir als *(1+1)* EA* .

Algorithmus 3 ((1+1)* EA).

1. *Initialisierung: Wähle $x \in \{0,1\}^n$ uniform zufällig.*
2. *$y := x$. Flippe jedes Bit in y mit Wahrscheinlichkeit $1/n$ unabhängig von den anderen Bits.*
3. *Falls $f(y) \geq f(x)$ und es eine Clique C_i gibt, $1 \leq i \leq k$, so dass gilt*

$$\begin{aligned} \mathrm{maj}(y, C_i) &\leq \mathrm{maj}(x, C_i) + 1 \\ \wedge \bigwedge_{j \neq i} \mathrm{maj}(y, C_j) &\leq \mathrm{maj}(x, C_j), \end{aligned}$$

   ersetze x durch y.
4. *Weiter bei 2.*

Auf den ersten Blick scheint der (1+1)* EA offensichtlich langsamer zu sein als der (1+1) EA: Alle Schritte des (1+1) EA, die den aktuellen φ-Wert um mehr als 1 erhöhen würden, werden vom (1+1)* EA automatisch verworfen.

Allerdings kann der Schuss auch nach hinten losgehen. Ein Schritt, in dem beispielsweise zwei Mehrheiten steigen, dafür aber fünf Mehrheiten sinken, wird bei entsprechenden Mehrheiten vom (1+1) EA akzeptiert, vom (1+1)* EA aber verworfen. Hier führt also das veränderte Akzeptanzkriterium dazu, dass der (1+1)* EA plötzlich im Vorteil ist. Es ist also kein einfacher Rückschluss auf die Random Walks auf der φ-Skala möglich.

Schließlich sei noch einmal daran erinnert, dass das Potenzial φ keine suffiziente Statistik ist und daher kein zuverlässiges Maß darstellt (siehe Abschnitt 3.2.2). Es ist gut möglich, dass beide Algorithmen unterschiedliche Verteilungen von Mehrheiten erzeugen; typische Suchpunkte beider Algorithmen mit gleichem Potenzial können sich stark voneinander unterscheiden.

Wir wollen nun in folgendem Theorem eine obere Schranke $O(n \log n)$ für die erwartete Optimierungszeit des (1+1)* EA zeigen. Anschließend zeigen wir in Abschnitt 3.2.5, dass sich diese obere Schranke auf den (1+1) EA übertragen lässt. Wir analysieren den (1+1) EA also nicht direkt, sondern mit Hilfe eines Umwegs über einen modifizierten Algorithmus, den (1+1)* EA.

Theorem 10. *Gegeben ein Cliquengraph $G = (V, E)$ mit $n := |V|$ Knoten und $1 \leq k \leq n/16$ unabhängigen Cliquen der Größe n/k. Sei T_{x^0} die Optimierungszeit des (1+1)* EA startend mit dem Suchpunkt x^0 und sei E_m die worst-case erwartete Optimierungszeit über alle Suchpunkte mit einem φ-Wert von m:*

$$E_m := \max_{x^0}\{\mathrm{E}(T_{x^0}) \mid \varphi(x^0) = m\}$$

Dann gilt

$$E_m < 150n \cdot \sum_{i=1}^{n-m} \frac{1}{i} = O(n \log n).$$

Beweis. Wir zeigen, dass die Drift in erfolgreichen Schritten durch eine positive Konstante 1/50 nach unten beschränkt ist. Der Rest des Beweises folgt dem Beweis von Theorem 9: Wir wenden das Drift-Argument aus Lemma 6 an, um die erwartete Zahl erfolgreicher Schritte, bis φ erhöht wird, nach oben abzuschätzen. Zusammen mit der erwarteten Wartezeit auf einen erfolgreichen Schritt folgt dann die Behauptung.

Genauer gesagt müssen wir folgendes zeigen. Sei x der aktuelle Suchpunkt, x' der erzeugte Nachkomme und $D_x := \varphi(x') - \varphi(x)$. Weiter sei A^* das Ereignis, dass ein Schritt des (1+1)* EA erfolgreich ist. Zu zeigen ist nun, dass die erwartete Drift in erfolgreichen Schritten durch eine positive Konstante beschränkt ist, selbst wenn D_x nur höchstens 1 ist. Da der (1+1)* EA aber nur eine Mehrheit um maximal 1 erhöht, gilt $D_x \leq 1$ trivialerweise und es gilt

$$\mathrm{E}(D_x \mid D_x \leq 1, A^*) = \mathrm{E}(D_x \mid A^*).$$

Um die positive Schranke für die erwartete Drift nachzuweisen, unterscheiden wir zwei Fälle.

Fall 1: $\varphi(x) \leq (49/50)n$

In einem erfolgreichen Schritt des (1+1)* EA wird die Mehrheit genau einer Clique um genau 1 erhöht. Wir unterschätzen die erwartete Drift, wenn wir annehmen, dass jedes weitere flippende Mehrheitsbit den φ-Wert um jeweils 1 senkt (flippende Minderheitsbits ignorieren wir). Dadurch erhalten wir wieder ein vereinfachtes Szenario, in dem wir negative Effekte auf die Fitness und kippende Mehrheiten ignorieren.

Wir zeigen, dass das vereinfachte Szenario pessimistisch ist, d. h. dass wir D_x im vereinfachten Szenario nicht überschätzen.

- So lange die Mehrheit in einer Clique C nicht kippt, können die dortigen Minderheitsbits die Mehrheit nur erhöhen, wenn sie flippen. Also unterschätzen wir D_x in diesem Fall, wenn wir flippende Minderheitsbits ignorieren.

- Falls die Mehrheit in einer Clique C kippt, rechnen wir im vereinfachten Szenario einen Verlust von mindestens $s := \mathrm{maj}(x, C) - \lceil n/(2k) \rceil$ an, da mindestens s Mehrheitsbits flippen müssen, um die Mehrheit zu kippen. Der wahre Verlust ist aber höchstens s, da die Mehrheit nicht unter die triviale untere Schranke $\lceil n/(2k) \rceil$ sinken kann.

Also ist unser Szenario in beiden Fällen pessimistisch.

Sei C die Clique, deren Mehrheit sich in einem erfolgreichen Schritt erhöht. Im pessimistischen Szenario können in x nur die $\varphi(x)$ Mehrheitsbits das Potenzial senken und flippende Mehrheitsbits sind trotz möglicher Fitnessverluste unabhängig vom Fitnessgewinn in C. Daher können wir die erwartete Veränderung des Potenzials nach unten abschätzen durch

$$\mathrm{E}(D_x \mid A^*) \geq 1 - \frac{1}{n} \cdot \varphi(x) = \frac{n - \varphi(x)}{n}.$$

Falls $\varphi(x) \leq (49/50)n$, ist die erwartete Drift in erfolgreichen Schritten

$$\mathrm{E}(D_x \mid A^*) \geq \frac{n - (49/50)n}{n} = \frac{1}{50}.$$

Fall 2: $\varphi(x) > (49/50)n$

Mit $A^*_C \subseteq A^*$ bezeichnen wir das Ereignis, dass der Schritt für den (1+1)* EA erfolgreich ist und die Mehrheit in C um 1 erhöht wird. Unser Ziel ist, die erwartete Drift unter der Bedingung A^* durch eine positive Konstante zu beschränken. Auf dem Weg dorthin zeigen wir eine ähnliche Aussage für Teilereignisse von A^*, dass nämlich die erwartete Drift für eine beliebige Clique C unter der Bedingung A^*_C durch eine positive Konstante r nach unten beschränkt ist: $\mathrm{E}(D_x \mid A^*_C) \geq r$. (Dies macht nur Sinn für Cliquen C, in denen sich die Mehrheit noch erhöhen kann, sprich: C ist noch nicht einheitlich gefärbt und somit $\mathrm{Prob}(A^*_C) > 0$.)

Wenn dies gezeigt ist, folgt daraus die gleiche untere Schranke für die bedingte erwartete Drift unter dem übergeordneten Ereignis A^*: Nach Definition bedingter Erwartungswerte lässt sich $\mathrm{E}(D_x \mid A^*)$, da $A^* = A^*_{C_1} \dot\cup \ldots \dot\cup A^*_{C_k}$, auf eine gewichtete Summe von Termen $\mathrm{E}(D_x \mid A^*_C)$ zurückführen, wobei wir die Summanden mit $\mathrm{Prob}(A^*_C) = 0$ von Anfang an auslassen:

$$\begin{aligned}
\mathrm{E}(D_x \mid A^*) &= \sum_{C \mid \mathrm{Prob}(A^*_C) > 0} \mathrm{Prob}(A^*_C) \cdot \mathrm{E}(D_x \mid A^*_C) \\
&\geq r \cdot \sum_{C \mid \mathrm{Prob}(A^*_C) > 0} \mathrm{Prob}(A^*_C) \\
&= r
\end{aligned}$$

Daher gilt die untere Schranke r auch für die erwartete Drift $\mathrm{E}(D_x \mid A^*)$.

Zu zeigen ist also nun $\mathrm{E}(D_x \mid A^*_C) \geq r$.

Wenn der φ-Wert groß ist, sind viele Mehrheiten groß. Konkret betragen die Mehrheiten in mindestens $(9/10)k$ Cliquen mindestens $(4/5)n/k$. Denn: Angenommen, die Zahl der Cliquen mit Mehrheiten von mindestens $(4/5)n/k$ wäre kleiner als $(9/10)k$, dann wäre die Zahl der Minderheitsbits in mehr als $(1/10)k$ Cliquen größer als $(1/5)n/k$, so dass wir insgesamt mehr als

$$\frac{1}{10} \cdot k \cdot \frac{1}{5} \cdot \frac{n}{k} = \frac{1}{50} \cdot n$$

Minderheitsbits erhalten. Dies steht im Widerspruch zu $\varphi(x) > (49/50)n$.

Mit M bezeichnen wir die Menge der Knoten, die zu Cliquen mit Mehrheiten von mindestens $(4/5)n/k$ gehören. Da die Anzahl dieser Cliquen mindestens $(9/10)k$ ist, gilt

$$|M| \geq \frac{9}{10} \cdot n.$$

Wie bereits angekündigt, wollen wir nun zeigen, dass nicht all zu viele große Mehrheiten sinken können, da ansonsten die Fitness zu stark absinkt und die gesamte Fitnessveränderung negativ wird, so dass der Nachkomme nicht akzeptiert wird.

Um die Fitnessveränderung durch eine Mutation zu berechnen, hilft uns die Separabilität der Fitnessfunktion f_G: Auf dem Graphen G ist die Fitness $f = f_G$ die Summe der einfarbigen Kanten in allen Cliquen. Wir können daher die Veränderung der Fitness in den einzelnen Cliquen unabhängig voneinander betrachten und anschließend alle Veränderungen addieren, um die gesamte Fitnessveränderung $f(x') - f(x)$ zu erhalten.

Wir partitionieren die Knoten in die disjunkten Mengen $C, M \setminus C$ und $V \setminus (M \cup C)$ und teilen die Drift in drei verschiedene Summanden auf:

$$\begin{aligned} D_x^C &:= \operatorname{maj}(x', C) - \operatorname{maj}(x, C), \\ D_x^M &:= \sum_{C_i | C_i \subseteq (M \setminus C)} \left(\operatorname{maj}(x', C_i) - \operatorname{maj}(x, C_i)\right), \\ D_x^V &:= \sum_{C_i | C_i \subseteq (V \setminus (M \cup C))} \left(\operatorname{maj}(x', C_i) - \operatorname{maj}(x, C_i)\right). \end{aligned}$$

Die gesamte Drift ergibt sich dann aus der Summe

$$D_x = D_x^C + D_x^M + D_x^V.$$

Mit der Linearität des Erwartungswertes gilt für die erwartete Drift

$$\mathrm{E}(D_x \mid A_C^*) = \mathrm{E}(D_x^C \mid A_C^*) + \mathrm{E}(D_x^M \mid A_C^*) + \mathrm{E}(D_x^V \mid A_C^*).$$

Von besonderem Interesse ist die Drift D_x^M in Cliquen mit hohen Mehrheiten, da sich hier sinkende Mehrheiten besonders stark auf die Fitness auswirken. Um wie viel kann D_x^M sinken, so dass der Nachkomme trotzdem akzeptiert wird?

Wenn das Ereignis A_C^* eintritt, wird die Mehrheit in der Clique C um genau $D_x^C = 1$ erhöht. Dies verbessert die Fitness in C um höchstens $n/k - 1$. Wenn hingegen die

Mehrheit in einer Clique $C_i \subseteq M \setminus C$ um 1 sinkt, beträgt die Fitnessveränderung in C_i

$$\frac{n}{k} - 2\,\mathrm{maj}(x, C_i) + 1 \;\leq\; \frac{n}{k} - 2 \cdot \frac{4}{5} \cdot \frac{n}{k} + 1 \;\leq\; -\frac{3}{5} \cdot \frac{n}{k} + 1.$$

Sinkt die Mehrheit in C_i um mindestens 2, ist die Fitnessveränderung in C_i durch $-(6/5)n/k + 4$ nach oben beschränkt. Wenn in einer anderen Clique C_j mit $C_i \neq C_j \subseteq M \setminus C$ ebenfalls die Mehrheit gesenkt wird, ist die Fitnessveränderung in $C_i \cup C_j$ durch $-(6/5)n/k + 2$ beschränkt.

Falls also $D_x^M \leq -2$ ist, ist die Fitnessveränderung in C_i bzw. in $C_i \cup C_j$ durch $-(6/5)n/k + 4$ nach oben beschränkt. Dieser Verlust kann höchstens durch den Fitnessgewinn in C kompensiert werden, da der $(1+1)^*$ EA nur eine Mehrheit um 1 erhöht und in keiner anderen Clique die Fitness steigt. Zusammen mit der Fitnesserhöhung in C erhalten wir eine obere Schranke für die gesamte Fitnessveränderung von

$$f(x') - f(x) \leq \frac{n}{k} - 1 - \frac{6}{5} \cdot \frac{n}{k} + 4 \leq -\frac{1}{5} \cdot \frac{n}{k} + 3.$$

Da nach Voraussetzung $k \leq n/16$, ist $n/k \geq 16$, und die Fitnessveränderung ist $f(x') - f(x) \leq -1/16$. Damit wird der Nachkomme nicht akzeptiert.

Wir haben damit gezeigt, dass die Mehrheiten von Cliquen in $M \setminus C$ zusammen nur um höchstens 1 sinken können. Die Bedingung $D_x^M \geq -1$ ist eine notwendige Voraussetzung für einen erfolgreichen Schritt und für D_x^M kommen nur die Werte -1 und 0 in Frage.

Mit dieser Erkenntnis über die *möglichen* Werte von D_x^M lässt sich jetzt der *erwartete* Wert von D_x^M abschätzen. Wenn wir die Definition des Erwartungswertes ausnutzen, gilt

$$\begin{aligned}
\mathrm{E}(D_x^M \mid A_C^*) &\overset{\text{Def.}}{=} \sum_{d=-1}^{0} \mathrm{Prob}(D_x^M = d \mid A_C^*) \cdot d \\
&= (-1) \cdot \mathrm{Prob}(D_x^M = -1 \mid A_C^*) \\
&= (-1) \cdot (1 - \mathrm{Prob}(D_x^M = 0 \mid A_C^*)) \\
&= -1 + \mathrm{Prob}(D_x^M = 0 \mid A_C^*).
\end{aligned}$$

Wie groß ist die bedingte Wahrscheinlichkeit $\mathrm{Prob}(D_x^M = 0 \mid A_C^*)$? Zunächst einmal gilt

$$\mathrm{Prob}(D_x^M = 0 \mid A_C^*) = \frac{\mathrm{Prob}(D_x^M = 0, A_C^*)}{\mathrm{Prob}(A_C^*)}.$$

Eine notwendige Bedingung für A_C^* ist, dass eines von $n/k - \mathrm{maj}(x, C)$ Minderheitsbits in C flippt oder dass die Mehrheit in C kippt und die neue Mehrheit um 1 größer ist als die alte Mehrheit. Aus dem Beweis von Lemma 5 wissen wir, dass wir folgende Abschätzung vornehmen können:

$$\mathrm{Prob}(A_C^*) \leq 2 \cdot \frac{n/k - \mathrm{maj}(x, C)}{n}.$$

Eine hinreichende Bedingung für das Ereignis $\{D_x^M = 0\} \cap A_C^*$ ist eine Operation, die eines von $n/k - \mathrm{maj}(x, C)$ Minderheitsbits in C flippt und keine anderen Bits flippt. Daher gilt

$$\mathrm{Prob}(D_x^M = 0, A_C^*) \geq \frac{n/k - \mathrm{maj}(x, C)}{en}$$

und zusammen genommen erhalten wir folgende Abschätzung:

$$\mathrm{Prob}(D_x^M = 0 \mid A_C^*) = \frac{\mathrm{Prob}(D_x^M = 0, A_C^*)}{\mathrm{Prob}(A_C^*)} \geq \frac{\frac{n/k - \mathrm{maj}(x,C)}{en}}{2 \,.\, \frac{n/k - \mathrm{maj}(x,C)}{n}} = \frac{1}{2e}.$$

Insgesamt haben wir also folgende untere Schranke für den bedingten Erwartungswert $\mathrm{E}(D_x^M \mid A_C^*)$ gezeigt:

$$\mathrm{E}(D_x^M \mid A_C^*) \quad \geq \quad -1 + \frac{1}{2e}.$$

Wir kennen nun Schranken für die bedingten Erwartungswerte von D_x^C und D_x^M. Wie groß ist die bedingte erwartete Drift $\mathrm{E}(D_x^V \mid A_C^*)$? Hier machen wir wieder die bereits aus Abschnitt 3.2.3 bekannte pessimistische Annahme, dass alle flippenden Bits aus M das Potenzial um jeweils 1 senken. Da $|M| \geq (9/10)n$, gilt $|V \setminus (M \cup C)| \leq n/10$ und die erwartete Drift lässt sich abschätzen durch

$$\mathrm{E}(D_x^V \mid A_C^*) \;\geq\; \frac{1}{n} \cdot |V \setminus (M \cup C)| \cdot (-1) \;\geq\; -\frac{1}{10}.$$

Insgesamt folgt eine erwartete Drift von

$$\begin{aligned} \mathrm{E}(D_x \mid A_C^*) &= \mathrm{E}(D_x^C \mid A_C^*) + \mathrm{E}(D_x^M \mid A_C^*) + \mathrm{E}(D_x^V \mid A_C^*) \\ &\geq 1 + \left(-1 + \frac{1}{2e}\right) + \left(-\frac{1}{10}\right) \\ &= \frac{1}{2e} - \frac{1}{10} > \frac{1}{12}. \end{aligned}$$

In beiden Fällen ist die erwartete Drift daher durch $1/50$ nach unten beschränkt.

Die erwartete Optimierungszeit lässt sich nun wie folgt abschätzen. Wir starten mit einem Suchpunkt x^0 mit einem φ-Wert von $m := \varphi(x^0)$ und zur Optimierung müssen maximal alle φ-Werte von m bis $n-1$ durch einen höheren Rekordwert übertroffen werden. Die worst-case erwartete Zahl erfolgreicher Schritte, bis φ erhöht wird, ist nach Lemma 6 höchstens 50. Die Wahrscheinlichkeit eines erfolgreichen Schrittes ist mindestens $(n - \varphi(x))/(en)$, da eine 1-Bit-Mutation eines von $n - \varphi(x)$ Minderheitsbits zu einem erfolgreichen Schritt führt. Wenn i der aktuelle Rekordwert für φ ist, ist $en/(n-i)$ eine obere Schranke für die erwartete Zeit auf einen erfolgreichen Schritt; diese obere Schranke gilt auch dann, wenn zwischendurch Suchpunkte mit kleinerem φ-Wert als i erreicht werden.

Die erwartete Optimierungszeit ist daher nach oben beschränkt durch

$$E_m \;\leq\; 1 + \sum_{i=m}^{n-1} \frac{en}{n-i} \cdot 50 \;<\; 150n \cdot \sum_{i=1}^{n-m} \frac{1}{i} \;=\; O(n \log n).$$

□

Die Voraussetzung $k \leq n/16$ ist notwendig, um eine Cliquengröße von mindestens 16 zu garantieren, so dass große Mehrheiten wirklich „groß“ sind. Nur so können wir noch garantieren, dass höchstens eine große Mehrheit sinken kann, ohne dass die Fitnessveränderung negativ wird; bei noch kleineren Cliquen verschwinden die Grenzen zwischen großen Mehrheiten und kleinen Mehrheiten endgültig.

Die Einschränkung auf $k \leq n/16$ schmerzt aber nicht, denn bei Graphen mit konstanter Cliquengröße garantiert uns der einfache Ansatz mit Fitnessschichten aus Theorem 8 direkt eine Schranke von $O(n/k \cdot n \log n) = O(n \log n)$ für den (1+1) EA.

3.2.5 Übertragung der Schranke auf den (1+1) EA

Für den (1+1)* EA haben wir unser Ziel erreicht, eine obere Schranke $O(n \log n)$ für die erwartete Optimierungszeit zu zeigen. Wie können wir nun eine gleichwertige Schranke für den (1+1) EA zeigen?

Eine Möglichkeit wäre, den (1+1)* EA und den (1+1) EA direkt miteinander zu vergleichen und nachzuweisen, dass der (1+1) EA schneller ist als der (1+1)* EA und deshalb eine kleinere (oder gleiche) erwartete Optimierungszeit hat. Dies erweist sich jedoch als schwierig. Zum einen ist es schwierig, die langfristigen Effekte der unterschiedlichen Akzeptanzkriterien abzusehen. Ein realistisches Ziel wäre, die Effekte eines einzigen Schritt für beide Algorithmen zu analysieren, wenn beide Algorithmen mit dem gleichen Suchpunkt starten. Über einen längeren Zeitraum ist eine solche Analyse jedoch kaum möglich, da beide Algorithmen unterschiedliche Verteilungen der Mehrheitsbits entwickeln können und daher unter Umständen völlig verschiedene Wege gehen.

Zum anderen wissen wir wenig über die „echte“ Laufzeit des (1+1)* EA. In Theorem 10 verwenden wir einige Abschätzungen, um die erwartete Optimierungszeit nach oben abzuschätzen (darunter auch das Drift-Argument aus Lemma 6). Da wir nicht wissen, wie scharf unsere Abschätzungen und Argumente sind, kann die echte erwartete Optimierungszeit deutlich kleiner sein als die obere Schranke. Wir sollten uns stets vor Augen führen, dass sich unser gesamtes, bisher erarbeitete Wissen über die erwartete Laufzeit des (1+1)* EA implizit auf die Abschätzungen und Drift-Argumente bezieht, die wir in Theorem 10 verwenden. Wenn wir unsere Überlegungen auf die echte Laufzeit des (1+1)* EA stützen, geben wir unser gesamtes Wissen aus der Hand.

Anstatt die erwartete Optimierungszeit des (1+1) EA durch die erwartete Optimierungszeit des (1+1)* EA zu beschränken, werden wir daher die erwartete Optimierungszeit des (1+1) EA durch die obere Schranke für den (1+1)* EA aus Theorem 10 beschränken. Dieser Weg ist direkter und er erlaubt uns, einige der Abschätzungen anzuwenden, die wir auch in Theorem 10 verwendet haben.

Genauer gesagt wenden wir eine Technik an, die bereits von Fischer und Wegener [7] angewendet wurde, um eine obere Schranke in kleinen Schritten von einem Algorithmus auf einen anderen Algorithmus zu übertragen.

Wir definieren eine Folge von Algorithmen, die „zwischen“ den beiden Algorithmen $(1+1)^*$ EA und (1+1) EA liegen: Für $t \geq 0$ sei A_t der Algorithmus, der in den ersten t Schritten den (1+1) EA simuliert und danach den $(1+1)^*$ EA. Für $t = 0$ erhalten wir den $(1+1)^*$ EA. Wenn wir t gegen unendlich gehen lassen, geht die Folge nach und nach in den (1+1) EA über. Unser Ziel ist, die obere Schranke für den $(1+1)^*$ EA $\widehat{=} A_0$ per Induktion auf alle folgenden Algorithmen A_t der Folge zu übertragen, so dass sie im Grenzübergang $t \to \infty$ auch für den (1+1) EA gilt.

Es wird sich zeigen, dass es leichter ist, die Folge der Algorithmen A_t zu betrachten, als die Schranke für den $(1+1)^*$ EA direkt auf den (1+1) EA zu übertragen. Zwei in der Folge benachbarte Algorithmen A_{t-1} und A_t unterscheiden sich nur im t-ten Schritt; hier arbeitet A_{t-1} wie der $(1+1)^*$ EA und A_t wie der (1+1) EA. Der Algorithmus A_t simuliert also gegenüber A_{t-1} in einem weiteren Schritt den (1+1) EA anstelle des $(1+1)^*$ EA. Da sich diese beiden Algorithmen damit sehr ähnlich sind, müssen wir nur die Auswirkungen eines einzigen Schritts betrachten, in dem A_{t-1} und A_t verschieden arbeiten. Damit ist es vergleichsweise leicht, eine obere Schranke für A_{t-1} auf den Algorithmus A_t zu übertragen.

Theorem 11. *Gegeben ein Cliquengraph $G = (V, E)$ mit $n := |V|$ Knoten und $1 \leq k \leq n/16$ unabhängigen Cliquen der Größe n/k. Dann gilt für alle $t \geq 0$: Die erwartete Zeit von A_t auf $f := \text{Ising}_G$ ist nach oben beschränkt durch die obere Schranke für den $(1+1)^*$ EA aus Theorem 10.*

Beweis. Wir zeigen die Behauptung durch Induktion über t.

Induktionsanfang: Für $t = 0$ entspricht A_0 dem $(1+1)^*$ EA, für den die obere Schranke aus Theorem 10 gezeigt wurde.

Induktionsschluss: Angenommen, die Behauptung gilt für A_{t-1}. Dann zeigen wir, dass die obere Schranke auch für A_t gilt.

Bei der Initialisierung und in den ersten $t-1$ Schritten sind beide Algorithmen identisch. Dies führt zum einen dazu, dass die bedingte erwartete Optimierungszeit für beide Algorithmen identisch ist, unter der Bedingung, dass ein Optimum innerhalb der ersten $t-1$ Schritte gefunden wird. Wir gehen also im Folgenden davon aus, dass noch kein Optimum erreicht wurde.

Zum anderen haben beide Algorithmen identische Verteilungen für die aktuellen Suchpunkte beider Algorithmen vor dem t-ten Schritt. Wir gehen daher im Folgenden davon aus, dass beide Algorithmen in Schritt t den gleichen aktuellen Suchpunkt haben.

Sei x ein beliebiger aktueller Suchpunkt beider Algorithmen in Schritt t. In Schritt t arbeitet A_{t-1} wie der $(1+1)^*$ EA und A_t wie der (1+1) EA; danach arbeiten beide Algorithmen wie der $(1+1)^*$ EA. Wir betrachten nun die erwartete restliche Optimierungszeit abhängig von x zum Zeitpunkt t und zeigen, dass wir für A_{t-1} und A_t die gleiche obere Schranke herleiten können. Dabei verwenden wir unser Wissen

über Abschätzungen, wie sie in den Rechnungen von Theorem 10 und dem Drift-Argument in Lemma 6 verwendet werden.

Sei $p_{x,y}$ die Wahrscheinlichkeit, dass der (1+1) EA ausgehend vom Suchpunkt x in der nächsten Generation y als aktuellen Suchpunkt hat. Analog sei $p^*_{x,y}$ die Wahrscheinlichkeit, dass der (1+1)* EA ausgehend vom Suchpunkt x in der nächsten Generation y als aktuellen Suchpunkt hat. Mit $T^t_x(\cdot)$ notieren wir die restliche Laufzeit des angegebenen Algorithmus ab Generation t, wenn x der aktuelle Suchpunkt in Generation t ist.

Wir betrachten nun den Übergang von x nach y und führen $\mathrm{E}(T^t_x(A_{t-1}))$ auf die bedingte erwartete restliche Optimierungszeit $\mathrm{E}(T^{t+1}_y(\cdot))$ ausgehend von y zurück. Da die Optimierungszeit nur vom aktuellen Suchpunkt, nicht aber von vergangenen Suchpunkten abhängt, ist $\mathrm{E}(T^{t+1}_y(\cdot))$ unabhängig von x. Zweitens ist es egal, ob wir die erwartete restliche Optimierungszeit $\mathrm{E}(T^{t+1}_y(\cdot))$ auf A_{t-1} oder auf A_t beziehen, da beide Algorithmen ab Generation $t+1$ gleich arbeiten.

Für die erwartete restliche Optimierungszeit von A_{t-1} gilt:

$$\begin{aligned}
\mathrm{E}(T^t_x(A_{t-1})) &= \sum_{y\in\{0,1\}^n} p^*_{x,y}\cdot \mathrm{E}(T^{t+1}_y(A_t)) \\
&\leq \sum_{y\in\{0,1\}^n} p^*_{x,y}\cdot E_{\varphi(y)} \\
&< \sum_{y\in\{0,1\}^n} p^*_{x,y}\cdot 150n\cdot \sum_{i=1}^{n-\varphi(y)} \frac{1}{i} \quad (*)
\end{aligned}$$

Dabei ist $E_{\varphi(y)}$ die in Theorem 10 definierte worst-case erwartete Optimierungszeit des (1+1)* EA unter allen Suchpunkten mit gleichem φ-Wert. Die zweite Ungleichung folgt aus Theorem 10. Beide Ungleichungen lassen sich anwenden, da sich $\mathrm{E}(T^{t+1}_y(A_t))$ auf A_t bezieht und A_t ab Generation $t+1$ wie der (1+1)* EA arbeitet.

Analog gilt für die erwartete restliche Optimierungszeit von A_t:

$$\begin{aligned}
\mathrm{E}(T^t_x(A_t)) &= \sum_{y\in\{0,1\}^n} p_{x,y}\cdot \mathrm{E}(T^{t+1}_y(A_t)) \\
&\leq \sum_{y\in\{0,1\}^n} p_{x,y}\cdot E_{\varphi(y)} \\
&< \sum_{y\in\{0,1\}^n} p_{x,y}\cdot 150n\cdot \sum_{i=1}^{n-\varphi(y)} \frac{1}{i} \quad (**)
\end{aligned}$$

Wichtig ist, zu erwähnen, dass die soeben verwendeten Ungleichungen im gleichen Zusammenhang auch beim Beweis der Schranke für $E_{\varphi(x)}$ verwendet werden. In Theorem 10 (und dem dort verwendeten Drift-Argument in Lemma 6) greifen wir implizit auf die gleichen Ungleichungen zurück, um die Laufzeit ab dem Suchpunkt x abzuschätzen. Wenn wir also hier ebenfalls diese Ungleichungen anwenden, verlieren wir nichts Wesentliches, denn auch aus dem abgeschätzten Ausdruck $(*)$ lässt sich die Schranke aus Theorem 10 herleiten.

Es kann sogar sein, dass die Ungleichungen $(*)$ und $(**)$ schärfer sind als die Ungleichungen, die für den gleichen Schritt in Theorem 10 verwendet werden: Falls vor y ein Suchpunkt z mit einem höheren Rekordwert $\varphi(z) = m > \varphi(y)$ erreicht wurde, wird in Theorem 10 die erwartete Wartezeit auf einen erfolgreichen Schritt nach oben durch $en \cdot \sum_{i=1}^{n-m} 1/i$ abgeschätzt statt durch den kleineren Wert $en \cdot \sum_{i=1}^{n-\varphi(y)} 1/i$ wie in $(*)$ und $(**)$.

Wenn wir nun zeigen können, dass der abgeschätzte Ausdruck $(**)$ für A_t nicht größer ist als $(*)$, können wir für A_t die gleiche obere Schranke herleiten wie für A_{t-1}. Damit wäre unser Ziel erreicht, die Schranke für A_{t-1} auf A_t zu übertragen.

Die zu zeigende Behauptung „$(**) \stackrel{!}{\leq} (*)$“ formen wir nun Schritt für Schritt um in einen leichter zu analysierenden Ausdruck. Die vorhin aufgestellten Formeln für $\mathrm{E}(T_x^t(\cdot))$ führen die erwarteten restlichen Optimierungszeiten ausgehend von x auf die bedingten erwarteten restlichen Optimierungszeiten ausgehend von allen möglichen Folgesuchpunkten y zurück. Dies war notwendig, um mit den aus dem Beweis von Theorem 10 bekannten Abschätzungen eine geschlossene Formel für $\mathrm{E}(T_x^t(\cdot))$ zu erhalten.

Für das weitere Vorgehen ist aber eine andere Schreibweise interessanter, die nur die Suchpunkte y berücksichtigt, die vom (1+1) EA akzeptiert werden, vom (1+1)* EA aber nicht. Daher werden wir nun die Behauptung „$(**) \stackrel{!}{\leq} (*)$“ Schritt für Schritt in eine solche Schreibweise überführen und von beiden Seiten die Terme subtrahieren, die für beide Algorithmen gleich sind.

Zunächst einmal teilen wir beide Seiten von „$(**) \stackrel{!}{\leq} (*)$“ durch $150n$:

$$\sum_{y \in \{0,1\}^n} p_{x,y} \cdot 150n \cdot \sum_{i=1}^{n-\varphi(y)} \frac{1}{i} \stackrel{!}{\leq} \sum_{y \in \{0,1\}^n} p^*_{x,y} \cdot 150n \cdot \sum_{i=1}^{n-\varphi(y)} \frac{1}{i}$$

$$\Leftrightarrow \sum_{y \in \{0,1\}^n} p_{x,y} \cdot \sum_{i=1}^{n-\varphi(y)} \frac{1}{i} \stackrel{!}{\leq} \sum_{y \in \{0,1\}^n} p^*_{x,y} \cdot \sum_{i=1}^{n-\varphi(y)} \frac{1}{i}$$

Sei Y die Menge aller Suchpunkte, die vom (1+1) EA akzeptiert werden, vom (1+1)* EA aber nicht. Falls $Y = \emptyset$, ist nichts zu zeigen, da sich beide Algorithmen gleich verhalten. Wir gehen daher im Folgenden davon aus, dass $Y \neq \emptyset$.

Beim Übergang zu einem Suchpunkt $y \neq x, y \notin Y$ verhalten sich beide Algorithmen gleich, da sowohl der Mutationsoperator als auch das Akzeptanzkriterium hier gleichermaßen arbeiten. Daher gilt $p_{x,y} = p^*_{x,y}$ für $y \neq x$ und $y \notin Y$ und wir können alle Summanden mit $y \notin Y \cup \{x\}$ von den Summen auf beiden Seiten subtrahieren. Wir erhalten:

$$\sum_{y \in Y \cup \{x\}} p_{x,y} \cdot \sum_{i=1}^{n-\varphi(y)} \frac{1}{i} \stackrel{!}{\leq} \sum_{y \in Y \cup \{x\}} p^*_{x,y} \cdot \sum_{i=1}^{n-\varphi(y)} \frac{1}{i}$$

Die Wahrscheinlichkeiten $p^*_{x,y}$ für $y \in Y$ sind allesamt 0, da y nach Definition von Y vom (1+1)* EA nicht akzeptiert wird. Wenn wir diese Summanden mit Wert 0

weglassen, erhalten wir die Ungleichung

$$\sum_{y \in Y \cup \{x\}} p_{x,y} \cdot \sum_{i=1}^{n-\varphi(y)} \frac{1}{i} \stackrel{!}{\leq} p^*_{x,x} \cdot \sum_{i=1}^{n-\varphi(x)} \frac{1}{i}$$

Wie groß ist $p^*_{x,x}$? Wir betrachten alle Zufallsentscheidungen, die bei der Mutation von x getroffen werden: Der (1+1)* EA erreicht den Suchpunkt x genau dann, wenn der (1+1) EA einen Suchpunkt $y \in Y$ erreicht (dieser wird vom (1+1)* EA verworfen) oder wenn der (1+1) EA beim Suchpunkt x verbleibt. Daher gilt

$$p^*_{x,x} = \sum_{y \in Y} p_{x,y} + p_{x,x}.$$

Wenn wir dies einsetzen und anschließend von beiden Seiten $p_{x,x} \cdot \sum_{i=1}^{n-\varphi(x)} 1/i$ subtrahieren, erhalten wir folgende Ungleichung (+):

$$\begin{aligned} \sum_{y \in Y \cup \{x\}} p_{x,y} \cdot \sum_{i=1}^{n-\varphi(y)} \frac{1}{i} &\stackrel{!}{\leq} \left(\sum_{y \in Y} p_{x,y} + p_{x,x} \right) \cdot \sum_{i=1}^{n-\varphi(x)} \frac{1}{i} \\ \Leftrightarrow \sum_{y \in Y} p_{x,y} \cdot \sum_{i=1}^{n-\varphi(y)} \frac{1}{i} &\stackrel{!}{\leq} \sum_{y \in Y} p_{x,y} \cdot \sum_{i=1}^{n-\varphi(x)} \frac{1}{i} \quad (+) \end{aligned}$$

Dies ist die angekündigte alternative Schreibweise für „$(**) \stackrel{!}{\leq} (*)$“, deren Semantik man sich leicht wie folgt erklären kann. Mit Wahrscheinlichkeit $p_{x,y}$ für $y \in Y$ geht der (1+1) EA nach y über und die erwartete restliche Optimierungszeit hängt von y ab. Mit der gleichen Wahrscheinlichkeit $p_{x,y}$ wird beim (1+1)* EA der gleiche Nachkomme y erzeugt und verworfen, so dass die restliche erwartete Optimierungszeit nun von x abhängt.

Zu zeigen ist also nun die Ungleichung (+), die für die folgenden Rechnungen ein handlicheres Format hat als die Ungleichung „$(**) \stackrel{!}{\leq} (*)$“. Trotzdem sind wir mit unseren Umformungen noch nicht am Ende; wir werden Ungleichung (+) nun in eine Form überführen, wie wir sie von Drift-Argumenten her kennen. Anschließend leiten wir aus der erwarteten Drift in Schritten, in denen sich der (1+1) EA und der (1+1)* EA unterscheiden, eine zweite Aussage ab, mit der wir Ungleichung (+) beweisen werden.

Wenn wir nun in (+) von beiden Seiten die rechte Seite subtrahieren, erhalten wir

$$\sum_{y \in Y} p_{x,y} \cdot \left(\sum_{i=1}^{n-\varphi(y)} \frac{1}{i} - \sum_{i=1}^{n-\varphi(x)} \frac{1}{i} \right) \stackrel{!}{\leq} 0.$$

Die Summe können wir in anderer Form schreiben und in eine Summe über d überführen, wie wir sie von Drift-Argumenten her kennen. Sei A das Ereignis, dass ein Schritt des (1+1) EA erfolgreich ist und A^* das Ereignis, dass ein Schritt des

(1+1)* EA erfolgreich ist. Wenn y der in einem Schritt erzeugte Nachkomme ist, ist das Ereignis $y \in Y$ äquivalent zum Ereignis $A \setminus A^*$. Sei $D_x := \varphi(y) - \varphi(x)$, dann gilt

$$\begin{aligned}
\sum_{y \in Y} p_{x,y} \cdot \left(\sum_{i=1}^{n-\varphi(y)} \frac{1}{i} - \sum_{i=1}^{n-\varphi(x)} \frac{1}{i} \right) &\overset{!}{\leq} 0 \\
\Leftrightarrow \sum_{d=-\infty}^{\infty} \sum_{y \in Y, D_x = d} p_{x,y} \cdot \left(\sum_{i=1}^{n-\varphi(y)} \frac{1}{i} - \sum_{i=1}^{n-\varphi(x)} \frac{1}{i} \right) &\overset{!}{\leq} 0 \\
\Leftrightarrow \sum_{d=-\infty}^{\infty} \mathrm{Prob}(D_x = d, A \setminus A^*) \cdot \left(\sum_{i=1}^{n-\varphi(x)-d} \frac{1}{i} - \sum_{i=1}^{n-\varphi(x)} \frac{1}{i} \right) &\overset{!}{\leq} 0
\end{aligned}$$

Wir definieren nun die Funktion $h_x \colon \mathbb{Z} \cap [n/2 - \varphi(x), n - \varphi(x)] \to \mathbb{R}$ als

$$h_x(d) := \sum_{i=1}^{n-\varphi(x)-d} \frac{1}{i} - \sum_{i=1}^{n-\varphi(x)} \frac{1}{i}$$

Damit lässt sich die Ungleichung (+) schreiben als

$$\sum_{d=-\infty}^{\infty} \mathrm{Prob}(D_x = d, A \setminus A^*) \cdot h_x(d) \overset{!}{\leq} 0 \quad (++)$$

Die Behauptung „$(**) \overset{!}{\leq} (*)$“ ist also äquivalent zu Ungleichung (++), die noch zu zeigen ist. Wir werden nun mit Hilfe der erwarteten Drift in Schritten, die nur vom (1+1) EA akzeptiert werden, eine ähnliche Ungleichung (+++) herleiten und dann mit Hilfe eines analytischen Arguments zeigen, dass sich (++) aus der Ungleichung (+++) herleiten lässt.

Alle Schritte, die für den (1+1)* EA erfolgreich sind, sind auch für den (1+1) EA erfolgreich. Was passiert in Schritten, die für den (1+1) EA erfolgreich sind, für den (1+1)* EA aber nicht? Nach Definition des (1+1)* EA muss in einem solchen Schritt eine Mehrheit um mehr als 1 wachsen oder es müssen mindestens zwei Mehrheiten wachsen.

Wie sieht in solchen Schritten die erwartete Drift aus? In einer oder zwei Cliquen wächst die Mehrheit insgesamt um mindestens 2. Selbst wenn wir davon ausgehen, dass jedes weitere flippende Bit das Potenzial senkt, ist die erwartete Zahl negativ flippender Bits kleiner als 1. (Hier wenden wir wieder das pessimistische Szenario aus Abschnitt 3.2.3 an.)

Also gilt

$$\mathrm{E}(D_x \mid A \setminus A^*) > 1.$$

(Hier verzichten wir auf die Bedingung $D_x \leq 1$, da wir die erwartete Drift in folgenden Rechnungen und nicht im Zusammenhang mit Lemma 6 anwenden wollen.)

Zweitens wissen wir, dass sich die bedingte erwartete Drift ausführlich schreiben lässt als

$$\begin{aligned} \mathrm{E}(D_x \mid A \setminus A^*) &\overset{\text{Def.}}{=} \sum_{d=-\infty}^{\infty} \mathrm{Prob}(D_x = d \mid A \setminus A^*) \cdot d \\ &= \frac{1}{\mathrm{Prob}(A \setminus A^*)} \cdot \sum_{d=-\infty}^{\infty} \mathrm{Prob}(D_x = d, A \setminus A^*) \cdot d \end{aligned}$$

Zusammen gilt

$$\begin{aligned} \frac{1}{\mathrm{Prob}(A \setminus A^*)} \cdot \sum_{d=-\infty}^{\infty} \mathrm{Prob}(D_x = d, A \setminus A^*) \cdot d &> 1 \\ \Leftrightarrow \sum_{d=-\infty}^{\infty} \mathrm{Prob}(D_x = d, A \setminus A^*) \cdot d &> \mathrm{Prob}(A \setminus A^*) \\ \Rightarrow \sum_{d=-\infty}^{\infty} \mathrm{Prob}(D_x = d, A \setminus A^*) \cdot d &\geq 0. \end{aligned}$$

Die gewünschte Ungleichung $(+++)$ erhalten wir nun, indem wir beide Seiten mit einer Konstanten $a < 0$ multiplizieren, deren Wert wir später festlegen werden.

$$\sum_{d=-\infty}^{\infty} \mathrm{Prob}(D_x = d, A \setminus A^*) \cdot a \cdot d \leq 0 \quad (+++).$$

Zum Vergleich führen wir nun die hergeleiteten Ungleichungen $(++)$ und $(+++)$ auf.

$$\begin{aligned} \sum_{d=-\infty}^{\infty} \mathrm{Prob}(D_x = d, A \setminus A^*) \cdot h_x(d) &\overset{!}{\leq} 0 \quad (++) \\ \sum_{d=-\infty}^{\infty} \mathrm{Prob}(D_x = d, A \setminus A^*) \cdot a \cdot d &\leq 0 \quad (+++) \end{aligned}$$

Wie man sieht, unterscheiden sich die Ungleichungen genau darin, dass die Wahrscheinlichkeiten in der Summe einmal mit der Funktion $h_x(d)$ und einmal mit der Funktion $a \cdot d$ multipliziert werden. Die Funktion $h_x(d)$ ist in ihrem Definitionsbereich streng monoton fallend, konkav und hat als Nullstelle den Wert $h_x(0) := 0$. Damit zeigen wir, dass wir $a < 0$ so wählen können, dass für alle d gilt: $h_x(d) \leq a \cdot d$.

Anschaulich kann man dies wie folgt begründen (siehe Abbildung 3.1): Wir stellen uns vor, $h_x(d)$ wäre nicht nur auf einer Teilmenge ganzer Zahlen definiert, sondern erweitern $h_x(d)$ gedanklich durch eine Ausgleichskurve zu einer stetigen, konkaven und monoton fallenden Funktion $\tilde{h}_x(d)$. Dann können wir a als Ableitung von $\tilde{h}_x$ an der Stelle $d = 0$ wählen, so dass $a \cdot d$ die Tangente zu $\tilde{h}_x$ an der Stelle $d = 0$ darstellt. Die Funktion $\tilde{h}_x$ liegt überall unter oder auf ihrer Tangente, so dass $\tilde{h}_x(d) \leq a \cdot d$ gilt. Da $\tilde{h}_x$ streng monoton fallend ist, ist die Steigung der Tangente negativ und $a < 0$.

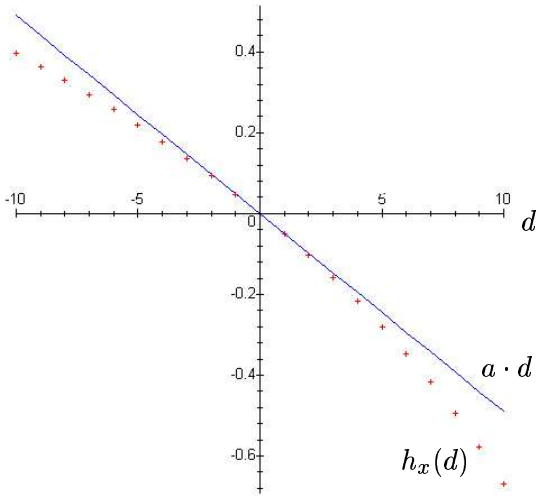


Abbildung 3.1: Die beiden Funktionen $h_x(d)$ und $a \cdot d$ in einem beispielhaften Szenario mit $n = 100$ und $\varphi(x) = 80$.

Da h_x nur auf einer Teilmenge der ganzen Zahlen definiert ist, bleiben uns bei der Wahl der Konstanten a im Gegensatz zu $\tilde{h}_x$ sogar noch mehr Freiheiten, da $h_x(d) \leq a \cdot d$ nur noch für den Definitionsbereich von h_x gelten muss. Nebenbei sei erwähnt, dass a nicht über alle Szenarien hinweg konstant gewählt werden kann, sondern dass a von n und $\varphi(x)$ abhängt.

Nun ist es leicht, aus $(+++)$ die Ungleichung $(++)$ zu folgern:

$$\begin{aligned} & \sum_{d=-\infty}^{\infty} \text{Prob}(D_x = d, A \setminus A^*) \cdot h_x(d) \\ \overset{h_x \leq a \cdot d}{\leq} & \sum_{d=-\infty}^{\infty} \text{Prob}(D_x = d, A \setminus A^*) \cdot a \cdot d \\ \overset{(+++)}{\leq} & \ 0 \end{aligned}$$

Damit haben wir gezeigt, dass die obere Schranke für die erwartete Optimierungszeit von A_{t-1} auch für A_t gilt. □

Für die erwarteten Optimierungszeiten aller Algorithmen der Folge $A_0, \ldots$ gilt also die obere Schranke $O(n \log n)$ und damit auch für den (1+1) EA. Dies gilt auf allen Cliquengraphen mit $1 \leq k \leq n/16$ gleich großen unabhängigen Cliquen. Auf Cliquengraphen mit mehr als $n/16$ gleich großen unabhängigen Cliquen folgt die Schranke $O(n \log n)$ aus Theorem 8.

Insgesamt haben wir also auf allen Cliquengraphen mit gleich großen unabhängigen Cliquen eine Schranke von $O(n \log n)$ für die erwartete Optimierungszeit des (1+1) EA gezeigt.

3.3 Die Analyse zweier verbundener Cliquen

In diesem Abschnitt wollen wir nun den (1+1) EA auf einem Cliquengraphen $G = (V, E)$ mit $n := |V| \geq 6$ Knoten analysieren. Dieser enthält zwei Cliquen C_1, C_2 der Größe $n/2$, die durch eine Brückenkante miteinander verbunden sind. (Welche Knoten Endpunkte der Brückenkante sind, spielt keine Rolle, da alle Knoten einer Clique austauschbar sind.)

Man kann sich leicht überlegen, dass alle Färbungen, bei denen beide Cliquen jeweils einheitlich gefärbt sind, lokale (oder globale) Optima sind, da jede 1-Bit-Mutation, die ein Bit einer einheitlich gefärbten Clique flippt, die Fitness senkt (vorausgesetzt, die Cliquen enthalten mindestens $n/2 \geq 3$ Knoten).

Es gibt vier Färbungen, die die zwei Cliquen jeweils einheitlich färben. Zwei von ihnen sind global optimal, da sei beide Cliquen mit der gleichen Farbe färben. Die zwei Färbungen, die beide Cliquen einheitlich, aber mit verschiedenen Farben färben, sind rein lokale, nicht globale Optima, da die Brückenkante zweifarbig ist. Die Fitnesslandschaft der Ising-Funktion auf G ist daher multimodal und enthält vier Optima, zwei rein lokale und zwei globale Optima, die symmetrisch im Suchraum verteilt sind und zueinander einen Abstand von $n/2$ oder n haben.

Wenn die Brückenkante nicht wäre, würde der (1+1) EA einen Suchpunkt, in dem beide Cliquen mit verschiedenen Farben gefärbt sind, mit Wahrscheinlichkeit 1/2 erreichen. Die Brückenkante gibt leichte Hinweise dahin gehend, beide Cliquen mit der gleichen Farbe zu färben, so dass die Wahrscheinlichkeit, einen Suchpunkt mit verschieden gefärbten Cliquen zu erreichen, dadurch verändert wird. Unter allen Möglichkeiten, die zwei Cliquen miteinander zu verbinden, gibt eine einzelne Brückenkante jedoch die wenigsten Hinweise darauf, beide Cliquen einheitlich zu färben. Daher ist es plausibel, dass die Wahrscheinlichkeit, eine Färbung mit zwei verschieden gefärbten Cliquen zu erreichen, nur leicht absinkt, wenn wir eine einzelne Brückenkante einfügen.

Konkret wollen wir im Folgenden zeigen: Die Wahrscheinlichkeit, dass der (1+1) EA lokales Optimum mit verschieden gefärbten Cliquen erreicht, beträgt immer noch mindestens $1/2 - o(1)$.

Beweisideen

Der Beweis folgt den Ideen von Wegener und Witt [22], die eine vergleichbare Fitnessfunktion analysieren und dazu die in Abschnitt 1.5.4 vorgestellte Methode des typischen Laufs verwenden. In typischen Läufen entscheidet sich in einer ersten, kurzen Phase, in welche Richtung der Optimierungsprozess läuft, d. h. ob sich der aktuelle Suchpunkt des (1+1) EA einem lokalen oder einem globalen Optimum annähert.

Wenn einmal eine solche Entscheidung getroffen ist und der aktuelle Suchpunkt einen genügend großen Abstand von den anderen Optima hat, sind aufgrund der vorliegenden Fitnesslandschaft große Sprünge nötig, um diese Entscheidung rückgängig zu machen und sich einem anderen Optimum anzunähern. Daher erreicht der (1+1) EA

in einer zweiten Phase mit hoher Wahrscheinlichkeit das Optimum, dem er nach der ersten Phase am nächsten war.

Die Ising-Funktion auf einem Graphen G mit zwei verbundenen Cliquen lässt sich auf eine ähnliche Art analysieren. In einer ersten Phase wollen wir sicher stellen, dass sich ausreichend große Mehrheiten heraus kristallisieren; Phase 1 wird daher dann beendet, wenn beide Mehrheiten ausreichend groß sind. In einer zweiten Phase stellen wir sicher, dass sich diese beiden Mehrheiten auch durchsetzen, d. h. dass beide Cliquen in Phase 2 einheitlich mit den Mehrheitsfarben gefärbt werden, die sich in Phase 1 heraus kristallisiert haben.

In beiden Phasen eines typischen Laufs spielen dabei die Mehrheiten $\text{maj}(x, C_1)$ und $\text{maj}(x, C_2)$ eine große Rolle, da sie den Abstand zu alternativen Optima angeben: Je höher eine Mehrheit $\text{maj}(x, C)$ mit Mehrheitsfarbe $c \in \{0, 1\}$, um so größer ist der Abstand zu dem Optimum, bei dem C einheitlich mit der anderen Farbe $1 - c$ gefärbt ist. Ein großer Abstand zu alternativen Optima ist in Phase 2 wichtig, da wir dort sicher stellen wollen, dass das einmal angepeilte Optimum tatsächlich erreicht wird. Falls Mehrheiten kippen, bedeutet dies, dass der (1+1) EA sich zu einem alternativen Optimum orientiert und der Lauf nicht mehr typisch ist.

Daher wollen wir im Folgenden möglichst hohe Werte für beide Mehrheiten sicher stellen, um die Wahrscheinlichkeit kippender Mehrheiten gering zu halten.

Experimente lassen darauf schließen, dass typischerweise eine Clique deutlich „in Führung liegt“, d. h. eine erkennbar größere Mehrheit besitzt als die andere Clique. Allgemein bezeichnen wir die Clique mit größerer Mehrheit als die *führende Clique* und die Clique mit kleinerer Mehrheit als *zurück liegende Clique*; falls beide Mehrheiten gleich groß sind, wird eine beliebige Clique als zurück liegende Clique bezeichnet, die andere als führende Clique.

Wir zeigen zuerst drei Aussagen, die wir anschließend verwenden werden, um die angestrebte Aussage über die Wahrscheinlichkeit, ein lokales Optimum mit verschieden gefärbten Cliquen zu erreichen, zu zeigen.

Zum einen gibt es eine einfache untere Schranke für die Mehrheit der jeweils führenden Clique. Dies ist plausibel, denn wenn beide Mehrheiten zu klein werden, sinkt die Fitness.

Zweitens betrachten wir die jeweils zurück liegende Clique. Da dort am meisten Minderheitsbits zu finden sind, ist die bedingte Wahrscheinlichkeit hoch, dass eine günstige 1-Bit-Mutation ein Minderheitsbit aus der zurück liegenden Clique trifft. Wir können unter bestimmten Bedingungen zeigen, dass die zurück liegende Clique in erfolgreichen Schritten eine erwartete Drift von $\Omega(1)$ hat.

Dadurch wird es unwahrscheinlich, dass die Mehrheit der zurück liegenden Clique sehr kleine Werte annimmt: Wenn wir große und damit sehr unwahrscheinliche Sprünge ausschließen, muss die Mehrheit in vielen kleinen Sprüngen „gegen den Strom schwimmen“, um einen kleinen Wert zu erreichen. Wir werden in einer dritten Aussage zeigen, dass dies sehr unwahrscheinlich ist.

3.3.1 Lemma zur Mehrheit der führenden Clique

Zuerst zeigen wir für unseren Graphen mit zwei verbundenen Cliquen eine untere Schranke für die Mehrheit der jeweils führenden Clique. Gegeben zwei Suchpunkte x, y mit $f(y) \geq f(x)$, dann wollen wir zeigen, dass die maximale Mehrheit in y nach unten beschränkt ist durch $\varphi(x)/2 - 1$. Der Term „-1“ entsteht dadurch, dass wir den winzigen, aber dennoch vorhandenen Einfluss der Brückenkante berücksichtigen.

Das folgende Lemma beweisen wir durch Kontraposition und nutzen dazu folgende einfache Beobachtung aus: Wenn die maximale Mehrheit in y kleiner ist als $\varphi(x)/2 - 1$, sind beide Mehrheiten kleiner als $\varphi(x)/2 - 1$ und man kann leicht zeigen, dass die Fitness von y dann kleiner ist als die Fitness von x.

Lemma 7. *Gegeben ein Cliquengraph $G = (V, E)$ mit $n := |V|$ Knoten und zwei disjunkten Cliquen C_1, C_2 der Größe $n/2$, die durch genau eine Brückenkante verbunden sind. Sei $f := \mathrm{Ising}_G$ und $x, y \in \{0,1\}^n$ Suchpunkte mit $f(y) \geq f(x)$, dann gilt*

$$\max_i \{\mathrm{maj}(y, C_i)\} \geq \frac{\varphi(x)}{2} - 1.$$

Beweis. Die Mehrheiten der einzelnen Cliquen gehen quadratisch in die Fitness ein. Sei $B(c) := \binom{c}{2} + \binom{n/2-c}{2}$ für $n/4 \leq c \leq n/2$, dann entspricht $B(c)$ dem Beitrag einer Clique C mit $\mathrm{maj}(x, C) = c$ zur Fitness und es ist

$$B(c) = \frac{c(c-1)}{2} + \frac{(n/2-c)(n/2-c-1)}{2} = \frac{n^2}{8} - \frac{n}{4} - \frac{c \cdot n}{2} + c^2.$$

Weiter sind die Werte $B(c)$ streng monoton in c:

$$\begin{aligned} B(c+1) &> B(c) \\ \Leftrightarrow -\frac{(c+1)\cdot n}{2} + (c+1)^2 &> -\frac{c \cdot n}{2} + c^2 \\ \Leftrightarrow -\frac{n}{2} + 2c + 1 &> 0 \\ \Leftrightarrow 2c + 1 &> \frac{n}{2} \end{aligned}$$

Die letzte Ungleichung gilt, da trivialerweise $c \geq n/4$.

Bei festem Potenzial $\varphi(x)$ ist die Fitness von x minimal, wenn die Brückenkante zweifarbig ist und alle Mehrheiten annähernd gleich groß sind, d. h. wenn beide Cliquen Mehrheiten von $\lfloor \varphi(x)/2 \rfloor$ und $\lceil \varphi(x)/2 \rceil$ haben. Letzteres lässt sich leicht einsehen: Wenn beide Mehrheiten betraglich eine Differenz von mindestens 2 haben, o. B. d. A. $\mathrm{maj}(x, C_1) - \mathrm{maj}(x, C_2) \leq -2$, kann eine 2-Bit-Mutation, die die größere Mehrheit $\mathrm{maj}(x, C_2)$ senkt und die kleinere Mehrheit $\mathrm{maj}(x, C_1)$ erhöht, die Fitness senken. Angenommen, x' entsteht aus x durch eine solche 2-Bit-Mutation, dann gilt mit unseren Vorüberlegungen aus Abschnitt 3.1.1

$$\begin{aligned} f(x') - f(x) &= (2\,\mathrm{maj}(x, C_1) - n/k + 1) + (n/k - 2\,\mathrm{maj}(x, C_2) + 1) \\ &= 2(\mathrm{maj}(x, C_1) - \mathrm{maj}(x, C_2)) + 2 \\ &\leq -2. \end{aligned}$$

So lange die Mehrheiten nicht annähernd gleich groß sind, lässt sich also noch ein anderer Suchpunkt mit kleinerer Fitness finden.

Also gilt

$$f(x) \geq B(\lfloor \varphi(x)/2 \rfloor) + B(\lceil \varphi(x)/2 \rceil).$$

Wir zeigen die Behauptung $\max_i\{\mathrm{maj}(y, C_i)\} \geq \varphi(x)/2 - 1$ durch Kontraposition. Angenommen, $\max_i\{\mathrm{maj}(y, C_i)\} < \varphi(x)/2 - 1$. Dann ist die Fitness von y maximal, wenn die Brückenkante einfarbig ist und alle Mehrheiten den Wert $c := \lceil \varphi(x)/2 \rceil - 2$ annehmen, den größten ganzzahligen Wert kleiner als $\varphi(x)/2 - 1$. Also gilt

$$f(y) \leq 2 \cdot B(c) + 1.$$

Da $B(c) < B(c+1)$ und B nur ganzzahlige Werte annimmt, gilt $B(c) \leq B(c+1) - 1$ und $2 \cdot B(c) + 1 \leq 2 \cdot B(c+1) - 1 < 2 \cdot B(c+1)$. Zusammen mit $c + 1 \leq \lfloor \varphi(x)/2 \rfloor$ gilt

$$f(y) \;<\; 2 \cdot B(c+1) \;\leq\; B(\lfloor \varphi(x)/2 \rfloor) + B(\lceil \varphi(x)/2 \rceil) \;\leq\; f(x),$$

Widerspruch zu $f(y) \geq f(x)$. □

3.3.2 Eine untere Schranke für die Drift der zurück liegenden Clique

Lemma 7 wird uns helfen, für die führende Clique ausreichend große Mehrheiten zu garantieren. Daher wenden wir uns nun der zurück liegenden Clique zu. In relevanten Schritten wird mindestens eine Mehrheit verändert. Fast alle relevanten Schritte sind auch erfolgreich, d. h. eine der beiden Mehrheiten wird erhöht. Es gibt nur wenige „pathologische“ Schritte, die relevant, aber nicht erfolgreich sind (d. h. beide Mehrheiten können nur sinken); um diese werden wir uns aber erst später kümmern. Vorerst konzentrieren wir uns auf die Effekte in erfolgreichen Schritten.

Wir betrachten im Folgenden einen einzelnen erfolgreichen Schritt und seine Auswirkungen auf die Mehrheit der zurück liegenden Clique. Dabei ist es uns egal, ob die zurück liegende Clique auch nach dem Schritt noch zurück liegend ist oder ob die Cliquen ihre Rollen tauschen.

Vergleicht man die Auswirkungen eines erfolgreichen Schritts auf die Mehrheiten der führenden Clique und der zurück liegenden Clique miteinander, kann man zwei widerstreitende Effekte ausmachen.

- Bitflips in der führenden Clique haben im Allgemeinen ein höheres Gewicht als Bitflips in der zurück liegenden Clique.

  Beispielsweise wird eine 2-Bit-Mutation, die die Mehrheit der führenden Clique erhöht und die Mehrheit der zurück liegenden Clique senkt, stets akzeptiert, da die Clique mit größerer Mehrheit ein höheres „Gewicht“ hat. Anders herum wird eine 2-Bit-Mutation, die die Mehrheit der führenden Clique senkt und die Mehrheit der zurück liegenden Clique erhöht, nicht akzeptiert, falls die Differenz der Mehrheiten betraglich mindestens 2 beträgt (dies haben wir im Beweis von Lemma 7 gezeigt).

- Dem niedrigeren „Gewicht“ der zurück liegenden Clique steht entgegen, dass es in der zurück liegenden Clique mehr Minderheitsbits gibt.

  Für den Mutationsoperator ist es leichter, einen Nachkommen zu erzeugen, in dem die Mehrheit der zurück liegenden Clique erhöht wird, als einen Nachkommen zu erzeugen, indem die Mehrheit der führenden Clique erhöht wird. Dies gilt wohlgemerkt nur für die rein „mechanische“ Mutation vor der Selektion, bei der die Einflüsse beider Cliquen auf die Fitness ins Spiel kommen.

Insgesamt wird die zurück liegende Clique durch den Mutationsschritt bevorteilt, durch die unterschiedlichen Einflüsse auf die Fitness aber benachteilt.

Nun wollen wir zeigen, dass die zurück liegende Clique in erfolgreichen Schritten eine erwartete Drift von $\Omega(1)$ hat, falls die Mehrheit der zurück liegenden Clique klein ist. Die Diskussion über die widerstreitenden Effekte soll verdeutlichen, dass diese Aufgabe nicht so einfach ist, wie sie zunächst aussehen mag.

Der folgende Beweis enthält eine Rechnung, die aus einer langen Kette aneinander gereihter Abschätzungen besteht. Diese Abschätzungen sind entweder einfach oder sie werden entsprechend motiviert und erklärt. An manchen Stellen verweisen wir auf die Erkenntnisse von Lemma 5, in dessen Beweis wir Aussagen darüber machen, mit welcher Wahrscheinlichkeit die Mehrheit einer Clique erhöht wird. Es sei daran erinnert, dass eine Mehrheit entweder dadurch erhöht werden kann, dass in der entsprechenden Clique mehr Minderheitsbits flippen als Mehrheitsbits, oder dadurch, dass die aktuelle Mehrheit der Clique kippt und die neue Mehrheit größer ist also die alte Mehrheit.

Lemma 8. *Gegeben ein Cliquengraph $G = (V, E)$ mit $n := |V|$ Knoten und zwei Cliquen C_1, C_2 der Größe $n/2$, die durch eine Brückenkante verbunden sind, und ein Suchpunkt $x \in \{0,1\}^n$. Für eine Clique, o. B. d. A. für C_1, seien die Ungleichungen $\mathrm{maj}(x, C_1) \leq \mathrm{maj}(x, C_2)$ und $\mathrm{maj}(x, C_1) \leq (13/50)n$ erfüllt.*

Sei A das Ereignis, dass ein Schritt des (1+1) EA auf $f := \mathrm{Ising}_G$ erfolgreich ist und x' der Suchpunkt, der im Mutationsschritt aus x erzeugt wird (vor der Selektion). Weiter sei $D_i := \mathrm{maj}(x', C_i) - \mathrm{maj}(x, C_i)$ für $i \in \{1, 2\}$.

Dann gilt für ausreichend große n:

$$\mathrm{E}(D_1 \mid A, D_1 \leq 1) \geq \frac{3}{200}.$$

Beweis. Zuerst formen wir den bedingten Erwartungswert $\mathrm{E}(D_1 \mid A, D_1 \leq 1)$ um:

$$\begin{aligned}
\mathrm{E}(D_1 \mid A, D_1 \leq 1) &\overset{\text{Def.}}{=} \sum_{d=-\infty}^{\infty} \mathrm{Prob}(D_1 = d \mid A, D_1 \leq 1) \cdot d \\
&= \sum_{d=-\infty}^{\infty} \frac{\mathrm{Prob}(D_1 = d, A, D_1 \leq 1)}{\mathrm{Prob}(A, D_1 \leq 1)} \cdot d \\
&= \frac{1}{\mathrm{Prob}(A, D_1 \leq 1)} \sum_{d=-\infty}^{1} \mathrm{Prob}(D_1 = d, A) \cdot d
\end{aligned}$$

Der Summand für $d = 0$ ist 0. Nur der Summand mit $d = 1$ ist positiv, daher behandeln wir ihn getrennt von den Summanden mit $d < 0$. Mit der obigen Gleichung können wir nun eine Rechnung aufstellen, die wir im Folgenden Schritt für Schritt weiter auflösen werden. Zunächst multiplizieren wir beide Seiten der Gleichung mit $\text{Prob}(A, D_1 \leq 1)$ und erhalten

$$\begin{aligned} & \text{E}(D_1 \mid A, D_1 \leq 1) \cdot \text{Prob}(A, D_1 \leq 1) \\ = & \sum_{d=-\infty}^{1} \text{Prob}(D_1 = d, A) \cdot d \\ = & \sum_{d=-\infty}^{-1} \text{Prob}(D_1 = d, A) \cdot d + \text{Prob}(D_1 = 1, A) \quad (*) \end{aligned}$$

Zunächst schätzen wir $\text{Prob}(D_1 = d, A)$ für $d < 0$ ab, wenn also die Mehrheit von C_1 in einem erfolgreichen Schritt sinkt. Dies kann nur dann geschehen, wenn in der anderen Clique C_2 die Mehrheit steigt, da ansonsten der Schritt nicht erfolgreich ist. Für einen erfolgreichen Schritt, der die Mehrheit in C_1 um $d < 0$ verändert, sind also notwendige Bedingungen, dass die Mehrheit in C_2 steigt, die Mehrheit in C_1 um d verändert wird und der Schritt trotzdem erfolgreich ist. Für $d < 0$ gilt

$$\{D_1 = d\} \cap A \ \subseteq \ \{D_2 > 0\} \cap \{D_1 = d\} \cap A \ \subseteq \ \{D_2 > 0\} \cap \{D_1 = d\}.$$

In der zweiten Inklusion verzichten wir auf die Bedingung A eines erfolgreichen Schritts. Dies kann man wie folgt interpretieren. Wir kreieren ein pessimistisches Szenario, indem wir davon ausgehen, dass jede Erhöhung der Mehrheit von C_2 alle negativen Effekte von Bitflips in C_1 dominiert und somit ein Schritt, der die Mehrheit in C_1 senkt, trotzdem immer erfolgreich ist. Durch diese pessimistische Annahme wird die Wahrscheinlichkeit von Schritten, die die Mehrheit in C_1 senken, höchstens größer, wie die Inklusion zeigt.

Die Folge dieser Vereinfachung ist, dass die Ereignisse $D_2 > 0$ und $D_1 = d$ nun nicht mehr an die Bedingung A eines erfolgreichen Schrittes gekoppelt sind. Da x' der Mutant vor der Selektion ist, betrachten wir bei den Veränderungen beider Mehrheiten nur die „mechanischen“ Auswirkungen des Mutationsschritts und abstrahieren von den Effekten auf die Fitness.

Da der Mutationsoperator alle Bits unabhängig behandelt, sind die beiden Ereignisse $D_2 > 0$ und $D_1 = d$ unabhängig voneinander. Für $d < 0$ gilt daher

$$\text{Prob}(D_1 = d, A) \leq \text{Prob}(D_1 = d) \cdot \text{Prob}(D_2 > 0).$$

Wenn wir dies in $(*)$ einsetzen, erhalten wir

$$\begin{aligned} & \text{E}(D_1 \mid A, D_1 \leq 1) \cdot \text{Prob}(A, D_1 \leq 1) \\ \geq & \sum_{d=-\infty}^{-1} \text{Prob}(D_1 = d) \cdot \text{Prob}(D_2 > 0) \cdot d + \text{Prob}(D_1 = 1, A) \\ = & \ \text{Prob}(D_2 > 0) \cdot \sum_{d=-\infty}^{-1} \text{Prob}(D_1 = d) \cdot d + \text{Prob}(D_1 = 1, A) \quad (**) \end{aligned}$$

Das Ereignis A eines erfolgreichen Schritts taucht nur noch in der Wahrscheinlichkeit $\text{Prob}(D_1 = 1, A)$ auf. Diese schätzen wir nun nach unten ab. Wenn eine Mutation die Mehrheit in C_1 um $D_1 = 1$ ändert und kein Bit aus der zweiten Clique flippt, ist der Schritt erfolgreich. Also gilt

$$\text{Prob}(D_1 = 1, A) \geq \text{Prob}(D_1 = 1) \cdot \left(1 - \frac{1}{n}\right)^{n/2}.$$

Wir setzen die Rechnung an $(**)$ fort und erhalten

$$\begin{aligned} & \text{E}(D_1 \mid A, D_1 \leq 1) \cdot \text{Prob}(A, D_1 \leq 1) \\ \geq\ & \text{Prob}(D_2 > 0) \cdot \sum_{d=-\infty}^{-1} \text{Prob}(D_1 = d) \cdot d + \text{Prob}(D_1 = 1) \cdot \left(1 - \frac{1}{n}\right)^{n/2} \qquad (***) \end{aligned}$$

In diesem Ausdruck haben wir endgültig von allen Effekten der Fitness abstrahiert und kümmern uns nun nur noch um die Wahrscheinlichkeiten „mechanischer“ Bitflips.

Als nächstes wollen wir die Summe über d abschätzen. Eine notwendige Bedingung dafür, dass D_1 um $d < 0$ verändert wird, ist, dass in C_1 mindestens $-d$ Mehrheitsbits flippen. (Dies ist keine hinreichende Bedingung, da auch Minderheitsbits flippen können und die Effekte flippender Mehrheitsbits neutralisieren können. Zweitens ist es auch möglich, dass die Mehrheit kippt und die aktuelle Minderheitsfarbe genau die Mehrheit von $\text{maj}(x, C_1) + d$ erreicht.) Da es $\text{maj}(x, C_1)$ Mehrheitsbits in C_1 gibt, gilt

$$\text{Prob}(D_1 = d) \leq \binom{\text{maj}(x, C_1)}{-d} \cdot n^d \leq \left(\frac{\text{maj}(x, C_1)}{n}\right)^{-d} \cdot \frac{1}{(-d)!}.$$

Zusammen mit der Voraussetzung $\text{maj}(x, C_1) \leq (13/50)n$ gilt

$$\text{Prob}(D_1 = d) \leq \left(\frac{13}{50}\right)^{-d} \cdot \frac{1}{(-d)!} \leq \left(\frac{13}{50}\right)^{-d} \cdot \frac{1}{-d}.$$

Für die Summe gilt dann

$$\sum_{d=-\infty}^{-1} \text{Prob}(D_1 = d) \cdot d \geq -\sum_{d=-\infty}^{-1} \left(\frac{13}{50}\right)^{-d} = -\sum_{d=1}^{\infty} \left(\frac{13}{50}\right)^{d} = -\frac{13}{37}.$$

Wenn wir dies in $(***)$ einsetzen, erhalten wir

$$\begin{aligned} & \text{E}(D_1 \mid A, D_1 \leq 1, M_m) \cdot \text{Prob}(A, D_1 \leq 1, M_m) \\ \geq\ & \text{Prob}(D_2 > 0) \cdot \left(-\frac{13}{37}\right) + \text{Prob}(D_1 = 1) \cdot \left(1 - \frac{1}{n}\right)^{n/2} \qquad (****) \end{aligned}$$

Als nächstes wollen wir die Wahrscheinlichkeiten für $D_2 > 0$ und $D_1 = 1$ auflösen. Der gewohnte Weg, dies zu tun, wäre, eine obere Schranke für $\text{Prob}(D_2 > 0)$ einzusetzen und eine untere Schranke für $\text{Prob}(D_1 = 1)$. Als Schranken könnte man

ähnliche untere und obere Schranken wie im Beweis von Lemma 5 entwickeln. Allerdings sind diese Schranken hier zu ungenau, da sie um einen Faktor $2e$ auseinander liegen und somit der negative Anteil an der Formel gegenüber dem positiven Anteil mit Faktor $2e$ überbewertet würde.

Wir gehen daher anders vor und vergleichen die Wahrscheinlichkeiten beider Ereignisse direkt miteinander. Dazu zeigen wir, dass

$$\text{Prob}(D_2 > 0) \leq \text{Prob}(D_1 = 1) + \frac{1}{16}.$$

Zuerst gilt aufgrund der Voraussetzung $\text{maj}(x, C_1) \leq \text{maj}(x, C_2)$, dass es für den Mutationsoperator leichter ist, in C_1 die Mehrheit zu erhöhen als in C_2 (bzw. genau so schwer). Mit den Rechnungen in Lemma 5 folgt: Sowohl in dem Fall, dass mehr Minderheitsbits als Mehrheitsbits flippen, als auch in dem Fall, dass Mehrheiten kippen und dadurch die Mehrheit steigt, ist die Wahrscheinlichkeit des Mutationsoperators, die Mehrheit von C_1 zu erhöhen, größer als (genau so groß wie) die Wahrscheinlichkeit, die Mehrheit von C_2 zu erhöhen.

Also gilt $\text{Prob}(D_2 > 0) \leq \text{Prob}(D_1 > 0)$ und

$$\text{Prob}(D_2 > 0) \leq \text{Prob}(D_1 > 0) = \text{Prob}(D_1 = 1) + \text{Prob}(D_1 > 1).$$

Eine notwendige Bedingung für das Ereignis $D_1 > 1$ ist, dass mindestens zwei Minderheitsbits flippen (Fall 1) oder dass die Mehrheit kippt und dabei mindestens $2\,\text{maj}(x, C_1) - n/2 + 2$ Mehrheitsbits flippen (Fall 2). Die Wahrscheinlichkeit für Fall 2 ist nicht größer als die Wahrscheinlichkeit für Fall 1, dass mindestens zwei Minderheitsbits flippen. (Für $\text{maj}(x, C_1) = n/4$ sind die Wahrscheinlichkeiten gleich groß, bei steigender Mehrheit fällt die Wahrscheinlichkeit für Fall 2 schneller als die für Fall 1, siehe Beweis von Lemma 5.) Also gilt

$$\begin{aligned}
\text{Prob}(D_1 > 1) &\leq 2 \cdot \binom{n/2 - \text{maj}(x, C_1)}{2} \cdot \frac{1}{n^2} \\
&\leq \left(\frac{n/2 - \text{maj}(x, C_1)}{n}\right)^2 \\
&\leq \left(\frac{n/4}{n}\right)^2 = \frac{1}{16}.
\end{aligned}$$

In der letzten Ungleichung haben wir die triviale untere Schranke $\text{maj}(x, C_1) \geq n/4$ verwendet. Insgesamt haben wir $\text{Prob}(D_2 > 0) \leq \text{Prob}(D_1 = 1) + 1/16$ gezeigt und damit können wir $(****)$ weiter vereinfachen.

$$\begin{aligned}
& \text{E}(D_1 \mid A, D_1 \leq 1) \cdot \text{Prob}(A, D_1 \leq 1) \\
\geq\ & \text{Prob}(D_2 > 0) \cdot \left(-\frac{13}{37}\right) + \text{Prob}(D_1 = 1) \cdot \left(1 - \frac{1}{n}\right)^{n/2} \\
\geq\ & \left(\text{Prob}(D_1 = 1) + \frac{1}{16}\right) \cdot \left(-\frac{13}{37}\right) + \text{Prob}(D_1 = 1) \cdot \left(1 - \frac{1}{n}\right)^{n/2} \\
=\ & \text{Prob}(D_1 = 1) \cdot \left(\left(1 - \frac{1}{n}\right)^{n/2} - \frac{13}{37}\right) - \frac{13}{592}
\end{aligned}$$

Der geklammerte Term ist für ausreichend große n positiv. Wie groß ist die Wahrscheinlichkeit $\text{Prob}(D_1 = 1)$? Eine hinreichende Bedingung für das Ereignis $D_1 = 1$ ist eine Mutation, die eines von $n/2 - \text{maj}(x, C_1)$ Minderheitsbits in C_1 flippt und die übrigen $n/2 - 1$ Bits von C_1 nicht flippt. Daher gilt

$$\text{Prob}(D_1 = 1) \geq \frac{n/2 - \text{maj}(x, C_1)}{n} \cdot \left(1 - \frac{1}{n}\right)^{n/2-1}.$$

Mit dieser Abschätzung erhalten wir

$$\begin{aligned}
& \text{E}(D_1 \mid A, D_1 \leq 1) \cdot \text{Prob}(A, D_1 \leq 1) \\
\geq\ & \frac{n/2 - \text{maj}(x, C_1)}{n} \cdot \left(1 - \frac{1}{n}\right)^{n/2-1} \cdot \left(\left(1 - \frac{1}{n}\right)^{n/2} - \frac{13}{37}\right) - \frac{13}{592} \\
=\ & \frac{n/2 - \text{maj}(x, C_1)}{n} \cdot \left(\left(1 - \frac{1}{n}\right)^{n-1} - \frac{13}{37} \cdot \left(1 - \frac{1}{n}\right)^{n/2-1}\right) - \frac{13}{592} \\
\geq\ & \frac{n/2 - \text{maj}(x, C_1)}{n} \cdot \left(\frac{1}{e} - \frac{13}{37} \cdot \left(1 - \frac{1}{n}\right)^{n/2-1}\right) - \frac{13}{592}
\end{aligned}$$

Der Term $(1 - 1/n)^{n/2-1}$ konvergiert gegen $e^{-1/2}$. Für ausreichend große n können wir die äußere Klammer nach unten abschätzen durch

$$\frac{1}{e} - \frac{13}{37} \cdot \left(1 - \frac{1}{n}\right)^{n/2-1} \geq \frac{1}{4e^{1/2}}.$$

Damit haben wir folgende Abschätzung für $\text{E}(D_1 \mid A, D_1 \leq 1)$ erreicht:

$$\text{E}(D_1 \mid A, D_1 \leq 1) \geq \frac{1}{\text{Prob}(A, D_1 \leq 1)} \cdot \left(\frac{n/2 - \text{maj}(x, C_1)}{n} \cdot \frac{1}{4e^{1/2}} - \frac{13}{592}\right)$$

Als nächstes wollen wir $1/\text{Prob}(A, D_1 \leq 1)$ nach unten abschätzen. Zuvor müssen wir jedoch noch nachweisen, dass der geklammerte Term nicht negativ ist. Dies lässt sich mit der Voraussetzung $\text{maj}(x, C_1) \leq (13/50)n$ zeigen:

$$\begin{aligned}
\frac{n/2 - \text{maj}(x, C_1)}{n} \cdot \frac{1}{4e^{1/2}} - \frac{13}{592} &\overset{\text{Vor.}}{\geq} \frac{n/2 - (13/50)n}{n} \cdot \frac{1}{4e^{1/2}} - \frac{13}{592} \\
&\geq \frac{12}{50} \cdot \frac{1}{4e^{1/2}} - \frac{13}{592} > 0{,}0144.
\end{aligned}$$

Wir sind nun fast am Ziel, es fehlt nur noch die untere Schranke für den Term $1/\text{Prob}(A, D_1 \leq 1)$. Aus Lemma 5 wissen wir, dass sich $\text{Prob}(A)$ nach oben durch $2 \cdot (n - \varphi(x))/n$ abschätzen lässt. Daher gilt die Ungleichungskette

$$\text{Prob}(A, D_1 \leq 1) \ \leq\ \text{Prob}(A) \ \leq\ 2 \cdot \frac{n - \varphi(x)}{n}.$$

Da für die Mehrheiten beider Cliquen $\text{maj}(x, C_1) \leq \text{maj}(x, C_2)$ gilt, gilt für die Minderheitsbits in beiden Cliquen $n/2 - \text{maj}(x, C_1) \geq n/2 - \text{maj}(x, C_2)$ und damit folgender Bezug zum Potenzial φ: $n/2 - \text{maj}(x, C_1) \geq (n - \varphi(x))/2$. Es folgt

$$\text{Prob}(A, D_1 \leq 1) \ \leq\ 2 \cdot \frac{n - \varphi(x)}{n} \ \leq\ 4 \cdot \frac{n/2 - \text{maj}(x, C_1)}{n}.$$

Unter Verwendung dieser Schranke erhalten wir

$$\begin{aligned}
\mathrm{E}(D_1 \mid A, D_1 \leq 1) &\geq \frac{1}{\mathrm{Prob}(A, D_1 \leq 1)} \cdot \left(\frac{n/2 - \mathrm{maj}(x, C_1)}{n} \cdot \frac{1}{4e^{1/2}} - \frac{13}{592} \right) \\
&\geq \frac{n}{4(n/2 - \mathrm{maj}(x, C_1))} \cdot \left(\frac{n/2 - \mathrm{maj}(x, C_1)}{n} \cdot \frac{1}{4e^{1/2}} - \frac{13}{592} \right) \\
&= \frac{1}{16e^{1/2}} - \frac{n}{4(n/2 - \mathrm{maj}(x, C_1))} \cdot \frac{13}{592}
\end{aligned}$$

Nun verwenden wir ein drittes Mal die Voraussetzung $\mathrm{maj}(x, C_1) \leq (13/50)n$:

$$\begin{aligned}
\mathrm{E}(D_1 \mid A, D_1 \leq 1) &\overset{\text{Vor.}}{\geq} \frac{1}{16e^{1/2}} - \frac{n}{4(n/2 - (13/50)n)} \cdot \frac{13}{592} \\
&= \frac{1}{16e^{1/2}} - \frac{n}{(24/25)n} \cdot \frac{13}{592} \\
&= \frac{1}{16e^{1/2}} - \frac{25}{24} \cdot \frac{13}{592} > \frac{3}{200}.
\end{aligned}$$

□

3.3.3 Lemma: Die Mehrheitsfarbe setzt sich durch

Wir haben gezeigt, dass die zurück liegende Clique bei kleinen Mehrheiten eine erwartete Drift von $\Omega(1)$ hat. In diesem Abschnitt wollen wir nun ein Anwendungsgebiet für diese Aussage entwickeln. Und zwar zeigen wir, dass es unwahrscheinlich ist, dass die Mehrheit der zurück liegenden Clique durch viele kleine Schritte stark absinkt, wenn bereits ein genügend großer Vorsprung vorhanden ist und es eine positive erwartete Drift gibt.

Diese Aussage gilt allgemein für alle Random Walks mit einer positiven erwarteten Drift (wie gewohnt unter der Bedingung, dass der aktuelle Zustand nur um höchstens 1 wächst), so dass wir eine allgemeinere Formulierung der Aussage wählen, die an das Drift-Argument aus Lemma 6 angelehnt ist und auf einer Abbildung α basiert. Für ein starkes Resultat brauchen wir als weitere Voraussetzung lediglich die Aussage, dass wir keinen riesig großen Sprung auf der α-Skala nach „links“, also in Richtung kleinerer Werte, machen.

Lemma 9. *Gegeben ein Wert $0 < \varepsilon < 1$, ein zeithomogener Random Walk auf dem Suchraum $\{0,1\}^n$ und eine Abbildung α von $\{0,1\}^n$ in ein Intervall $[a,b]$ ganzer Zahlen, $a, b \in \mathbb{Z}$:*

$$\alpha\colon \{0,1\}^n \to \mathbb{Z} \cap [a,b]$$

Wir betrachten nun ausgehend von einem Suchpunkt $x^0 \in \{0,1\}^n$ eine Folge von Schritten des Random Walks, in denen folgende Bedingungen erfüllt sind:

- *Es kommt nicht vor, dass sich der α-Wert des jeweils aktuellen Suchpunkts in einem Schritt um mindestens $n^{\varepsilon/2}$ verringert.*

- *Sei y der Suchpunkt, der ausgehend vom aktuellen Suchpunkt x im nächsten Schritt erreicht wird und $D_x := \alpha(y) - \alpha(x)$ eine Zufallsvariable, die abhängig von x die Veränderung des α-Werts im nächsten Schritt angibt. Den Wert r definieren wir als*

$$r := \min_{x,\alpha(x)<b} \mathrm{E}(D_x \mid D_x \leq 1).$$

  Dann soll gelten: $r \geq c > 0$ für eine Konstante c.

In einem zusammenhängenden Zeitintervall, in dem diese Bedingungen erfüllt sind, ist die Wahrscheinlichkeit, dass ein Suchpunkt x mit $\alpha(x) \leq \alpha(x^0) - n^{\varepsilon}/2$ erreicht wird, nach oben beschränkt durch $n^{-\Omega(n^{\varepsilon/2})}$.

Dies gilt auch dann, wenn zwischen zwei Schritten ein Suchpunkt durch einen anderen Suchpunkt mit gleichem α-Wert ersetzt wird.

Beweis. Zuerst definieren wir drei Ereignisse.

- Mit L bezeichnen wir das Ereignis, dass wir einen „Ausflug nach links“ machen und ein Zustand mit einem α-Wert „weit links“ vom letzten Rekordwert für α erreicht wird.

  Formal definieren wir das Ereignis L wie folgt. Es gibt einen beliebigen Suchpunkt x, für den gilt: Ausgehend vom Suchpunkt x wird zu einem beliebigen späteren Zeitpunkt ein beliebiger Suchpunkt z mit $\alpha(z) \leq \alpha(x) - n^{\varepsilon/2}$ erreicht, bevor ein beliebiger Suchpunkt y mit $\alpha(y) > \alpha(x)$ erreicht wird. Diese Konstellation ist Abbildung 3.2 skizziert.

  Abbildung 3.2: Eine Skizze der α-Skala mit α-Werten von x, y und z.

- Das Drift-Argument in Lemma 6 besagt: Der Erwartungswert der Zeit T_x, bis wir ausgehend von x mit $\alpha(x) < b$ einen Suchpunkt y mit $\alpha(y) > \alpha(x)$ erreichen, ist durch $1/r$ nach oben beschränkt.

  Mit Ereignis D (Delay) bezeichnen wir eine Verzögerung bei dieser Erhöhung des α-Werts: Ereignis D sei das Ereignis, dass $T_x \geq n^{\varepsilon/4}$.

- Ereignis J (Jump) sei das Ereignis, dass wir in einem Schritt den α-Wert des jeweils aktuellen Suchpunkts um mindestens $n^{\varepsilon/4}$ verringern.

Wir zeigen nun: Wenn wir einen Ausflug nach links machen und einen Suchpunkt z weit links von x erreichen, folgt eines der beiden Ereignisse D und J. Beide Ereignisse haben jeweils nur eine kleine Wahrscheinlichkeit, so dass die Wahrscheinlichkeit für einen Ausflug auch klein ist.

Angenommen, ein Ausflug nach links tritt ein und wir erreichen z von x aus, bevor y erreicht wird. Wir betrachten die Zahl T der Schritte, die wir benötigen, um von x nach z zu gelangen, und unterscheiden zwei Fälle.

Fall 1: Falls $T \geq n^{\varepsilon/4}$, benötigen wir mindestens T Schritte, um einen Suchpunkt y mit $\alpha(y) > \alpha(x)$ zu erreichen. Es folgt eine Verzögerung bei der Erhöhung des α-Werts ausgehend von x und damit Ereignis D.

Wie hoch ist die Wahrscheinlichkeit von Ereignis D? Wir wissen bereits, dass $\mathrm{E}(T_x) \leq 1/r$ ist. Nach Markoff hat daher Ereignis D eine Wahrscheinlichkeit von höchstens

$$\mathrm{Prob}(D) \leq \frac{1}{r \cdot n^{\varepsilon/4}}.$$

Fall 2: Falls $T < n^{\varepsilon/4}$, wird der α-Wert in weniger als $n^{\varepsilon/4}$ Schritten um mindestens $n^{\varepsilon/2}$ gesenkt. Nach dem Schubfachprinzip folgt, dass dabei ein Schritt den α-Wert um mindestens $n^{\varepsilon/4}$ senkt.

Wie hoch ist die Wahrscheinlichkeit eines solchen Sprungs? Dies lässt sich aus der erwarteten Drift ablesen: Da wir in der Konstanten $r > 0$ durch die Bedingung $D_x \leq 1$ nur Erhöhungen um höchstens 1 berücksichtigen und r trotzdem größer als Null ist, können große Schritte nach links nur eine kleine Wahrscheinlichkeit haben.

Dies werden wir im Folgenden ausnutzen. Zunächst formen wir die erwartete Drift um.

$$\begin{aligned}
\mathrm{E}(D_x \mid D_x \leq 1) &\overset{\text{Def.}}{=} \sum_{d=-\infty}^{\infty} \mathrm{Prob}(D_x = d \mid D_x \leq 1) \cdot d \\
&= \sum_{d=-\infty}^{\infty} \frac{\mathrm{Prob}(D_x = d, D_x \leq 1)}{\mathrm{Prob}(D_x \leq 1)} \cdot d \\
&= \frac{1}{\mathrm{Prob}(D_x \leq 1)} \cdot \sum_{d=-\infty}^{1} \mathrm{Prob}(D_x = d) \cdot d
\end{aligned}$$

Angenommen, die Wahrscheinlichkeit, um mindestens s nach links zu springen, wäre mindestens $1/s$. Dann machen die Summanden mit $d \leq -s$ in der Summe über d zusammen einen Wert von höchstens $1/s \cdot (-s) = -1$ aus. Wenn wir alle Summanden mit nichtpositiven Werten größer als $-s$ weglassen, erhalten wir folgende Abschätzung:

$$\begin{aligned}
\mathrm{E}(D_x \mid D_x \leq 1) &\leq \frac{1}{\mathrm{Prob}(D_x \leq 1)} \cdot \sum_{d \leq -s, d=1} \mathrm{Prob}(D_x = d) \cdot d \\
&\leq \frac{1}{\mathrm{Prob}(D_x \leq 1)} \cdot \left(\frac{1}{s} \cdot (-s) + \mathrm{Prob}(D_x = 1) \cdot 1 \right) \\
&= \frac{1}{\mathrm{Prob}(D_x \leq 1)} \cdot (\mathrm{Prob}(D_x = 1) - 1) \;\leq\; 0
\end{aligned}$$

Dies steht im Widerspruch zu $r > 0$. Also ist die Wahrscheinlichkeit, um mindestens s nach links zu springen, kleiner als $1/s$. Wenn wir $s := n^{\varepsilon/4}$ wählen, gilt damit

$$\mathrm{Prob}(J) \leq \frac{1}{n^{\varepsilon/4}}.$$

Der Ausflug nach links impliziert also entweder eine Verzögerung bei der Erhöhung von α oder einen großen Sprung nach links in einem Schritt. Wir haben also gezeigt, dass folgende Inklusion gilt:

$$L \subseteq D \cup J.$$

Für die Wahrscheinlichkeiten gilt

$$\text{Prob}(L) \leq \text{Prob}(D) + \text{Prob}(J) \leq \frac{1}{r \cdot n^{\varepsilon/4}} + \frac{1}{n^{\varepsilon/4}} \leq \frac{2}{r \cdot n^{\varepsilon/4}}.$$

Die letzte Ungleichung gilt, da $r \leq 1$.

Damit ist gezeigt, dass ein Ausflug nach links unwahrscheinlich ist. Wir zeigen nun, dass wir mindestens $n^{\varepsilon/2}/4$ solcher Ausflüge nacheinander durchführen müssen, damit der α-Wert insgesamt um mindestens $n^{\varepsilon}/2$ verringert wird.

Sei $x^0, x^1, \ldots$ die Folge von Suchpunkten, die der Random Walk über die Zeit durchläuft. Eine notwendige Bedingung für eine Verringerung um $n^{\varepsilon}/2$ ist, dass es einen Zeitpunkt t gibt, an dem zum ersten Mal ein Suchpunkt x^t mit $\alpha(x^t) \leq \alpha(x^0) - n^{\varepsilon/2}$ erreicht wird. Sei $m := \arg\max\{\alpha(x^0), \ldots, \alpha(x^t)\}$ ein Zeitpunkt, an dem ein maximaler α-Wert der Teilfolge $x^0, \ldots, x^t$ erreicht wurde. Wir wählen nun $z := x^t$ und $x := x^m$. Dann gilt: $\alpha(z) \leq \alpha(x) - n^{\varepsilon/2}$ und ausgehend von x wird der Suchpunkt z vor jedem Suchpunkt y mit $\alpha(y) > \alpha(x)$ erreicht, was der Charakterisierung aus Ereignis L entspricht.

Wir haben den Zeitpunkt t so gewählt, dass wir in x^t gegenüber x^0 zum ersten Mal eine Strecke von $n^{\varepsilon/2}$ nach links zurück gelegt haben. Da nur maximal $n^{\varepsilon/2}$ Bits in einem Schritt flippen, können wir gegenüber x^0 insgesamt nur um weniger als $2n^{\varepsilon/2}$ nach links gewandert sein.

Diese Betrachtungen können wir nun analog fortsetzen, indem wir anstelle der Folge $x^0, x^1, \ldots$ die Teilfolge $x^t, x^{t+1}, \ldots$ betrachten. Es lassen sich analog weitere Ausflüge nach links ausmachen, die uns aber jeweils nur um weniger als $2n^{\varepsilon/2}$ nach links führen.

Um α um insgesamt $n^{\varepsilon}/2$ zu verringern, benötigen wir daher mehr als

$$\frac{n^{\varepsilon}/2}{2n^{\varepsilon/2}} = \frac{n^{\varepsilon-\varepsilon/2}}{4} = \frac{n^{\varepsilon/2}}{4}$$

Ausflüge. Da die betrachteten Teilfolgen der α-Werte zeitlich aufeinander folgen, sind die Ereignisse L unabhängig voneinander. Also ist die Wahrscheinlichkeit, den α-Wert um $n^{\varepsilon/2}$ zu verringern, höchstens

$$(\text{Prob}(L))^{n^{\varepsilon/2}/4} \leq \left(\frac{1}{r \cdot n^{\varepsilon/4}}\right)^{n^{\varepsilon/2}/4} = n^{-\Omega(n^{\varepsilon/2})}.$$

□

Es mag verwundern, dass Lemma 9 für einen beliebig langen Zeitraum garantiert, dass der α-Wert nicht um $n^{\varepsilon/2}$ absinkt. Dies ist darin begründet, dass das Ereignis

L eines „Ausflugs“ für einen beliebig langen Zeitraum definiert ist. (Ein langer Weg von x zu z kann nicht schaden, denn je länger der Ausflug, um so größer wird die Verzögerung.) Das übergeordnete Ereignis, dass der α-Wert in einem beliebig langen Zeitraum um $n^{\varepsilon/2}$ absinkt, benötigt mehrere Einzelereignisse, nämlich dass man in beliebig langen Zeiträumen mehrere Ausflüge nach links macht. Dies ermöglicht die Rückführung auf mehrere unabhängige Ereignisse L.

Allerdings gilt Lemma 9 nur, so lange alle Voraussetzungen erfüllt sind. Dadurch wird der Zeitrahmen auf natürliche Weise begrenzt, denn spätestens wenn ein maximaler α-Wert von b erreicht wird, ist die erwartete Drift von $\Omega(1)$ nicht mehr gegeben.

Weiter sei erwähnt, dass der Beweis von Lemma 9 weitgehend unabhängig von der Verteilung von D_x arbeitet. Aus der erwarteten Drift lässt sich zum einen mit dem Drift-Argument aus Lemma 6 herleiten, dass die erwartete Zeit bis zu einem höheren α-Wert klein ist. Zum anderen nutzen wir die Bedingung $D_x \leq 1$ aus, dass in der erwarteten Drift der aktuelle Zustand nur um 1 wächst, um die Wahrscheinlichkeit von Sprüngen um $n^{\varepsilon/4}$ nach links zu beschränken. Dies ist ein Argument, das sich ausschließlich aus der erwarteten Drift ergibt und dem Beweis der Markoff-Ungleichung (Theorem 1) ähnelt.

Der Beweis benötigt lediglich die Aussage, dass Sprünge um mindestens $n^{\varepsilon/2}$ nach links ausgeschlossen sind, um zu zeigen, dass wir viele „Ausflüge“ hintereinander durchführen müssen, um den α-Wert start abzusenken. Das Argument aus der erwarteten Drift ist hier zu schwach, da wir für die Wahrscheinlichkeit, mindestens $n^{\varepsilon/2}$ α-Werte nach links zu springen, nur eine untere Schranke von $n^{-\varepsilon/2}$ erhalten würden. Ohne jegliche Annahmen über die Verteilung von D_x können wir daher nicht ausschließen, dass man mit einer Wahrscheinlichkeit von $n^{-\varepsilon/2}$ direkt von x^0 aus zu einem um mindestens $n^{\varepsilon/2}$ kleineren α-Wert springt.

3.3.4 Beweis des Theorems

Mit der geleisteten Vorarbeit können wir uns nun an den Beweis des angepeilten Theorems wagen, das zeigen soll, dass der (1+1) EA mit einer Wahrscheinlichkeit von mindestens $1/2 - o(1)$ ein lokales Optimum mit verschieden gefärbten Cliquen erreicht. Zudem zeigen wir zwei weitere Aussagen: Aus der Wahrscheinlichkeit $1/2 - o(1)$ folgt leicht, dass die erwartete Optimierungszeit des (1+1) EA in der Größenordnung $\Theta(n^{n/2})$ liegt. Damit lässt sich beweisen, dass der Graph G mit zwei durch eine Brückenkante verbundene Cliquen der Größe $n/2$ eine Worst-Case-Instanz ist, die zu einer asymptotischen Worst-Case-Laufzeit führt. Drittens beweisen wir eine Schranke für die erwartete Zeit, ein lokales oder globales Optimum zu erreichen; diese Schranke wird uns bei der Analyse der Wahrscheinlichkeit helfen, mit der ein rein lokales Optimum erreicht wird.

Ein Hindernis ist allerdings noch zu überwinden. Wir wollen im Beweis des Theorems Drift-Aussagen aus Korollar 2 und Lemma 8 verwenden. Diese Aussagen beziehen sich jedoch auf die erwartete Drift in erfolgreichen Schritten, sprich: Schritten, in denen mindestens eine Mehrheit erhöht wird. Bei Cliquen ohne Brückenkanten ist eine

Beschränkung auf erfolgreiche Schritte korrekt, da jeder relevante Schritt (sprich: mindestens eine Mehrheit wird verändert) auch ein erfolgreicher Schritt ist: Wenn mindestens eine Mehrheit sinkt und keine Mehrheit steigt, verschlechtert sich die Fitness und der neue Suchpunkt wird nicht akzeptiert.

Wenn Brückenkanten ins Spiel kommen, ändert sich die Situation: Es kann vorkommen, dass sich durch Mehrbitflips die Färbung von Brückenkanten verbessert. Wenn dieser Effekt groß genug ist, kann es passieren, dass in einer Clique die Mehrheit sinkt und der Schritt akzeptiert wird, obwohl keine Mehrheit wächst.

Bezogen auf das Potenzial φ, die Summe aller Mehrheiten, haben wir hier also Schritte, die das Potenzial zusätzlich zu den bekannten Effekten nur senken können. Es ist nicht mehr korrekt, sich allein auf erfolgreiche Schritte zu beschränken, sondern wir müssen den Random Walk des Potenzials auf alle relevanten Schritte ausweiten.

Wenn es nur eine einzige Brückenkante gibt, ist ein relevanter und nicht erfolgreicher Schritt jedoch sehr unwahrscheinlich. Wir werden zeigen, dass die erwartete Drift in relevanten Schritten nur um einen Term $O(1/n)$ kleiner ist als die erwartete Drift in erfolgreichen Schritten. So lassen sich unsere Drift-Aussagen über erfolgreiche Schritte auf relevante Schritte übertragen.

Theorem 12. *Gegeben ein Cliquengraph $G = (V, E)$ mit $n := |V|$ Knoten und zwei disjunkten Cliquen C_1, C_2 der Größe $n/2$, die durch eine Brückenkante verbunden sind. Dann gilt für $f := \mathrm{Ising}_G$ und ausreichend große n:*

(*i*) *Die erwartete Zeit, bis der (1+1) EA ausgehend von einem beliebigen Startpunkt auf f ein lokales oder globales Optimum erreicht, ist nach oben beschränkt durch $8n \log n$.*

(*ii*) *Mit einer Wahrscheinlichkeit von mindestens $1/2 - o(1)$ erreicht der (1+1) EA auf f ein lokales Optimum, bei dem beide Cliquen einheitlich, aber mit verschiedenen Farben gefärbt sind.*

(*iii*) *Die erwartete Optimierungszeit des (1+1) EA auf f ist $\Theta(n^{n/2})$.*

Beweis. Aussage (*iii*) folgt aus der allgemeinen oberen Schranke $O(n^{n/2})$ in Theorem 5 und aus Aussage (*ii*). Angenommen, ein lokales, nicht globales Optimum mit verschieden gefärbten Cliquen wird erreicht. Dann ist eine notwendige Bedingung dafür, dass ein Schritt akzeptiert wird, dass mindestens eine Clique komplett in einem Schritt flippt. Die erwartete Zeit für ein solches Ereignis ist $\Omega(n^{n/2})$, die bedingte erwartete Optimierungszeit in dieser Situation also $\Omega(n^{n/2})$. Aus der Definition bedingter Erwartungswerte folgt eine unbedingte erwartete Optimierungszeit von mindestens $(1/2 - o(1)) \cdot \Omega(n^{n/2}) = \Omega(n^{n/2})$.

Als nächstes zeigen wir Aussage (*i*) mit Hilfe von Korollar 2, das uns in erfolgreichen Schritten eine Drift von $1/2$ garantiert.

Wir haben bereits erwähnt, dass fast jeder relevante Schritt auch ein erfolgreicher Schritt ist und ein relevanter, aber nicht erfolgreicher Schritt selten ist. Bei nur einer

Brückenkante ist dies ein extrem seltener Sonderfall, der nur dann eintreten kann, wenn die Cliquengröße $n/2$ gerade ist, die Brückenkante im aktuellen Suchpunkt x zweifarbig ist und in einer Clique C die Mehrheit genau den Wert $\mathrm{maj}(x, C) = n/4+1$ annimmt. In diesem Fall kann die Mehrheit in C um 1 sinken, was die Fitness um 1 senkt. Wenn gleichzeitig (z. B. wenn zwei weitere passende Bitflips eine 0 und eine 1 innerhalb der gleichen Clique treffen) die zweifarbige Brückenkante einfarbig gefärbt wird, wird der Schritt trotzdem akzeptiert. Einen solchen Schritt nennen wir im Folgenden einen *pathologischen Schritt*.

Da für einen pathologischen Schritt ein Endpunkt der Brückenkante flippen muss, ist die unbedingte Wahrscheinlichkeit eines solchen Schrittes in dem geschilderten Szenario höchstens $2/n$. In diesem Szenario gibt es jedoch, da $\mathrm{maj}(x, C) = n/4 + 1$, auch eine lineare Zahl von Minderheitsbits. Jede 1-Bit-Mutation eines Minderheitsbits führt zu einem relevanten Schritt, so dass die Wahrscheinlichkeit eines relevanten Schrittes $\Omega(1)$ ist. Damit ist die bedingte Wahrscheinlichkeit, dass ein relevanter Schritt ein pathologischer Schritt ist, beschränkt durch $O(1/n)$.

Welchen Einfluss hat ein pathologischer Schritt auf die erwartete Drift des Potenzials φ in relevanten Schritten, verglichen mit der erwarteten Drift in erfolgreichen Schritten? Da ein pathologischer Schritt selten ist und er das Potenzial nur um 1 senkt, ist der Einfluss auf die erwartete Drift in relevanten Schritten klein. Sei R das Ereignis, dass ein Schritt relevant ist und A das Ereignis, dass ein Schritt erfolgreich ist. Angenommen, die erwartete Drift in *erfolgreichen* Schritten ist durch einen Wert $0 < r \leq 1$ nach unten beschränkt: $\mathrm{E}(D_x \mid D_x \leq 1, A) \geq r$. Dann ist die erwartete Drift in *relevanten* Schritten nach unten beschränkt durch

$$\begin{aligned}
\mathrm{E}(D_x \mid D_x \leq 1, R) &= \mathrm{Prob}(A \mid D_x \leq 1, R) \cdot \mathrm{E}(D_x \mid D_x \leq 1, R, A) \\
&\quad + \mathrm{Prob}(\overline{A} \mid D_x \leq 1, R) \cdot \mathrm{E}(D_x \mid D_x \leq 1, R, \overline{A}) \\
&\geq \left(1 - O\left(\frac{1}{n}\right)\right) \cdot r + O\left(\frac{1}{n}\right) \cdot (-1) \\
&= r - O\left(\frac{1}{n}\right)
\end{aligned}$$

Für ausreichend große n erhalten wir wieder eine positive Konstante als untere Schranke, so dass wir den Einfluss pathologischer Schritte weitestgehend vernachlässigen können.

Wir werden an mehreren Stellen des Beweises auf ein ähnliches Ereignis treffen, das – genau wie ein pathologischer Schritt – sehr unwahrscheinlich ist, aber dennoch die erwartete Drift in relevanten Schritten beeinflusst. Und zwar werden wir an einigen Stellen Drift-Argumente anwenden unter der Bedingung, dass höchstens $n^{1/3}$ Bits in einem Schritt flippen.

Sei F das Ereignis, dass mehr als $n^{1/3}$ Bits in einem Schritt flippen. Wenn wir über die erwartete Drift des Potenzials φ in relevanten Schritten unter der Bedingung $\overline{F}$ reden, beeinflusst die Bedingung $\overline{F}$ die erwartete Drift, da beispielsweise Senkungen des Potenzials um mehr als $n^{1/3}$ ausgeschlossen sind. Wir werden daher $\overline{F}$ als zusätzliche Bedingung berücksichtigen und zeigen, dass auch unter dieser Bedingung

die bedingte erwartete Drift $\mathrm{E}(D_x \mid D_x \leq 1, R, \overline{F})$ in relevanten Schritten durch eine positive Konstante nach unten beschränkt ist. Dies wird den Rest des Beweises sehr erleichtern.

Das Ereignis F hat eine Wahrscheinlichkeit von höchstens

$$\mathrm{Prob}(F) \leq \frac{1}{(n^{1/3})!}.$$

Sei R das Ereignis, dass ein Schritt relevant ist und $r > 0$ eine untere Schranke für $\mathrm{E}(D_x \mid D_x \leq 1, R)$. Mit der Definition bedingter Erwartungswerte spalten wir die erwartete Drift auf in bedingte Erwartungswerte, in denen zusätzliche Bedingungen F bzw. $\overline{F}$ gegeben sind.

$$\begin{array}{rl} & \mathrm{E}(D_x \mid D_x \leq 1, R) \\ = & \mathrm{Prob}(F) \cdot \mathrm{E}(D_x \mid D_x \leq 1, R, F) + \mathrm{Prob}(\overline{F}) \cdot \mathrm{E}(D_x \mid D_x \leq 1, R, \overline{F}) \\ \overset{\text{Vor.}}{\geq} & r. \end{array}$$

Wenn wir von beiden Seiten der Ungleichung $\mathrm{Prob}(\overline{F}) \cdot \mathrm{E}(D_x \mid D_x \leq 1, R, \overline{F})$ subtrahieren und beide Seiten durch $\mathrm{Prob}(\overline{F})$ teilen, erhalten wir

$$E(D_x \mid D_x \leq 1, R, \overline{F}) \geq \frac{1}{\mathrm{Prob}(\overline{F})} \cdot (r + \mathrm{Prob}(F) \cdot \mathrm{E}(D_x \mid D_x \leq 1, R, F))$$

Die bedingte erwartete Drift für D_x ist unter beliebigen Bedingungen trivialerweise durch $-n$ nach unten beschränkt, da der Wertebereich für φ nur Werte zwischen $n/2$ und n umfasst. Zweitens gilt $1/\mathrm{Prob}(\overline{F}) \geq 1$, da $\mathrm{Prob}(\overline{F}) \leq 1$ trivialerweise gilt. Es folgt

$$\begin{array}{rcl} E(D_x \mid D_x \leq 1, R, \overline{F}) & \geq & r - \mathrm{Prob}(F) \cdot n \\ & \geq & r - \dfrac{n}{(n^{1/3})!}. \end{array}$$

Diesen Term können wir ebenfalls durch $r - O(1/n)$ nach unten abschätzen.

Insgesamt haben wir gezeigt, dass sowohl der Einfluss pathologischer Schritte, als auch der Einfluss der Bedingung, dass höchstens $n^{1/3}$ Bits in einem Schritt flippen, die erwartete Drift des Potenzials φ in relevanten Schritten jeweils nur um einen Subtrahenden in der Größenordnung $O(1/n)$ senkt, verglichen mit der erwarteten Drift in erfolgreichen Schritten.

Die gleichen Resultate kann man analog auch für die erwartete Drift eines anderen Random Walks zeigen, nämlich den Random Walk auf der Mehrheit der zurück liegenden Clique. Auch hier verringert ein pathologischer Schritt die Mehrheit höchstens um 1, so dass die Drift in relevanten Schritten um $O(1/n)$ sinkt. Beim Ereignis F können wir die gleichen Abschätzungen und Rechnungen verwenden wie beim Random Walk des Potenzials φ.

Wir fassen die gezeigten Resultate zusammen.

- Aus Korollar 2 und der Betrachtung negativer Einflüsse folgt: Die erwartete Drift des Potenzials φ in relevanten Schritten ist durch $1/2 - O(1/n)$ nach unten beschränkt. Dies gilt auch dann, wenn höchstens $n^{1/3}$ Bits in einem Schritt flippen dürfen.
- Aus Lemma 8 und der Betrachtung negativer Einflüsse folgt: Falls die Bedingungen für Lemma 8 erfüllt sind, ist die erwartete Drift der Mehrheit der zurück liegenden Clique in relevanten Schritten mindestens $3/200 - O(1/n)$. Dies gilt auch dann, wenn höchstens $n^{1/3}$ Bits in einem Schritt flippen dürfen.

Mit diesen Erkenntnissen und den aus Abschnitt 3.2 bekannten Argumenten ist es nun leicht, Aussage (i) zu zeigen. Wir wissen, dass genau die lokalen Optima ein Potenzial von n haben; zu zeigen ist also eine Schranke für die erwartete Zeit, ein Potenzial von n zu erreichen.

Bei einer Drift von mindestens $1/2 - O(1/n)$ in relevanten Schritte benötigen wir, wenn wir Lemma 6 auf alle relevanten Schritte anwenden, im Erwartungswert nur $2 + O(1/n)$ relevante Schritte, um einen neuen Rekordwert für φ zu erreichen. Die Wahrscheinlichkeit für einen relevanten Schritt ausgehend von x beträgt mindestens $(n - \varphi(x))/en$, da eine 1-Bit-Mutation, die eines von $n - \varphi(x)$ Minderheitsbits flippt, zu einem relevanten Schritt führt. Die erwartete Zeit, bis ein Suchpunkt mit Potenzial n erreicht wird, ist daher für jeden Startpunkt nach oben beschränkt durch

$$\begin{aligned} 1 + \sum_{i=1}^{\lfloor n/2 \rfloor} \frac{en}{n-i} \cdot (2 + O(1/n)) &\leq 1 + (2 + O(1/n)) \cdot en(\ln n + 1) \\ &= 2en \cdot \log(e) \cdot \log(n) + O(n). \end{aligned}$$

Für ausreichend große n lässt sich dies nach oben durch $8n \log n$ abschätzen.

Nun bleibt nur noch Aussage (ii) zu zeigen. Dazu betrachten wir typische Läufe, die zu einem lokalen Optimum mit verschieden gefärbten Cliquen führen und zeigen, dass ein solcher typischer Lauf mit einer Wahrscheinlichkeit von mindestens $1/2 - o(1)$ eintritt.

Gemäß dem Ansatz von Wegener und Witt [22] teilen wir einen typischen Lauf in zwei Phasen ein. In Phase 1 warten wir darauf, dass sich ausreichend große Mehrheiten heraus kristallisieren und in Phase 2 spekulieren wir darauf, dass sich die am Ende von Phase 1 vorhandenen Mehrheiten am Ende auch durchsetzen. Die Zahl der Schritte, die der (1+1) EA in Phase i zubringt, bezeichnen wir mit T_i. Jeder Phase ist ein Ziel zugeordnet, dessen Erreichen die Phase beendet und die nächste Phase einleitet. Phase 1 hat zudem ein Zwischenziel, das uns helfen wird, die erwartete Zeit von Phase 1 besser abzuschätzen. In jeder Phase gibt es Fehlerereignisse, die nicht eintreten dürfen, da ansonsten der Lauf nicht als typisch gilt.

Zusätzlich gibt es in jeder Phase ein Zeitlimit, das in einem typischen Lauf nicht überschritten werden darf. Um Missverständnissen vorzubeugen, sei gesagt: Das Zeitlimit dient nicht dazu, die erwartete Zeit des Algorithmus für eine Phase auszurechnen, sondern dient ausschließlich dazu, die Fehlerereignisse in einer Phase auf

ein vernünftiges Zeitintervall zu beschränken. Ohne eine Zeitbeschränkung lassen sich die Fehler, die in jedem Schritt eintreten können, nicht mehr kontrollieren, da in beliebig vielen Versuchen praktisch jeder Fehler mit hoher Wahrscheinlichkeit eintritt.

Wir beschränken daher das Zeitintervall, in dem Fehler eintreten können, indem wir für jede Phase ein Zeitlimit ansetzen. Der Preis für diese Strategie ist, dass es nun neue Fehler für das Überschreiten des Zeitlimits gibt: Wenn das Zeitlimit überschritten wird, werten wir den Lauf als untypisch, da wir die anderen Fehler nicht über das Zeitlimit hinaus kontrollieren können.

Ein typischer Lauf verläuft nun wie folgt.

Phase 1

Zwischenziel: Ein Suchpunkt x^* wird erreicht mit $\varphi(x^*) \geq n/2 + 6n^{2/3} + 2$.

Ziel: Ein Suchpunkt x^{**} wird erreicht mit $\mathrm{maj}(x^{**}, C_1), \mathrm{maj}(x^{**}, C_2) \geq n/4 + 2n^{2/3}$.

Zeitlimit: Phase 1 benötigt höchstens $n^{3/4}$ Schritte: $T_1 \leq n^{3/4}$.

Fehler F_1: In den ersten $\min\{T_1, n^{3/4}\}$ Schritten flippt in einer Mutation ein Endpunkt der Brückenkante.

Fehler F_2: Am Ende von Phase 1 sind die Mehrheitsfarben in C_1 und C_2 gleich.

Phase 2

Ziel: Startend mit x^{**} wird ein lokales oder globales Optimum erreicht.

Zeitlimit: Phase 2 benötigt höchstens $n^2 \log n$ Schritte: $T_2 \leq n^2 \log n$.

Fehler F_3: In den ersten $\min\{T_2, n^2 \log n\}$ Schritten von Phase 2 flippen mindestens $n^{1/3}$ Bits in einem Schritt.

Fehler F_4: In den ersten $\min\{T_2, n^2 \log n\}$ Schritten von Phase 2 wird ein Suchpunkt x erreicht, bei dem für eine Clique C gilt: $\mathrm{maj}(x, C) < n/4 + n^{2/3}/2$.

Analyse von Phase 1

Wir zeigen zuerst, dass die Fehler F_1 und F_2 zusammen mit einer Wahrscheinlichkeit von $1/2 + o(1)$ eintreten. Dazu verwenden wir folgende Rechnung, wobei wir in der Ungleichung ausnutzen, dass $\mathrm{Prob}(A, B) = \mathrm{Prob}(A \mid B) \cdot \mathrm{Prob}(B) \leq \mathrm{Prob}(A \mid B)$.

$$\begin{aligned} \mathrm{Prob}(F_1 \cup F_2) &= \mathrm{Prob}(F_1 \,\dot\cup\, (F_2 \cap \overline{F_1})) \\ &= \mathrm{Prob}(F_1) + \mathrm{Prob}(F_2, \overline{F_1}) \\ &\leq \mathrm{Prob}(F_1) + \mathrm{Prob}(F_2 \mid \overline{F_1}) \end{aligned}$$

Die Wahrscheinlichkeit, dass in höchstens $n^{3/4}$ Schritten einer von zwei Endpunkten der Brückenkante flippt, ist nach oben beschränkt durch

$$\text{Prob}(F_1) \leq n^{3/4} \cdot \frac{2}{n} = o(1).$$

Nun zur Wahrscheinlichkeit F_2 gegeben $\overline{F_1}$, dass beide Mehrheitsfarben am Ende von Phase 1 verschieden bzw. gleich sind. Wir nehmen an, dass $\overline{F_1}$ eintritt und das Zeitlimit eingehalten wird, was bedeutet, dass in der gesamten Phase 1 kein Endpunkt der Brückenkante flippt. Nun verwenden wir unsere in Abschnitt 2.4 erarbeiteten Resultate über unabhängige Färbungen ausgewählter Pivot-Knoten in disjunkten Knotenmengen. In Definition 3 haben wir eine Folge von Partitionen $\mathcal{P}^0, \mathcal{P}^1, \ldots$ über die Zeit definiert, die mögliche Abhängigkeiten zwischen Knotenmengen widerspiegeln. In Lemma 4 haben wir gezeigt, dass die Färbungen einzelner Pivot-Knoten aus verschiedenen Mengen einer Partition $\mathcal{P}^t$ zum Zeitpunkt t unabhängig voneinander sind.

Wenn in Phase 1, also in den ersten T_1 Schritten, kein Endpunkt der Brückenkante flippt, können wir die Partition $\mathcal{P}^{T_1}$ wie folgt charakterisieren.

Angenommen, unter der Bedingung $\overline{F_1}$ entsteht zu einem Zeitpunkt $t \leq T_1$ in der Partition $\mathcal{P}^t$ zum ersten Mal eine Menge S_i^t, die Knoten aus beiden Cliquen enthält (formal: $S_i^t \cap C_1 \cap C_2 \neq \emptyset$). Da zu allen Zeitpunkten $t' < t$ alle Mengen aus $\mathcal{P}^{t'}$ nur Knoten aus einer Clique enthalten, muss S_i^t dadurch entstanden sein, dass ein Knoten flippt, der zu Knoten aus verschiedenen Cliquen adjazent ist. Da hierfür nur die Endpunkte der Brückenkante in Frage kommen, steht dies im Widerspruch zu $\overline{F_1}$.

Wir haben also gezeigt, dass unter der Bedingung $\overline{F_1}$ die Partition $\mathcal{P}^{T_1}$ nur Mengen enthält, die auf Knoten einer Clique beschränkt sind. (Falls in jeder Clique in Phase 1 mindestens ein Bitflip auftritt, ist $\mathcal{P}^{T_1} = \{C_1, C_2\}$. Tritt hingegen in einer Clique C_i in der gesamten Phase 1 kein Bitflip auf, bilden alle Knoten der Clique einzelne Mengen in $\mathcal{P}^{T_1}$.)

Als Pivot-Knoten wählen wir nun zwei Knoten $u \in C_1, v \in C_2$, die mit den jeweiligen Mehrheitsfarben ihrer Cliquen gefärbt sind. Lemma 4 besagt, dass die Färbungen von u und v zum Zeitpunkt T_1 unabhängig voneinander sind. Daher sind sie mit Wahrscheinlichkeit $1/2$ mit verschiedenen Farben gefärbt. Da diese Färbungen nach Wahl der Pivot-Knoten den beiden Mehrheitsfarben entsprechen, gilt auch für die Mehrheitsfarben, dass sie mit Wahrscheinlichkeit $1/2$ verschieden sind.

Wir haben also gezeigt:

$$\text{Prob}(F_2 \mid \overline{F_1}) = \frac{1}{2}.$$

Insgesamt können wir die Fehlerwahrscheinlichkeiten in Phase 1 abschätzen durch

$$\text{Prob}(F_1 \cup F_2) \leq \text{Prob}(F_1) + \text{Prob}(F_2 \mid \overline{F_1}) = \frac{1}{2} + o(1).$$

Als nächstes kümmern wir uns um die Wahrscheinlichkeit, mit der das Zeitlimit überschritten wird. Dabei hilft uns das Zwischenziel x^*; wir teilen die erwartete Zeit in Phase 1 auf in die erwartete Zeit bis zum Zwischenziel x^* und in die erwartete restliche Zeit in Phase 1 ausgehend von x^*.

Zuerst schätzen wir die erwartete Zeit bis zum Erreichen des Zwischenziels x^* ab. Wir haben im Beweis von Aussage (i) gezeigt, dass der (1+1) EA im Erwartungswert nur höchstens $2 + O(1/n)$ relevante Schritte benötigt, um einen neuen Rekordwert für das Potenzial φ zu erreichen.

Selbst wenn der Lauf mit dem minimalen Wert $\varphi(x) = \lceil n/2 \rceil$ gestartet wird, ist die erwartete Zahl nötiger relevanter Schritte, bis ein Suchpunkt x^* mit $\varphi(x^*) \geq n/2 + 6n^{2/3} + 2$ erreicht ist, in der Größenordnung $O(n^{2/3})$. So lange das Ziel von Phase 1 noch nicht erreicht wurde, gilt $\varphi(x) = n - \Theta(n)$ und es gibt $\Theta(n)$ Minderheitsbits. Eine 1-Bit-Mutation, die ein solches Bit flippt, ist ein relevanter Schritt. Die erwartete Wartezeit auf einen relevanten Schritt ist daher $O(1)$ und die erwartete Zeit, bis das Zwischenziel erreicht ist, ist daher immer noch in der Größenordnung $O(n^{2/3})$.

In x^* ist die Summe der Mehrheiten ausreichend groß, so dass die Bedingungen, die wir an x^{**} stellen, erfüllt sein können. Dies muss aber nicht der Fall sein, da wir für x^{**} verlangen, dass *beide* Mehrheiten mit Werten von mindestens $n/4 + 2n^{2/3}$ ausreichend groß sein sollen. Daher fragen wir uns nun, wie groß die erwartete Zeit ist, bis wir ausgehend von x^* einen Suchpunkt x^{**} erreichen.

Angenommen, wir haben einen Suchpunkt x^* mit $\varphi(x^*) \geq n/2 + 6n^{2/3} + 2$ erreicht. Dann sagt uns Lemma 7 zur Größe der maximalen Mehrheit, angewandt auf die Suchpunkte $x := x^*$ und einen beliebigen, später erreichten Suchpunkt y, dass die maximale Mehrheit in y stets mindestens

$$\frac{\varphi(x^*)}{2} - 1 \;\geq\; \frac{n/2 + 6n^{2/3} + 2}{2} - 1 \;=\; \frac{n}{4} + 3n^{2/3}$$

beträgt. Damit erfüllt die jeweils führende Clique stets die Bedingung von x^{**}; es muss also jetzt nur noch die jeweils zurück liegende Clique eine Mehrheit von mindestens $n/4 + 2n^{2/3}$ erreichen.

Wir wollen nun den Random Walk der zurück liegenden Clique analysieren. Dabei ergibt sich jedoch das Problem, dass die führende Clique und die zurück liegende Clique ihre Rollen tauschen können, indem die zurück liegende Clique die führende Clique überholt. Ein solches Ereignis wirkt sich störend auf die Analyse des Random Walks aus; wir zeigen daher, dass ein Wechsel der zurück liegenden Clique unwahrscheinlich ist.

Da die Mehrheit der führenden Clique mindestens $n/4 + 3n^{2/3}$ beträgt, ist die Differenz beider Mehrheiten mindestens $n^{2/3}$, so lange x^{**} noch nicht erreicht wurde. Für einen Wechsel der zurück liegenden Clique müssen also mindestens $n^{2/3}$ Bits in einem Schritt flippen. Wir nehmen an, dass in Phase 1 ab dem Erreichen von x^* in jedem Schritt nur höchstens $n^{1/3}$ Bits flippen; eine Mutation, die mehr Bits flippt, tritt in höchstens $n^{3/4}$ Schritten nur mit exponentiell kleiner Wahrscheinlichkeit ein.

In der folgenden Analyse des Random Walks der Mehrheit der zurück liegenden Clique nehmen wir an, dass die zurück liegende Clique nicht wechselt. Falls dies trotzdem geschieht, starten wir eine neue Analyse des betrachteten Random Walks. Dies vergrößert die erwartete Laufzeit in Phase 1 nur um einen Faktor $1 + o(1)$.

O. B. d. A. sei C_1 die zurück liegende und C_2 die führende Clique. Mit x bezeichnen wir den aktuellen Suchpunkt des (1+1) EA. Wir haben bereits gezeigt, dass die Mehrheit der zurück liegenden Clique in relevanten Schritten eine erwartete Drift von mindestens $3/200 - O(1/n)$ hat, selbst dann, wenn die Mehrheit nur um 1 wachsen kann und wir annehmen, dass höchstens $n^{1/3}$ Bits in einem Schritt flippen. (Die Bedingungen $\mathrm{maj}(x, C_1) \leq \mathrm{maj}(x, C_2)$ und $\mathrm{maj}(x, C_1) \leq (13/50)n$ für Lemma 8 sind für ausreichend große n erfüllt.)

Wir wenden nun das Drift-Argument aus Lemma 6 auf den Random Walk in relevanten Schritten an, wobei wir diesmal $\alpha(x) := \mathrm{maj}(x, C_1)$ wählen. In nicht relevanten Schritten kann sich zwar der aktuelle Suchpunkt x, nicht aber dessen α-Wert verändern. Daher ist es korrekt, sich auf relevante Schritte zu beschränken und die erwartete Zahl relevanter Schritte, bis die Mehrheit einen höheren Rekordwert erreicht hat, ist nach Lemma 6 in der Größenordnung $O(1)$.

Wie zuvor ist die Wartezeit auf einen relevanten Schritt $O(1)$. Falls die zurück liegende Clique C_1 nicht wechselt, ist daher die erwartete Zeit, bis ein Suchpunkt x^{**} mit $\mathrm{maj}(x^{**}, C_1) \geq n/4 + 2n^{2/3}$ erreicht wird, in der Größenordnung $O(n^{2/3})$.

Wenn man beide Teile von Phase 1 zusammen nimmt, lässt sich die erwartete Zeit, bis wir das Ziel von Phase 1 erreichen, nach oben abschätzen durch $(1+o(1))\cdot O(n^{2/3})$. Nach Markoff ist die Wahrscheinlichkeit, dass wir hierfür mehr als $n^{3/4}$ Schritte brauchen,

$$\mathrm{Prob}(T_1 > n^{3/4}) \leq \frac{O(n^{2/3})}{n^{3/4}} = O(n^{-1/12}) = o(1).$$

Eine Überschreitung des Zeitlimits ist also unwahrscheinlich.

Analyse von Phase 2

Nun zur Analyse von Phase 2. Genau wie in Phase 1 verwenden wir die Formel

$$\mathrm{Prob}(F_3 \cup F_4) \leq \mathrm{Prob}(F_3) + \mathrm{Prob}(F_4 \mid \overline{F_3}),$$

um die gesamte Fehlerwahrscheinlichkeit in Phase 2 abzuschätzen.

Zunächst zu F_3. Die Wahrscheinlichkeit, dass in mindestens einem von höchstens $n^2 \log n$ Schritten mindestens $n^{1/3}$ Bits flippen, ist nach oben beschränkt durch

$$\mathrm{Prob}(F_3) \leq n^2 \log n \cdot \frac{1}{(n^{1/3})!} = n^2 \log n \cdot 2^{-\Omega(n^{1/3} \cdot \log n)} = o(1).$$

Mit welcher Wahrscheinlichkeit tritt mit F_4 ein Suchpunkt auf, bei dem eine Mehrheit unter den Schwellenwert $n/4 + n^{2/3}/2$ fällt?

In der Analyse von Phase 1 haben wir bereits gezeigt, dass die jeweils führende Clique immer eine Mehrheit von mindestens $\varphi(x^*)/2 - 1 \geq n/4 + 3n^{2/3}$ hat. Damit kann die Mehrheit der jeweils führenden Clique unter $\overline{F_3}$ nicht mehr auf einen Wert kleiner als $n/4 + n^{2/3}/2$ absinken.

Eine zu kleine Mehrheit kann also nur mit der jeweils zurück liegenden Clique erreicht werden. Wir unterscheiden zwei Fälle.

Fall 1: Die zurück liegende Clique hat eine Mehrheit von mindestens $n/4 + 2n^{2/3}$.

Dieser Fall tritt mit x^{**} direkt am Anfang von Phase 2 ein. Falls die zurück liegende Clique eine Mehrheit von mindestens $n/4+2n^{2/3}$ hat, kann F_4 gegeben $\overline{F_3}$ im nächsten Schritt nicht eintreten, da wesentlich mehr als $n^{1/3}$ Bits in einem Schritt flippen müssten und dies $\overline{F_3}$ widerspricht.

Fall 2: Die zurück liegende Clique hat eine Mehrheit von weniger als $n/4 + 2n^{2/3}$.

O. B. d. A. sei C_1 die zurück liegende Clique. Mit x bezeichnen wir den aktuellen Suchpunkt des (1+1) EA. Es kann passieren, dass die Mehrheit $\text{maj}(x, C_1)$ der zurück liegenden Clique irgendwann unter den Wert $n/4 + 2n^{2/3}$ sinkt. Zum ersten Zeitpunkt, an dem der Wert unterschritten wird und Fall 2 eintritt, ist die Mehrheit allerdings immerhin noch mindestens $\text{maj}(x, C_1) \geq n/4 + 2n^{2/3} - n^{1/3}$, da die Mehrheit unter $\overline{F_3}$ in einem Schritt nicht um mehr als $n^{1/3}$ sinken kann.

Immer dann, wenn Fall 2 eintritt und die Mehrheit der zurück liegenden Clique unter den Wert $n/4 + 2n^{2/3}$ sinkt, wenden wir Lemma 9 mit den Parametern $\alpha(x) := \text{maj}(x, C_1)$ und $\varepsilon := 2/3$ an auf den Random Walk der Mehrheit der zurück liegenden Clique C_1 in relevanten Schritten. Damit zeigen wir, dass die Mehrheit mit hoher Wahrscheinlichkeit nicht um weitere $n^{2/3}/2$ Werte absinkt und somit F_4 mit hoher Wahrscheinlichkeit nicht eintritt.

Zu beachten ist, dass wir Lemma 9 nur für eine Zeitspanne anwenden, die wir in Fall 2 zubringen. Falls zwischendurch Fall 1 eintritt, starten wir beim nächsten Auftreten von Fall 2 eine erneute Anwendung von Lemma 9.

Wir zeigen zunächst, dass in Fall 2 alle nötigen Voraussetzungen für Lemma 9 erfüllt sind. Dabei sei angemerkt, dass während einer Anwendung von Lemma 9 unter $\overline{F_3}$ die zurück liegende Clique nicht wechseln kann, da sich die Mehrheiten beider Cliquen um mindestens $n^{2/3}$ unterscheiden.

Die erste Bedingung von Lemma 9 ist, dass sich der α-Wert nicht in einem Schritt um mindestens $n^{\varepsilon/2}$ verringern kann. Diese Bedingung wird durch $\overline{F_3}$ erfüllt, da mit $\varepsilon = 2/3$ gilt: $n^{\varepsilon/2} = n^{1/3}$ und sich α daher nur um weniger als $n^{1/3}$ verringern kann.

Die zweite Bedingung von Lemma 9 besagt, dass die erwartete Drift des Random Walks durch eine positive Konstante nach unten beschränkt ist. In den Vorbetrachtungen haben wir bereits gezeigt, dass die erwartete Drift der Mehrheit $\text{maj}(x, C_1)$ in relevanten Schritten mindestens $3/200 - O(1/n)$ ist, auch wenn in einem Schritt höchstens $n^{1/3}$ Bits flippen können. Die Voraussetzungen für Lemma 8 sind hierbei erfüllt, da während der gesamten Anwendung

für ausreichend große n stets die Ungleichungen $\text{maj}(x, C_1) \leq \text{maj}(x, C_2)$ und $\text{maj}(x, C_1) < n/4 + 2n^{2/3} \leq (13/50)n$ gelten.

Für ausreichend große n können wir $3/200 - O(1/n)$ durch eine positive Konstante nach unten abschätzen. Daher ist auch die zweite Bedingung von Lemma 9 erfüllt und F_4 kann für einen beliebigen zusammenhängenden Zeitraum in Fall 2 nur dann eintreten, wenn sich die Mehrheit von C_1 um mindestens $n^{2/3}/2$ verringert.

Insgesamt tritt F_4 in Phase 2 nicht ein, wenn sich in allen Anwendungen von Lemma 9 in Fall 2 die Mehrheit der zurück liegenden Clique nicht um mindestens $n^{2/3}/2$ verringert.

Wie oft müssen wir Lemma 9 anwenden? Eine grobe Abschätzung reicht hier aus: In maximal $n^2 \log n$ Schritten müssen wir das Lemma höchstens $n^2 \log n$ Mal anwenden. Jede Anwendung des Lemmas birgt eine Wahrscheinlichkeit von höchstens $n^{-\Omega(n^{1/3})}$, dass F_4 eintritt. Wenn wir die Wahrscheinlichkeiten für alle Anwendungen addieren, erhalten wir

$$\text{Prob}(F_4 \mid \overline{F_3}) \leq n^2 \log n \cdot n^{-\Omega(n^{1/3})} = o(1).$$

Insgesamt folgt in Phase 2 eine Fehlerwahrscheinlichkeit von

$$\text{Prob}(F_3 \cup F_4) = \text{Prob}(F_3) + \text{Prob}(F_4 \mid \overline{F_3}) = o(1).$$

Es fehlt noch der Nachweis, dass das Zeitlimit von Phase 2 mit hoher Wahrscheinlichkeit eingehalten wird. Dazu teilen wir die Zeitspanne von $n^2 \log n$ Schritten in $n/16$ „Epochen“ ein, die jeweils eine Länge von $16n \log n$ Schritten haben und in denen wir dem Algorithmus jeweils die Chance geben, ein lokales (globales) Optimum zu erreichen. Aussage (i) liefert eine obere Schranke $8n \log n$ für die erwartete Zeit, bis ein lokales (globales) Optimum erreicht ist, unabhängig vom Startpunkt. Die Wahrscheinlichkeit, in einer Epoche von $16n \log n$ Schritten kein Optimum zu erreichen, ist daher nach Markoff höchstens $1/2$. Die Wahrscheinlichkeit, in allen Epochen kein Optimum zu erreichen, ist daher höchstens $2^{-n/16}$. Also gilt

$$\text{Prob}(T_2 > n^2 \log n) \leq 2^{-n/16}.$$

Wir fassen nun die Fehlerwahrscheinlichkeiten beider Phasen und die Wahrscheinlichkeiten, die Zeitlimits zu überschreiten, zusammen. Es gilt

$$\begin{aligned} & \text{Prob}(F_1 \cup \cdots \cup F_4 \cup \{T_1 > n^{3/4}\} \cup \{T_2 > n^2 \log n\}) \\ \leq\ & \text{Prob}(F_1 \cup F_2) + \text{Prob}(F_3 \cup F_4) + \text{Prob}(T_1 > n^{3/4}) + \text{Prob}(T_2 > n^2 \log n) \\ =\ & \frac{1}{2} + o(1). \end{aligned}$$

Mit der Gegenwahrscheinlichkeit $1/2 - o(1)$ ist der Lauf damit typisch und wir erreichen ein lokales Optimum mit verschieden gefärbten Cliquen, woraus Aussage (ii) folgt. □

Trotz der hohen Komplexität des Beweises lässt sich die Grundidee von Wegener und Witt [22] wiedererkennen. Die Länge von Phase 1 ist so gewählt, dass wir zum einen einen ausreichend großen Abstand zu alternativen Optima gewinnen. Zum anderen ist es aber auch noch möglich, die Wahrscheinlichkeit abzuschätzen, mit der wir uns einem lokalen Optimum annähern, da nach einer sublinearen Zahl von Schritten die Mehrheitsfarben mit hoher Wahrscheinlichkeit noch unabhängig voneinander sind.

Das Herzstück der Analyse von Phase 2 ist die Aussage, dass es unwahrscheinlich ist, bei einer hohen erwarteten Drift „gegen den Strom zu schwimmen“ und die Mehrheit gefährlich stark zu senken. Da sich die untere Schranke für die erwartete Drift nur für kleine Mehrheiten nachweisen lässt, können wir dieses Argument nur für kleine Mehrheiten anwenden. Der Beweis zeigt aber, dass diese Strategie hier ausreicht, um die gewünschte Aussage zu zeigen, da ohnehin nur kleine Mehrheiten Schwierigkeiten bereiten können. Außerdem stellt die Beschränkung auf kleine Mehrheiten in Fall 2 der Fallunterscheidung sicher, dass die zurück liegende Clique nicht wechseln kann, da die Differenz der Mehrheiten größer als die Zahl maximal flippender Bits ist.

Die Komplexität des Beweises rührt daher, dass wir viele Einzelaussagen miteinander verknüpfen, um die gewünschten Resultate zu zeigen. Außerdem sind einige Sonderfälle zu beachten wie die Existenz pathologischer Schritte, der minimale Einfluss des Ereignisses F_3 auf die erwartete Drift und die Möglichkeit, dass die führende Clique wechselt. Dies verkompliziert den Beweis und macht eine sehr sorgfältige Argumentation nötig.

Kapitel 4

Das Ising-Modell auf Bäumen

In diesem Kapitel wollen wir uns mit dem Ising-Modell auf Bäumen beschäftigen, wobei wir uns hauptsächlich auf vollständige binäre Bäume beschränken werden.

Zuerst definieren wir in Abschnitt 4.1 die Graphklasse der vollständigen binären Bäume und führen einige gebräuchliche und verbreitete Begriffe zu Bäumen ein. Anschließend betrachten wir die Fitnesslandschaft der Ising-Funktion auf vollständigen binären Bäumen und zeigen, dass es exponentiell viele lokale Optima gibt – Suchpunkte, bei denen alle Hammingnachbarn eine echt kleinere Fitness haben.

In Abschnitt 4.2 analysieren wir mutationsbasierte Algorithmen auf vollständigen binären Bäumen, nämlich die randomisierte lokale Suche und den (1+1) EA. Dabei wird sich zeigen, dass sich beide Algorithmen auf der Fitnessfunktion sehr schwer tun. Während die randomisierte lokale Suche nur mit exponentiell kleiner Wahrscheinlichkeit zu einem globalen Optimum gelangt, hat der (1+1) EA eine exponentielle erwartete Optimierungszeit.

In Abschnitt 4.3 analysieren wir Algorithmen mit Rekombination. Als Rekombinationsoperatoren verwenden wir *Zwei-Punkt-Kreuzungen*, einen üblichen Operator, den wir in Abschnitt 4.3 genauer definieren werden. Da für den Einsatz von Rekombination die Diversität der Population essenziell ist, werden die Algorithmen zusätzlich diversitätserhaltene Maßnahmen beinhalten.

Als erste naive Annäherung an eine sinnvolle Methode der Diversitätserhaltung diskutieren wir in Abschnitt 4.3.1 zunächst einen speziellen genetischen Algorithmus, den so genannten *(1+1) GIGA*, der stets eine Population aus einem Suchpunkt x und seinem bitweisen Komplement $\overline{x}$ verwaltet. Wir werden zeigen, dass der Algorithmus eine polynomielle erwartete Laufzeit hat.

Dieses Ergebnis überrascht nicht, da der Algorithmus durch die spezielle Form der Populationsverwaltung speziell auf die Belange der Ising-Funktion zugeschnitten ist und somit einen unfairen Vorteil gegenüber allgemeinen Varianten evolutionärer Algorithmen hat. Daher stellen wir in Abschnitt 4.3.2 einen anderen genetischen Algorithmus vor, der eine Standardmethode verwendet, um Diversität zu erhalten, nämlich das so genannte *Fitness Sharing*, das bereits in Abschnitt 1.2.1 erläutert

wurde. Überraschenderweise lässt sich auch hier eine polynomielle obere Schranke für die erwartete Optimierungszeit zeigen, die nur wenig schlechter als die obere Schranke für den (1+1) GIGA ist.

4.1 Einführung

4.1.1 Definitionen und Codierung

Definition allgemeiner Bäume

Um Bäume zu definieren, verwenden wir eine rekursive Definition, aus der der hierarchische Aufbau von Bäumen deutlich wird. Informell ausgedrückt besagt die folgende Definition: Ein Baum ist entweder leer oder er enthält eine Wurzel $r \in V$, dessen Nachbarn wiederum Wurzeln von Bäumen sind (bzw. als solche dargestellt werden können).

Definition 9. *Ein Graph $G = (V, E)$ ist ein* Baum, *wenn $V = \emptyset$ oder wenn folgendes gilt. Es gibt einen Knoten $r \in V$ und eine Partition der Menge $V \setminus \{r\}$ in nicht leere Knotenmengen $V_1, \ldots, V_k$ für ein $k \geq 0$. Wenn G_i der Subgraph ist, der durch V_i induziert wird, so ist in jeder Menge V_i genau ein Knoten zu r adjazent und alle Subgraphen $G_1, \ldots, G_k$ sind wiederum Bäume.*

Den Knoten r bezeichnen wir als *Wurzel* des Baums. Dabei ist zu beachten, dass die Wurzel im Allgemeinen nicht eindeutig ist, sondern frei gewählt werden kann. Wenn r als Wurzel festgelegt wird, bezeichnen wir die Subgraphen $G_1, \ldots, G_k$ als *Teilbäume*. Ein Teilbaum, dessen Wurzel ein Knoten $v \in V$ bildet, wird auch mit $T(v)$ notiert. Knoten mit Grad 1 bezeichnen wir als *Blätter*, alle anderen Knoten werden als *innere Knoten* bezeichnet.

Zur Visualisierung von Bäumen wählen wir folgende Darstellung. Wir stellen uns Bäume von oben nach unten gerichtet vor, wobei oben die Wurzel steht und die Teilbäume unten in Blättern enden. Zusätzlich teilen wir die Knoten in *Ebenen* ein: Ebene i enthält alle Knoten v, die einen Abstand von i zur Wurzel r des Baums haben. Unter dem Abstand verstehen wir dabei die Zahl der Kanten auf dem Pfad zwischen v und r. (Da alle Teilbäume eines Knotens paarweise disjunkt sind, gibt es in Bäumen stets genau einen Pfad zwischen zwei Knoten und der Pfad zwischen v und r ist eindeutig.) Ein Beispiel für eine solche Darstellung zeigt Abbildung 4.1.

In Abbildung 4.1 sind neben der Wurzel r drei Knoten u, v, w eingezeichnet. Wenn es eine Kante $\{u, v\}$ gibt und u auf der höheren (d. h. kleineren) Ebene liegt, bezeichnen wir u als *Elter* von v und v als *Kind* von u. Zwei verschiedene Teilbäume $T(v) \neq T(w)$, dessen Wurzeln den gleichen Elter haben, bezeichnen wir als *benachbarte Teilbäume*.

Unter der *Tiefe* eines (Teil-)Baums verstehen wir die maximale Länge eines Pfades von der Wurzel zu einem Blatt, gemessen in der Zahl der Kanten. Im Baum aus

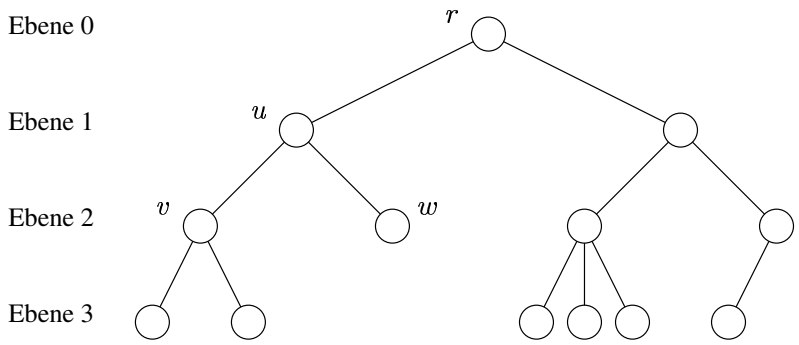


Abbildung 4.1: Ein Beispiel für einen allgemeinen Baum und die Einteilung in Ebenen.

Abbildung 4.1 hat also der Gesamtbaum $G = T(r)$ die Tiefe 3, der Teilbaum $T(u)$ hat die Tiefe 2 und der Teilbaum $T(w)$ hat die Tiefe 0. Alternativ sprechen wir manchmal auch von der *Höhe* eines Teilbaums.

Die Zahl der Kanten beträgt in jedem Baum genau $n - 1$, da alle Knoten v mit Ausnahme der Wurzel eine Kante zum jeweiligen Elter besitzen und wir alle Kanten genau einmal zählen, wenn wir sie dem jeweiligen Kind v zuordnen. Der maximale Fitnesswert beim Ising-Modell auf beliebigen Bäumen ist also stets $n - 1$.

Der minimale Fitnesswert ist 0, da Bäume bipartite Graphen sind. Dies lässt sich leicht einsehen, wenn wir die Knoten ebenenweise durchlaufen und die Knoten auf geraden Ebenen (gemeint sind Ebenen mit gerader Nummerierung) zur einen Menge zählen und die Knoten auf ungeraden Ebenen zur anderen Menge. Wenn wir also die Ebenen abwechselnd mit 0 und 1 färben, können wir alle Kanten zweifarbig färben und eine Fitness von 0 erreichen.

Vollständige binäre Bäume

Das Beispiel aus Abbildung 4.1 deutet an, dass allgemeine Bäume sehr unterschiedlich aussehen können. Zum einen kann es lange Pfade geben, die sich nicht weiter verzweigen, was zu einer sehr hohen Tiefe führt. Zum anderen können Knoten sehr viele Kinder haben, was zu einer sehr kleinen Tiefe führen kann. In Extremfällen besteht ein Baum aus einer linearen Kette von Knoten oder aus einem sternförmigen Graphen, bei dem alle Knoten außer der Wurzel Blätter sind, die direkt mit der Wurzel verbunden sind.

In dieser Arbeit werden wir uns auf spezielle Bäume konzentrieren, die zum einen vielleicht die „typischsten“ Bäume darstellen und zum anderen starke Symmetrieeigenschaften haben, die eine Analyse der Ising-Funktion wesentlich vereinfachen.

Definition 10. *Ein Baum $G = (V, E)$ mit Wurzel $r \in V$ heißt* binärer Baum, *wenn G ein Baum ist und jeder innere Knoten in V genau zwei Kinder hat.*

Ein binärer Baum $G = (V, E)$ mit Wurzel $r \in V$ heißt zudem vollständig, *wenn alle Blätter auf der gleichen Ebene liegen.*

Ein Beispiel für einen vollständigen binären Baum ist in Abbildung 4.2 dargestellt. Es sei angemerkt, dass die Wurzel in einem vollständigen binären Baum eindeutig bestimmt ist.

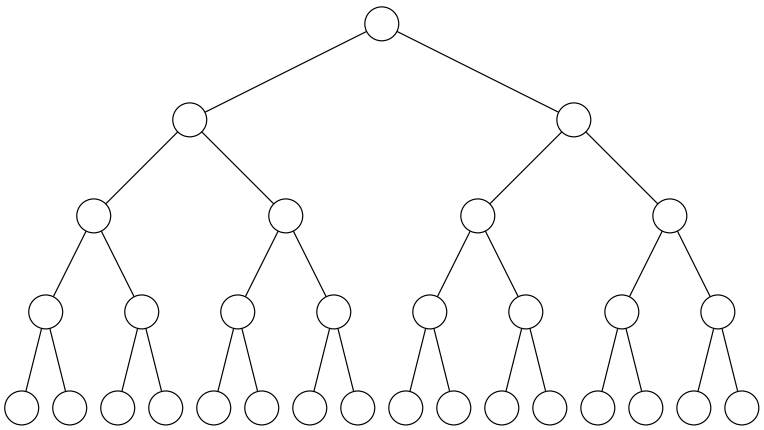


Abbildung 4.2: Ein Beispiel für einen vollständigen binären Baum der Tiefe 4.

Die Bedingung, dass jeder innere Knoten genau zwei Kinder hat, sorgt dafür, dass alle inneren Knoten mit Ausnahme der Wurzel den gleichen Grad haben; sie sind zu genau 3 anderen Knoten adjazent, zum jeweiligen Elter und zu beiden Kindern. Lediglich die Wurzel des Gesamtbaums hat keinen Elter und ist daher nur zu zwei Knoten adjazent. Blätter haben per Definition Grad 1 und sind somit nur zu ihrem Elter adjazent.

Der Grad eines Knotens spielt gerade beim Ising-Modell eine große Rolle, da die Fitness und die lokale Fitnessveränderung an einem Knoten v stark von der Zahl inzidenter Kanten abhängt. Außerdem macht es einen großen Unterschied, ob der Grad eines Knotens gerade ist oder ungerade: Wenn der Grad eines Knotens v ungerade ist, gibt es unter den Nachbarn von v, also unter den zu v adjazenten Knoten, stets eine eindeutige Mehrheitsfarbe. Wenn der Grad von v jedoch gerade ist, kann es bei einer passenden Färbung x vorkommen, dass genau so viele Nachbarn von v in x 0-gefärbt wie 1-gefärbt sind. Wenn wir wie in Abschnitt 2.2 mit $\deg_x^+(v)$ $[\deg_x^-(v)]$ die Zahl der in x einfarbigen [zweifarbigen] und zu v inzidenten Kanten bezeichnen, gilt daher $\deg_x^+(v) = \deg_x^-(v)$. Die lokale Fitnessveränderung $\ell(x, x_v)$, also die Veränderung der Fitness, wenn in x nur das dem Knoten v zugeordnete Bit x_v flippt, beträgt dann

$$\ell(x, x_v) = \deg_x^-(v) - \deg_x^+(v) = 0.$$

Da hier Hammingnachbarn die gleiche Fitness haben, führen Knoten mit geradem Grad zu Plateaus in der Fitnesslandschaft. Auf binären Bäumen hat nur die Wurzel einen geraden Grad, so dass hier nur Plateaus mit 2 Suchpunkten entstehen können (falls beide Kinder der Wurzel verschieden gefärbt sind); alle 1-Bit-Mutationen, die einen anderen Knoten $v \neq r$ treffen, verändern die Fitness in jedem Fall um einen von Null verschiedenen Wert.

Die zweite Bedingung, dass alle Blätter auf einer Ebene liegen sollen, garantiert uns, dass alle Teilbäume, deren Wurzeln auf der gleichen Ebene liegen, die gleiche Tiefe und die gleiche Zahl an Knoten haben. Die Symmetrie dieser Teilbäume wird uns in späteren Analysen von Nutzen sein. Allerdings hat die Bedingung auch zur Folge, dass es nur wenige vollständige binäre Bäume gibt, denn nicht für jede Knotenzahl n lässt sich ein vollständiger binärer Baum konstruieren.

Eine mögliche Alternative wäre, balancierte Bäume zu definieren, bei denen alle Blätter auf den beiden untersten Ebenen liegen, so dass alle Teilbäume, deren Wurzeln auf der gleichen Ebene liegen, zumindest ähnliche Tiefen haben. Allerdings ist unklar, wie die Blätter auf der untersten Ebene angeordnet werden sollen, wenn die Ebene nicht vollständig ausgefüllt werden kann. Hier können sich bei unterschiedlichen Verteilungen der Blätter unterschiedliche Effekte ergeben, die schwierig zu berücksichtigen sind. Daher werden wir uns von vorn herein auf vollständige binäre Bäume beschränken.

Wie viele Knoten enthält ein vollständiger binärer Baum mit Tiefe d? Auf Ebene i befinden sich 2^i Knoten, also beträgt die Zahl n der Knoten

$$n = \sum_{i=0}^{d} 2^i = 2^{d+1} - 1.$$

Die Zahl der Knoten ist also stets um 1 kleiner als eine Zweierpotenz, entsprechend selten sind vollständige binäre Bäume unter allen möglichen Suchraumdimensionen zu finden.

Wenn wir die Gleichung nach d auflösen, erhalten wir

$$\begin{array}{rrcl} & 2^{d+1} - 1 & = & n \\ \Leftrightarrow & 2^{d+1} & = & n+1 \\ \Leftrightarrow & d+1 & = & \log(n+1) \\ \Leftrightarrow & d & = & \log(n+1) - 1 \end{array}$$

Die Tiefe eines Baums mit n Knoten beträgt also $\log(n+1) - 1$. Dieser Term wird uns noch häufiger begegnen.

Ein binärer Baum mit n Knoten und Tiefe d enthält $2^d = 2^{\log(n+1)-1} = (n+1)/2$ Blätter. Allgemein lässt sich die Zahl der Knoten auf Ebene i einerseits durch 2^i ausdrücken und andererseits durch

$$2^i = 2^d \cdot 2^{i-d} = 2^{\log(n+1)-1} \cdot 2^{i-d} = (n+1) \cdot 2^{i-d-1}.$$

Aus letzterer Darstellung folgt: Die Zahl verschiedener Teilbäume mit Tiefe k ist gleich der Zahl der Knoten auf Ebene $i := d - k$ und damit $2^{d-k} = (n+1) \cdot 2^{-k-1}$. Auch auf diese Terme werden wir später zurück greifen.

Codierung

Wir planen, im Laufe dieses Kapitels Rekombinationsoperatoren auf Suchpunkte anwenden. Anders als bei Cliquen, bei denen wir nur mit Mutation gearbeitet haben,

müssen wir daher für Bäume festlegen, wie die Bits, die die n Knoten repräsentieren, im Bitstring angeordnet werden sollen.

Diese Codierung sollte „sinnvoll“ gewählt werden und die Charakteristika des Problems widerspiegeln. Da Bäume hierarchisch aufgebaut und rekursiv aus Teilbäumen zusammengesetzt sind, macht es Sinn, die Codierung so zu wählen, dass die Knoten eines Teilbaums im Bitstring eine zusammenhängende Teilfolge bilden. Eine Möglichkeit, die Knoten auf diese Art im Bitstring zu repräsentieren ist, die Knoten gemäß eines so genannten *Preorder-Durchlaufs* zu nummerieren.

Definition 11. *Gegeben ein Baum $G = (V, E)$ mit Wurzel r und $n := |V|$ Knoten. Ein Preorder-Durchlauf durch G erzeugt einen String* $\mathrm{pre}(r) = (\mathrm{pre}_1, \ldots, \mathrm{pre}_n)$ *mit* $\mathrm{pre}_i \in V$ *von Knoten nach folgendem rekursiven Schema.*

Falls $v_1, \ldots, v_k$ für $k \geq 0$ die Kinder eines Knotens $v \in V$ sind, sei

$$\mathrm{pre}(v) := v \circ \mathrm{pre}(v_1) \circ \cdots \circ \mathrm{pre}(v_k).$$

Für jeden Knoten v geben wir also zuerst den Knoten v aus und starten dann rekursive Aufrufe für alle Kinder von v. Aus der rekursiven Definition wird auch deutlich, dass alle Knoten eines beliebigen Teilbaums nacheinander ausgegeben werden und somit in der Folge $\mathrm{pre}(r) = (\mathrm{pre}_1, \ldots, \mathrm{pre}_n)$ eine zusammenhängende Teilfolge bilden. Ein Beispiel für die *Preorder-Nummerierung* der Knoten, sprich ihre Position in der Ausgabe, ist in Abbildung 4.3 dargestellt.

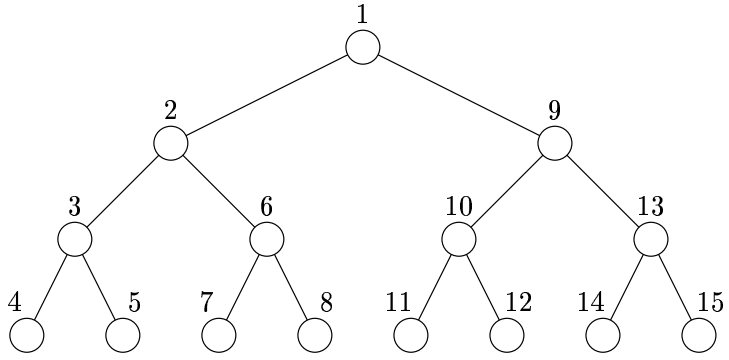


Abbildung 4.3: Eine Preorder-Nummerierung eines vollständigen binären Baumes der Tiefe 3. Die Nummern bezeichnen die Position des Knotens in der Ausgabe $\mathrm{pre}(r)$.

Wenn $x_1, \ldots, x_n$ die Bits des Bitstrings sind, ordnen wir das Bit x_i dem Knoten pre_i zu, der also in der Ausgabe des Preorder-Durchlaufs $\mathrm{pre}(r)$ an Position i steht.

Als einfachere Notation werden wir im Folgenden alternativ den Knoten direkt angeben, indem wir von x_v sprechen als dem Bit, das dem Knoten v zugeordnet ist.

4.1.2 Die Fitnesslandschaft des Ising-Modells auf Bäumen

Wie bereits angekündigt, wollen wir uns in diesem Abschnitt mit der Fitnesslandschaft der Ising-Funktion auf vollständigen binären Bäumen befassen. Zur einfache-

ren Sprechweise definieren wir folgende Bezeichnungen.

Definition 12. *Gegeben ein Graph $G = (V, E)$ mit $n := |V|$ Knoten und ein Suchpunkt $x \in \{0,1\}^n$. Einen Knoten $v \in V$ bezeichnen wir als*

- (lokal) gut gefärbt in x, *wenn* $\deg_x^+(v) > \deg_x^-(v)$,
- (lokal) schlecht gefärbt in x, *wenn* $\deg_x^+(v) < \deg_x^-(v)$
- *und als* (lokal) neutral gefärbt in x, *wenn* $\deg_x^+(v) = \deg_x^-(v)$.

Wenn der Bezug zum Suchpunkt x aus dem Kontext ersichtlich ist, erlauben wir uns, den Suchpunkt gegebenenfalls wegzulassen.

Da für eine neutrale Färbung der Grad von v gerade sein muss, kommen für alle Knoten außer der Wurzel nur gute und schlechte Färbungen in Frage. Wenn alle Knoten lokal gut gefärbt sind, ist ein lokales Optimum erreicht, da jede 1-Bit-Mutation die Fitness senkt. Umgekehrt gilt: Gibt es eine 1-Bit-Mutation, die die Fitness nicht senkt, kann der geflippte Knoten nicht gut gefärbt gewesen sein. Also ist eine Färbung genau dann lokal optimal, wenn alle Knoten lokal gut gefärbt sind.

In einer Färbung, in der alle Knoten lokal gut gefärbt sind, gilt: Die Wurzel und alle Blätter müssen ausschließlich zu einfarbigen Kanten inzident sein und alle inneren Knoten mit Ausnahme der Wurzel dürfen nur zu höchstens einer zweifarbigen Kante inzident sein. Andererseits sind die Bedingungen an die inneren Knoten jedoch so schwach, dass sich hier große Freiräume ergeben, wie lokale Optima aussehen können. Ein Beispiel für eine lokal optimale Färbung ist in Abbildung 4.4 dargestellt.

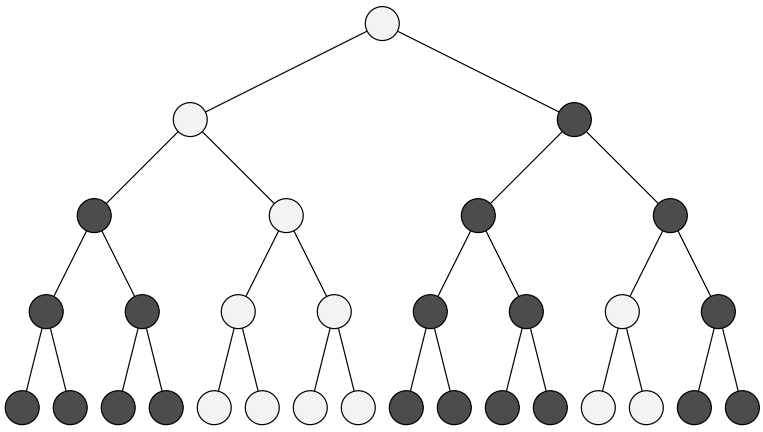


Abbildung 4.4: Ein Beispiel für eine lokal optimale Färbung eines vollständigen binären Baums der Tiefe 4 mit den zwei Farben „hell“ und „dunkel“.

Es ist plausibel, dass es exponentiell viele Färbungen gibt, bei denen alle Knoten gut gefärbt sind. Um diese Zahl zu bestimmen, werden wir nun eine Rekursionsformel aufstellen, die die genaue Zahl dieser Färbungen und damit die genaue Zahl lokaler Optima beschreibt.

Lemma 10. *Gegeben ein vollständiger binärer Baum $G = (V, E)$ mit $n := |V|$ Knoten und Tiefe $d := \log(n+1) - 1$. Für $i \geq 1$ definieren wir Werte A_i durch $A_1 := 1, A_2 := 3$ und für $i \geq 3$ durch die Rekursionsgleichung*

$$A_i := A_{i-1}^2 + 2A_{i-1}A_{i-2}^2.$$

Dann gilt: Die Zahl lokaler Optima auf Ising_G *beträgt* $2A_{d-1}^2$.

Beweis. Wir betrachten im Folgenden Teilbäume der Höhe i, d. h. Teilbäume, dessen Wurzeln im Gesamtbaum G auf Ebene $d-i$ liegen. Für einen solchen Teilbaum $T(v)$ mit Wurzel v zählen wir die Zahl verschiedener Teilfärbungen, bei denen alle Knoten in $T(v)$ gut gefärbt sind. Solche Teilfärbungen bezeichnen wir auch als *stabil*. Dabei schränken wir die Teilfärbungen ein, indem wir die Farbe der Wurzel v des Teilbaums vorgeben, o. B. d. A. die Farbe 0. Wir zählen also nur die stabilen Teilfärbungen der Knoten in $T(v)$, bei denen v mit 0 gefärbt ist.

Wenn der Elter von v ebenfalls mit 0 gefärbt ist, darf eine Kante von v zu einem Kind zweifarbig sein. Wenn der Elter von v mit 1 gefärbt ist oder wenn v die Wurzel des Gesamtbaums G ist, müssen jedoch die Kanten zu beiden Kindern einfarbig sein, da ansonsten v nicht gut gefärbt wäre.

Für die Bestimmung der Zahl stabiler Teilfärbungen ist also die Farbe des Elters von v entscheidend (bzw. die Frage, ob v Wurzel des Gesamtbaums ist). Daher unterscheiden wir zwei Fälle und berechnen die Zahl stabiler Teilfärbungen in beiden Fällen separat.

- A_i^+ sei die Zahl stabiler Teilfärbungen eines Teilbaums $T(v)$ mit 0-gefärbter Wurzel v und Höhe i, bei denen v einen 0-gefärbten Elter hat.
- A_i^- sei die Zahl stabiler Teilfärbungen eines Teilbaums $T(v)$ mit 0-gefärbter Wurzel v und Höhe i, bei denen v einen 1-gefärbten Elter hat oder v Wurzel des Gesamtbaums G ist.

Aufgrund der Bit-Flip-Symmetrie geben A_i^+ und A_i^- auch die Zahl stabiler Teilfärbungen an, bei denen die Wurzel v mit 1 gefärbt ist. Entscheidend ist nur, dass die Farbe von v fest vorgegeben ist.

Für Teilbäume $T(v)$ der Höhe 1 gilt, dass es bei einer 0-gefärbten Wurzel v nur eine stabile Teilfärbung gibt, wenn nämlich beide Blätter auch mit 0 gefärbt sind. Dies gilt unabhängig von der Färbung eines eventuellen Elters von v. Also gilt

$$A_1^+ = 1 \quad \text{und} \quad A_1^- = 1.$$

Für den Fall, dass $T(v)$ eine Höhe von $i \geq 2$ hat, die 0-gefärbte Wurzel v einen ebenfalls 0-gefärbten Elter hat und v_1, v_2 die beiden Kinder von v sind, führen wir nun die Zahl A_i^+ stabiler Teilfärbungen in $T(v)$ zurück auf die Zahlen stabiler Teilfärbungen in $T(v_1)$ und $T(v_2)$.

Wenn v einen 0-gefärbten Elter hat, darf eines der beiden Kinder von v in einer stabilen Teilfärbung 1-gefärbt sein, also eine andere Farbe haben als v. Angenommen,

v_1 ist 0-gefärbt und v_2 ist 1-gefärbt. Dann gibt es für $T(v_1)$ genau A^+_{i-1} stabile Teilfärbungen (bei vorgegebener Farbe der Wurzel v_1) und im Teilbaum $T(v_2)$ genau A^-_{i-1} stabile Teilfärbungen (bei vorgegebener Farbe der Wurzel v_2). Alternativ kann auch v_1 das 1-gefärbte Kind sein, so dass wir hier $2 \cdot A^+_{i-1} \cdot A^-_{i-1}$ stabile Teilfärbungen erhalten.

Andere stabile Teilfärbungen entstehen, wenn die Kinder von v beide 0-gefärbt sind. In diesem Fall gibt es $(A^+_{i-1})^2$ stabile Teilfärbungen für die Teilbäume $T(v_1)$ und $T(v_2)$. Insgesamt erhalten wir die Rekursionsformel

$$A^+_i = (A^+_{i-1})^2 + 2 \cdot A^+_{i-1} \cdot A^-_{i-1}.$$

Im Fall $i = 2$ gilt $A^+_2 = 1^2 + 2 \cdot 1 = 3$, die Startwerte für A^+_i sind also $A^+_1 = 1$ und $A^+_2 = 3$.

Nun zum Term A^-_i. Falls die 0-gefärbte Wurzel v keinen Elter hat oder der Elter 1-gefärbt ist, müssen die Kinder von v beide 0-gefärbt sein, damit v gut gefärbt ist. Es folgt

$$A^-_i = (A^+_{i-1})^2.$$

Wenn wir beide Rekursionsgleichungen ineinander einsetzen, erhalten wir für $i \geq 3$

$$A^+_i = (A^+_{i-1})^2 + 2 \cdot A^+_{i-1} \cdot (A^+_{i-2})^2.$$

Damit können wir nun die Zahl stabiler Färbungen des Gesamtbaums ausrechnen. Der Gesamtbaum hat Tiefe d. Bei vorgegebener Farbe der Wurzel gibt es A^-_d stabile Färbungen des Gesamtbaums. Da es zwei Möglichkeiten gibt, die Wurzel des Gesamtbaums zu färben, beträgt die Zahl stabiler Färbungen des Gesamtbaums und damit die Zahl lokaler Optima genau

$$2A^-_d = 2(A^+_{d-1})^2.$$

Wenn wir A^+_i in A_i umbenennen, folgt die Behauptung. □

Die Werte A_i werden durch die Rekursionsgleichung exakt beschrieben. Allerdings lassen sich die genauen Werte nur schwer ablesen, so dass wir an einer geschlossenen Form für die Zahl lokaler Optima interessiert sind.

Da die Rekursionsgleichung recht komplex ist, ist sie nur schwer in eine geschlossene Form zu überführen. Wir können dennoch geschlossene obere und untere Schranken für die Zahl lokaler Optima zeigen, indem wir die Rekursionsgleichung durch einfachere Rekursionsgleichungen ersetzen, die untere bzw. obere Schranken für die komplexe Rekursionsgleichung darstellen.

Theorem 13. *Gegeben ein vollständiger binärer Baum $G = (V, E)$ mit $n := |V|$ Knoten und Tiefe $d := \log(n+1) - 1$. Die Zahl lokaler Optima auf Ising_G ist beschränkt durch $\Omega(2^{0,264n})$ und $O(2^{0,343n})$.*

Beweis. Zunächst zeigen wir die obere Schranke. Für $i \geq 4$ gilt folgende einfache Abschätzung für A_{i-1}.

$$A_{i-1} \;=\; A_{i-2}^2 + 2A_{i-2} \cdot A_{i-3}^2 \;\geq\; A_{i-2}^2.$$

Wenn wir dies in die Rekursionsgleichung aus Lemma 10 einsetzen, erhalten wir für $i \geq 4$ die obere Schranke

$$\begin{aligned} A_i &= A_{i-1}^2 + 2A_{i-1} \cdot A_{i-2}^2 \\ &\leq A_{i-1}^2 + 2A_{i-1} \cdot A_{i-1} \\ &= 3A_{i-1}^2. \end{aligned}$$

Wir zeigen nun induktiv, dass für $i \geq 3$ und $0 \leq k \leq i-3$ gilt

$$A_i \leq 3^{2^k-1} A_{i-k}^{2^k}.$$

Für $k = 0$ erhalten wir

$$A_i \leq 3^{2^0-1} A_{i-0}^{2^0} = A_i.$$

Angenommen, die Behauptung gilt für k mit $1 \leq k \leq i-4$. Da mit dieser Voraussetzung $i - k \geq 4$, wenden wir die Abschätzung $A_{i-k} \leq 3A_{i-k-1}^2$ an:

$$\begin{aligned} A_i &\overset{\text{I.V.}}{\leq} 3^{2^k-1} A_{i-k}^{2^k} \\ &\leq 3^{2^k-1} (3A_{i-k-1}^2)^{2^k} \\ &= 3^{2^k-1} \cdot 3^{2^k} \cdot A_{i-k-1}^{2^{k+1}} \\ &= 3^{2^{k+1}-1} A_{i-(k+1)}^{2^{k+1}}. \end{aligned}$$

Also gilt die Behauptung auch für $k+1$.

Mit dieser Erkenntnis werden wir nun die vereinfachte Rekursionsgleichung in eine geschlossene Form überführen. Wir setzen $k := d-4 = \log(n+1)-5$. Nach Lemma 10 ist die Zahl lokaler Optima damit höchstens

$$\begin{aligned} 2A_{d-1}^2 &\leq 2\left(3^{2^k-1} A_{d-1-k}^{2^k}\right)^2 \\ &= 2\left(3^{2^k-1} A_3^{2^k}\right)^2. \end{aligned}$$

Für das Ende der Rekursion lässt sich leicht ausrechnen, dass $A_3 = 15$. Wenn wir dies und $k = \log(n+1) - 5$ einsetzen, erhalten wir

$$\begin{aligned} 2A_{d-1}^2 &\leq 2\left(3^{2^{\log(n+1)-5}-1} \cdot 15^{2^{\log(n+1)-5}}\right)^2 \\ &= 2\left(3^{(n+1)/32-1} \cdot 15^{(n+1)/32}\right)^2 \\ &= \frac{1}{2} \cdot 3^{(n+1)/16} \cdot 15^{(n+1)/16} \\ &= \frac{1}{2} \cdot 2^{(n+1)/16 \cdot (\log 3 + \log 15)} = O(2^{0{,}343n}). \end{aligned}$$

Die untere Schranke zeigen wir analog. Da $A_i \geq A_{i-1}^2$, gilt für $i \geq 3$ die einfache Abschätzung

$$\begin{aligned} A_i &= A_{i-1}^2 + 2A_{i-1}A_{i-2}^2 \\ &\geq A_{i-2}^4 + 2A_{i-2}^2 A_{i-2}^2 \\ &= 3A_{i-2}^4. \end{aligned}$$

Wir zeigen nun induktiv, dass für $i \geq 1$ und $0 \leq k < i/2$ gilt

$$A_i \geq 3^{\frac{2^{2k}-1}{3}} A_{i-2k}^{2^{2k}}.$$

Für $k = 0$ erhalten wir

$$A_i \geq 3^{\frac{2^0-1}{3}} A_i^{2^0} = A_i.$$

Angenommen, die Behauptung gilt für k mit $1 \leq k < i/2 - 1$. Aus dieser Voraussetzung folgt $i-2 > 2k \Rightarrow i-2k \geq 3$ und wir wenden die Abschätzung $A_{i-2k} \geq 3A_{i-2k-2}^4$ an:

$$\begin{aligned} A_i &\overset{\text{I.V.}}{\geq} 3^{\frac{2^{2k}-1}{3}} A_{i-2k}^{2^{2k}} \\ &\geq 3^{\frac{2^{2k}-1}{3}} \cdot \left(3A_{i-2k-2}^4\right)^{2^{2k}} \\ &= 3^{\frac{2^{2k}-1}{3}} \cdot 3^{2^{2k}} A_{i-2(k+1)}^{2^{2(k+1)}} \\ &= 3^{\frac{2^{2k}-1+3\cdot 2^{2k}}{3}} A_{i-2(k+1)}^{2^{2(k+1)}} \\ &= 3^{\frac{2^{2(k+1)}-1}{3}} A_{i-2(k+1)}^{2^{2(k+1)}} \end{aligned}$$

Also gilt die Behauptung auch für $k+1$.

Nun überführen wir auch diese vereinfachte Rekursionsgleichung in eine geschlossene Form. Falls d ungerade, setzen wir $k := (d-1)/2 - 1$ und erhalten nach Lemma 10 eine Zahl lokaler Optima von

$$\begin{aligned} 2A_{d-1}^2 &\geq 2\left(3^{\frac{2^{2k}-1}{3}} A_{d-1-2k}^{2^{2k}}\right)^2 \\ &= 2\left(3^{\frac{2^{2k}-1}{3}} A_2^{2^{2k}}\right)^2. \end{aligned}$$

Wir setzen $A_2 = 3$ und $2k = d - 3 = \log(n+1) - 4$ ein und erhalten

$$\begin{aligned} 2A_{d-1}^2 &\geq 2\left(3^{\frac{2^{\log(n+1)-4}-1}{3}} \cdot 3^{2^{\log(n+1)-4}}\right)^2 \\ &= 2\left(3^{\frac{(n+1)/16-1}{3}} \cdot 3^{(n+1)/16}\right)^2 \\ &= 2 \cdot 3^{-2/3}\left(3^{(n+1)/48} \cdot 3^{(n+1)/16}\right)^2 \\ &= 2 \cdot 3^{-2/3} \cdot 3^{(n+1)/6} \\ &= 2 \cdot 3^{-2/3} \cdot 2^{(n+1)/6 \cdot \log 3} = \Omega(2^{0{,}264n}). \end{aligned}$$

Falls d gerade, setzen wir $k := d/2 - 2$ und erhalten

$$\begin{aligned} 2A_{d-1}^2 &\geq 2\left(3^{\frac{2^{2k}-1}{3}} A_{d-1-2k}^{2^{2k}}\right)^2 \\ &= 2\left(3^{\frac{2^{2k}-1}{3}} A_3^{2^{2k}}\right)^2. \end{aligned}$$

Mit $A_3 = 15$ und $2k = d - 4 = \log(n+1) - 5$ erhalten wir

$$\begin{aligned} 2A_{d-1}^2 &\geq 2\left(3^{\frac{2^{\log(n+1)-5}-1}{3}} \cdot 15^{2^{\log(n+1)-5}}\right)^2 \\ &= 2\left(3^{\frac{(n+1)/32-1}{3}} \cdot 15^{(n+1)/32}\right)^2 \\ &= 2 \cdot 3^{-2/3}\left(3^{(n+1)/96} \cdot 15^{(n+1)/32}\right)^2 \\ &= 2 \cdot 3^{-2/3} \cdot 3^{(n+1)/48} \cdot 15^{(n+1)/16} \\ &= 2 \cdot 3^{-2/3} \cdot 2^{(n+1)\cdot(1/48\cdot\log 3 + 1/16\cdot\log 15)} = \Omega(2^{0{,}277n}). \end{aligned}$$

In beiden Fällen ist die Zahl lokaler Optima $\Omega(2^{0{,}264n})$, was die untere Schranke beweist. □

4.2 Mutationsbasierte Algorithmen

4.2.1 Die randomisierte lokale Suche auf Bäumen

Die Betrachtung der Fitnesslandschaft hat ergeben, dass es exponentiell viele lokale Optima gibt. Darunter sind nur zwei globale Optima, nämlich 0^n und 1^n, alle anderen lokalen Optima sind suboptimal. Daher ist zu vermuten, dass die randomisierte lokale Suche mit hoher Wahrscheinlichkeit in nicht optimalen lokalen Optima hängen bleibt und somit eine sehr kleine Wahrscheinlichkeit hat, ein globales Optimum zu finden.

Theorem 14. *Gegeben ein vollständiger binärer Baum $G = (V, E)$ mit $n := |V|$ Knoten. Die Wahrscheinlichkeit, dass die randomisierte lokale Suche auf* Ising_G *ein globales Optimum findet, ist* $2^{-\Theta(n)}$.

Beweis. Die untere Schranke $2^{-O(n)}$ ist trivial, da mit Wahrscheinlichkeit $2 \cdot 2^{-n}$ bei der zufälligen Initialisierung ein globales Optimum gewählt wird.

Eine obere Schranke $2^{-\Omega(n)}$ lässt sich ebenfalls aus der zufälligen Initialisierung herleiten, ohne näher auf den Lauf des Algorithmus einzugehen. Die bisherigen Erkenntnisse aus Theorem 13 reichen jedoch dafür noch nicht aus: Wenn wir darauf spekulieren, direkt in einem lokalen, nicht globalen Optimum zu starten, besagt Theorem 13 zwar, dass es exponentiell viele dieser lokalen Optima gibt; allerdings ist der Anteil lokaler Optima gemessen an der Größe 2^n des Suchraums exponentiell klein, so dass die randomisierte lokale Suche nur mit exponentiell kleiner Wahrscheinlichkeit direkt in einem lokalen Optimum startet.

Wir wählen daher einen anderen Weg und zeigen, dass wir bei der zufälligen Initialisierung mit Wahrscheinlichkeit $2^{-\Omega(n)}$ einen Suchpunkt erzeugen, von dem aus es keine Folge von 1-Bit-Mutationen gibt, die zu einem globalen Optimum führt. Wenn ein solcher Suchpunkt ausgewählt wird, kann die randomisierte lokale Suche also kein globales Optimum erreichen.

Sei $\mathcal{T}$ die Menge aller Teilbäume der Höhe 1 (d. h. Teilbäume, dessen Wurzeln auf der zweituntersten Ebene liegen). Wenn ein solcher Teilbaum einheitlich gefärbt ist, verringert jede 1-Bit-Mutation die Fitness, da dann bei Blättern die Kante zum Elter zweifarbig wird und bei einer Mutation des Elters beide Kanten zu den Kindern zweifarbig werden. Die randomisierte lokale Suche hat also keine Möglichkeit, einen einheitlich gefärbten Teilbaum aus $\mathcal{T}$ zu verändern.

Wenn es in einer Färbung x einen einheitlich 0-gefärbten Teilbaum $T_0 \in \mathcal{T}$ und einen einheitlich 1-gefärbten Teilbaum $T_1 \in \mathcal{T}$ gibt, kann die randomisierte lokale Suche ausgehend von x kein globales Optimum erreichen.

Um die Wahrscheinlichkeit für ein solches Ereignis zu berechnen, zeigen wir zunächst folgende Äquivalenz. Dabei notieren wir mit $x_T \in \{0,1\}^3$ die Teilfärbung eines Baums $T \in \mathcal{T}$ in der Färbung x. Für alle Suchpunkte x gilt dann: Genau dann, wenn es keine Teilbäume T_0, T_1 in $\mathcal{T}$ gibt, sind entweder alle Teilbäume in $\mathcal{T}$ nicht einheitlich 0-gefärbt oder alle Teilbäume in $\mathcal{T}$ nicht einheitlich 1-gefärbt. Formal:

$$(\nexists T_0, T_1 \in \mathcal{T}\colon x_{T_0} = 0^3, x_{T_1} = 1^3) \;\Leftrightarrow\; (\forall T \in \mathcal{T}\colon x_T \neq 0^3) \vee (\forall T \in \mathcal{T}\colon x_T \neq 1^3)$$

Wenn alle Teilbäume nicht einheitlich 0-gefärbt sind oder alle Teilbäume nicht einheitlich 1-gefärbt sind, kann es keine Teilbäume T_0 und T_1 geben. Daraus folgt die Richtung „$\Leftarrow$“. Die Richtung „$\Rightarrow$“ folgt mit der Kontraposition: Wenn die rechte Seite der Äquivalenz falsch ist, gibt es einen einheitlich 0-gefärbten Teilbaum und einen einheitlich 1-gefärbten Teilbaum und die linke Seite der Äquivalenz ist falsch.

Die Wahrscheinlichkeit, dass alle Teilbäume nicht einheitlich mit 0 gefärbt sind, ist gleich der Wahrscheinlichkeit, dass alle Teilbäume nicht einheitlich mit 1 gefärbt sind. Wir erhalten daher

$$\text{Prob}(\nexists T_0, T_1 \in \mathcal{T}\colon T_0 = 0^3, T_1 = 1^3) \leq 2 \cdot \text{Prob}(\forall T \in \mathcal{T}\colon T \neq 0^3).$$

Es gibt insgesamt 8 mögliche Färbungen für die drei Knoten eines Teilbaums $T \in \mathcal{T}$, von denen 7 nicht einer einheitlichen Färbung mit Farbe 0 entsprechen. Zudem gibt es $(n+1)/4$ Teilbäume in $\mathcal{T}$, da es $(n+1)/4$ Knoten auf der zweituntersten Ebene gibt. Es folgt

$$\begin{aligned}
\text{Prob}(\nexists T_0, T_1 \in \mathcal{T}\colon x_{T_0} = 0^3, x_{T_1} = 1^3) &\leq 2 \cdot \left(\frac{7}{8}\right)^{(n+1)/4} \\
&= 2 \cdot 2^{\log(7/8)(n+1)/4} \\
&= 2 \cdot 2^{-\log(8/7)(n+1)/4} \\
&\leq 2^{-0.048n+1}.
\end{aligned}$$

□

4.2.2 Die Laufzeit des (1+1) EA

Der (1+1) EA hat prinzipiell keine Probleme, suboptimale lokale Optima zu verlassen, da der (1+1) EA beispielsweise ganze Teilbäume in einem Schritt komplett flippen kann. Angenommen, in einem Schritt wird genau ein Teilbaum $T(v)$ komplett geflippt, $T(v) \subsetneq V$. Aufgrund der Bit-Flip-Symmetrie gilt dann, dass der Fitnessbeitrag aller Kanten innerhalb von $T(v)$ konstant bleibt, da von jeder Kante beide Endpunkte flippen. Alle Kanten, die nicht zu Knoten aus $T(v)$ inzident sind, werden nicht berührt und haben daher auch den gleichen Fitnessbeitrag wie zuvor.

Die einzige Kante, deren Fitnessbeitrag sich ändert, ist die Kante von v zum Elter von v. Falls diese Kante zweifarbig ist, ist sie im Nachkommen einfarbig und die Fitness erhöht sich um 1. Da ein solcher Teilbaum die Fitness verbessert, wenn er geflippt wird, bezeichnen wir ihn als *verbessernden Teilbaum*.

Definition 13. *Gegeben ein beliebiger Baum $G = (V, E)$ mit $n := |V|$ Knoten und eine Färbung $x \in \{0,1\}^n$. Ein Teilbaum $T(v)$ heißt* verbessernder Teilbaum *in x, wenn v einen Elter u besitzt und $x_v \neq x_u$.*

Wie üblich, erlauben wir uns, die Färbung gegebenenfalls wegzulassen, wenn sie aus dem Kontext ersichtlich wird. Die Färbung, die uns bereits aus Abbildung 4.4 bekannt ist, induziert drei verbessernde Teilbäume, die in Abbildung 4.5 dargestellt sind.

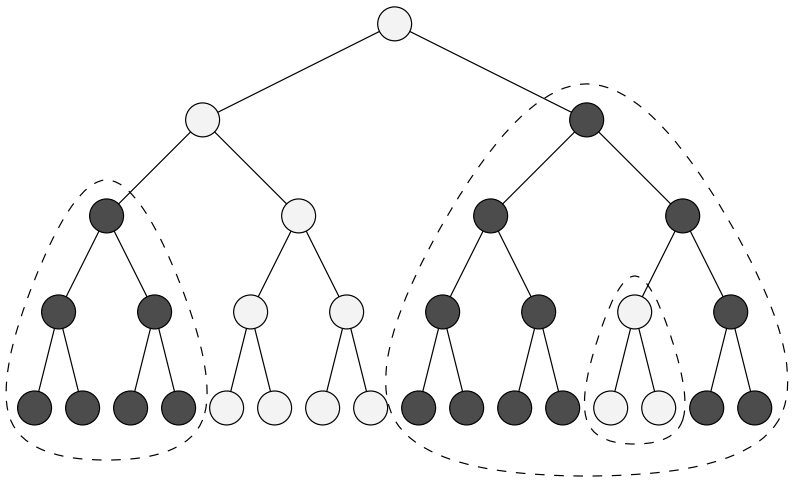


Abbildung 4.5: Ein Beispiel für eine Färbung x eines vollständigen binären Baums und die verbessernden Teilbäume in x.

Jeder verbessernde Teilbaum setzt eine zweifarbige Kante voraus und jede zweifarbige Kante induziert einen verbessernden Teilbaum. Es gibt also eine eineindeutige Beziehung zwischen verbessernden Teilbäumen und zweifarbigen Kanten.

Eine obere Schranke für die erwartete Optimierungszeit

Die Erkenntnis, dass Mutationen verbessernder Teilbäume die Fitness erhöhen, werden wir zum Beweis einer oberen Schranke für die erwartete Optimierungszeit verwenden. Eine Mutation, die einen verbessernden Teilbaum flippt, ist um so wahrscheinlicher, je kleiner der verbessernde Teilbaum ist. Daher konzentrieren wir uns im Folgenden auf verbessernde Teilbäume minimaler Tiefe unter allen verbessernden Teilbäumen der entsprechenden Färbung.

Zur Verkürzung der Schreibweise führen wir eine Funktion für die minimale Tiefe aller verbessernden Teilbäume ein.

Definition 14. *Gegeben ein vollständiger binärer Baum $G = (V, E)$ mit $n := |V|$ Knoten. Für $x \in \{0,1\}^n$ mit $f(x) < n - 1$ sei $\psi(x)$ die minimale Tiefe eines Teilbaums unter allen verbessernden Teilbäumen in x.*

Mit ψ haben wir ein Maß gefunden, mit dem sich die Wahrscheinlichkeit abschätzen lässt, dass ein komplett flippender Teilbaum die Fitness erhöht. Da ein Teilbaum der Tiefe $\psi(x)$ genau $2^{\psi(x)+1} - 1$ Knoten enthält, ist die Wahrscheinlichkeit, die Fitness durch eine solche Operation zu erhöhen, $\Omega(n^{-2^{\psi(x)+1}-1})$.

Allerdings ist es bei weitem keine notwendige Bedingung, dass ein verbessernder Teilbaum der Tiefe $\psi(x)$ flippt. Es kann Operationen geben, bei denen beispielsweise eine zusammenhängende Menge innerer Knoten flippt und die Fitness dadurch erhöht wird, ohne dass ein ganzer Teilbaum von der Mutation betroffen wird. (Wir werden später in Abbildung 4.7 Beispiele für solche Operationen kennen lernen.) Solche Mutationen sind aber nur schwer zu analysieren, da die Färbung großer Teile des Baums berücksichtigt werden muss.

Andere Operationen können den Wert von ψ ändern, ohne dass dies gleich zu einer Fitnesserhöhung führt. Angenommen, $T(v)$ ist in einem Suchpunkt x ein verbessernder Teilbaum der minimalen Tiefe $\psi(x)$ und v_1, v_2 sind Kinder von v. Nun kann eine Mutation, die aus x den Nachkommen x' erzeugt, beispielsweise genau die Knoten in $T(v_1) \cup \{v\}$ flippen. Dadurch wird die zweifarbige Kante von v zu seinem Elter u einfarbig gefärbt und die einfarbige Kante $\{v, v_2\}$ wird einfarbig gefärbt, wie in Abbildung 4.6 veranschaulicht. Alle anderen Kanten behalten ihren alten Fitnessbeitrag. Diese Operation lässt daher die Fitness konstant, aber mit $T(v_2)$ gibt es nun einen verbessernden Teilbaum mit kleinerer Tiefe; es gilt $\psi(x') = \psi(x) - 1$.

Aber auch die umgekehrte Richtung ist möglich. Angenommen, $T(v_2)$ mit Elter v und benachbartem Teilbaum $T(v_1)$ ist in x der einzige verbessernde Teilbaum mit Tiefe $\psi(x)$. Wenn die Kante von v zum Elter existiert und einfarbig ist und genau die Knoten $T(v_1) \cup \{v\}$ flippen, ist $T(v)$ im Nachkommen x' ein verbessernder Teilbaum minimaler Tiefe und es ist $\psi(x') = \psi(x) + 1$; die Fitness bleibt ebenfalls konstant. Auch diese Operation ist in Abbildung 4.6 zu sehen.

Der ψ-Wert des jeweils aktuellen Suchpunkts kann sich also über die Zeit verändern, auch ohne dass die Fitness erhöht wird.

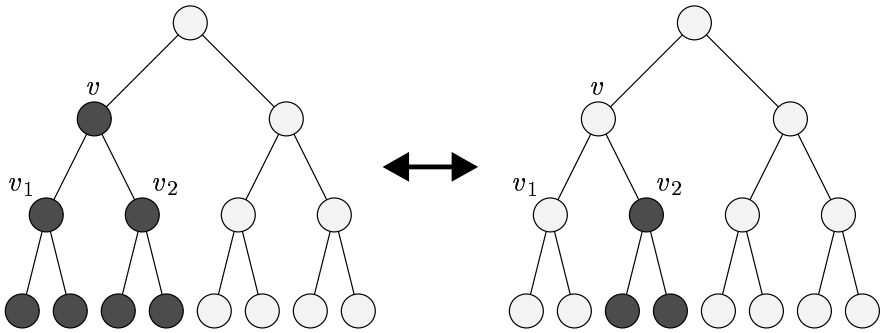


Abbildung 4.6: Ein Beispiel für zwei Operationen, durch die der aktuelle ψ-Wert ohne Fitnessgewinn verändert wird. Wird aus der linken Färbung durch Mutation der Knoten in $T(v_1) \cup \{v\}$ die rechte Färbung erzeugt, sinkt der ψ-Wert; wird aus der rechten Färbung durch die gleiche Mutation die linke Färbung erzeugt, steigt der ψ-Wert.

Die als Beispiel angeführten Mutationen zeigen außerdem, dass der (1+1) EA mit etwas Glück einen verbessernden Teilbaum auch in mehreren Schritten flippen kann. Statt einen verbessernden Teilbaum $T(v)$ auf einen Schlag zu flippen, ist es auch möglich, zuerst v und den Teilbaum eines Kindes zu flippen, um dann in einem späteren Schritt den Teilbaum des anderen Kindes zu flippen. Allerdings geht dieser Plan nur dann auf, wenn zwischendurch keine Mutationen auftreten, die uns einen Strich durch die Rechnung machen; z. B., indem die Tiefe des verbessernden Teilbaums zwischenzeitlich wieder erhöht wird.

Theorem 15. *Gegeben ein vollständiger binärer Baum $G = (V, E)$ mit $n := |V|$ Knoten und Tiefe $d := \log(n+1) - 1$. Die Laufzeit des (1+1) EA auf $f := \text{Ising}_G$ ist nach oben beschränkt durch $O(n^{(n+1)/4})$.*

Beweis. Wir verwenden die Methode der Fitnessschichten mit der kanonischen Einteilung in Schichten $A_i := \{x \mid f(x) = i\}$ und summieren die erwarteten Zeiten, bis alle Fitnessschichten verlassen werden.

Die Fitnessschichten $A_0, \ldots, A_{n-3}$ sind leicht zu verlassen. Sei $x \in A_i$ der aktuelle Suchpunkt in Fitnessschicht i mit $0 \leq i \leq n-3$. Da es mindestens zwei zweifarbige Kanten gibt, gilt folgendes. Entweder beide zweifarbigen Kanten sind zur Wurzel inzident oder eine zweifarbige Kante liegt auf einer niedrigeren Ebene. Sind beide Kanten zur Wurzel inzident, ist die Wahrscheinlichkeit, die aktuelle Fitnessschicht zu verlassen, mindestens $1/(en)$, da eine 1-Bit-Mutation der Wurzel die Fitness erhöht.

Liegt eine zweifarbige Kante auf einer niedrigeren Ebene, ist $\psi(x) \leq d-2$, da der untere Endpunkt einer zweifarbigen Kante mindestens auf Ebene 2 liegt. Sei $T(v)$ ein entsprechender verbessernder Teilbaum der Tiefe $\psi(x)$. Da $T(v)$ genau $2^{\psi(x)+1} - 1 \leq 2^{d-1} - 1 = 2^{\log(n+1)-2} - 1 = (n+1)/4 - 1$ Knoten enthält, ist die Wahrscheinlichkeit, dass $T(v)$ in einem Schritt komplett geflippt wird, $\Omega(n^{-((n+1)/4-1)})$.

In beiden Fällen ist also die Wahrscheinlichkeit einer Fitnesserhöhung nach unten

beschränkt durch $\Omega(n^{-((n+1)/4-1)})$ und die erwartete Zeit, bis wir die Fitnessschichten $A_0, \ldots, A_{n-3}$ verlassen haben, ist beschränkt durch

$$(n-2) \cdot O(n^{(n+1)/4-1}) = O(n^{(n+1)/4}).$$

Nun zur letzten suboptimalen Fitnessschicht A_{n-2}. Sei $M \subseteq A_{n-2}$ die Menge der Suchpunkte, bei denen beide Teilbäume der Wurzel jeweils einheitlich, aber mit verschiedenen Farben gefärbt sind. Damit enthält M genau vier Färbungen, bei denen die Endpunkte der einzigen zweifarbigen Kante auf den höchst möglichen Ebenen 0 und 1 liegen.

Angenommen, der aktuelle Suchpunkt des (1+1) EA ist ein Suchpunkt $x \in M$ und $T(v)$ ist der einzige verbessernde Teilbaum in x. Es wäre nun unvernünftig, darauf zu warten, dass $T(v)$ in einem Schritt komplett flippt. Statt dessen warten wir darauf, dass ein kleinerer verbessernder Teilbaum entsteht und der ψ-Wert des aktuellen Suchpunkts sinkt. Dies kann beispielsweise dadurch geschehen kann, dass genau die Knoten in $T(v_1) \cup \{v\}$ oder $T(v_2) \cup \{v\}$ flippen, wobei v_1, v_2 die Kinder von v sind. Da die Teilbäume $T(v_i)$ genau $2^{d-1}-1$ Knoten enthalten und die Mengen $T(v_i) \cup \{v\}$ damit $2^{d-1} = 2^{\log(n+1)-2} = (n+1)/4$ Knoten beinhalten, ist die erwartete Zeit, bis ein Suchpunkt $y \notin M$ erreicht wird, nach oben beschränkt durch $O(n^{(n+1)/4})$.

Der erreichte Suchpunkt y enthält einen verbessernden Teilbaum $T(w)$ mit Tiefe $\psi(y) \leq d-2$. Mit etwas Glück flippt der (1+1) EA anschließend den verbessernden Teilbaum $T(w)$. Die Wahrscheinlichkeit für einen solchen Schritt ist $\Omega(n^{-((n+1)/4-1)})$, und wenn dies geschieht, erreichen wir direkt ein globales Optimum.

Allerdings kann es auch passieren, dass der (1+1) EA ausgehend von y, bevor er $T(w)$ flippt, wieder einen Suchpunkt $x' \in M$ erreicht und damit zur schlechten Ausgangsposition zurück kehrt. Wir können jedoch zeigen, dass die Wahrscheinlichkeit hierfür vergleichsweise klein ist.

Zunächst schätzen wir die Wahrscheinlichkeit ab, in einem Schritt zu einem Suchpunkt aus M zurück zu kehren. Es gibt in M genau vier Suchpunkte, bei denen beide Teilbäume der Wurzel einheitlich und mit verschiedenen Farben gefärbt sind; die Menge M hat also nur konstante Größe.

O. B. d. A. sei nun der verbessernde Teilbaum $T(w)$ in y einheitlich 1-gefärbt. Da $T(w)$ höchstens $(n+1)/4-1$ Knoten enthält und genau die Knoten in $T(w)$ mit Farbe 1 gefärbt sind, enthält y höchstens $(n+1)/4-1$ Einsen. Alle Teilbäume in M enthalten jedoch mindestens $(n+1)/2-1$ Einsen, da ein Teilbaum der Wurzel einheitlich 1-gefärbt ist. Damit beträgt der Hammingabstand von y zu allen Suchpunkten aus M mindestens $(n+1)/4$ und die Wahrscheinlichkeit, in einem Schritt von y zu einem der vier Suchpunkte aus M zu gelangen, ist $O(n^{-(n+1)/4})$.

Der (1+1) EA kann, bevor ein globales Optimum erreicht wird, beliebig oft zwischen den Mengen M und $A_{i-2} \setminus M$ hin und her springen. Von einem Suchpunkt aus $A_{n-2} \setminus M$ ausgehend, können wir, wenn die Menge $A_{n-2} \setminus M$ verlassen wird, entweder einen Suchpunkt in M erreichen oder ein globales Optimum. Sei A das Ereignis, dass

wir ausgehend von einem Suchpunkt in $A_{n-2} \setminus M$ einen Suchpunkt in M erreichen. Hingegen sei B das Ereignis, dass wir ausgehend von einem Suchpunkt in $A_{n-2} \setminus M$ ein globales Optimum erreichen.

Die Wahrscheinlichkeit $\text{Prob}(A \dot\cup B)$, mit der die Menge $A_{n-2} \setminus M$ in einem Schritt verlassen wird, ist $\Omega(n^{(n+1)/4-1})$. Wenn das Ereignis $A \dot\cup B$ eintritt, ist die bedingte Wahrscheinlichkeit, dass dabei das unerwünschte Ereignis A eintritt,

$$\text{Prob}(A \mid A \cup B) = \frac{\text{Prob}(A \cap (A \cup B))}{\text{Prob}(A \cup B)} = \frac{\text{Prob}(A)}{\text{Prob}(A \cup B)} = \frac{O(n^{(n+1)/4})}{\Omega(n^{(n+1)/4-1})} = O\left(\frac{1}{n}\right).$$

Falls wir doch wider Erwarten ausgehend von einem Suchpunkt in $A_{n-2} \setminus M$ einen Suchpunkt in M erreichen, warten wir darauf, dass die Menge M erneut verlassen wird. Daher müssen wir M im Erwartungswert nur höchstens $1 + O(1/n)$ Mal verlassen, bevor ausgehend von einem Suchpunkt $y \in A_{n-2} \setminus M$ Ereignis B eintritt und wir ein globales Optimum erreichen.

Wir addieren nun die erwarteten Zeiten, um M zu verlassen und die erwartete Zeit, um ausgehend von der Menge $A_{n-2} \setminus M$ ein Optimum zu erreichen. Damit erhalten wir eine obere Schranke für die erwartete Zeit, die Fitnessschicht A_{n-2} zu verlassen:

$$(1 + O(1/n)) \cdot O(n^{(n+1)/4}) + O(n^{(n+1)/4-1}) = O(n^{(n+1)/4}).$$

Insgesamt folgt daher eine obere Schranke für die erwartete Optimierungszeit von $O(n^{(n+1)/4})$. □

Im Beweis der oberen Schranke spielen die vier Suchpunkte der Menge M eine große Rolle, also Färbungen, bei denen beide Teilbäume der Wurzel einheitlich mit verschiedenen Farben gefärbt sind. Die erwartete Optimierungszeit wird durch die erwartete Zeit dominiert, bis wir ausgehend von einem Suchpunkt $x \in M$ zu einem globalen Optimum gelangen.

Die Zeiten für die übrigen Fitnesserhöhungen sind deutlich kleiner. Dies lässt sich aus dem Beweis nicht direkt ablesen; es zeigt sich darin, dass wir uns erlaubt haben, in den Fitnessschichten $A_0, \ldots, A_{n-3}$ sehr großzügige Abschätzungen vorzunehmen und darauf zu warten, dass ein verbessernder Teilbaum in einem einzigen Schritt flippt. Die wahren Zeiten für diese Fitnesserhöhungen sind wesentlich kleiner, da auch in den ersten Fitnessschichten ein verbessernder Teilbaum in mehreren Schritten flippen kann.

Eine untere Schranke für die erwartete Optimierungszeit

Auch beim Beweis einer unteren Schranke für die erwartete Optimierungszeit des (1+1) EA spielen die Suchpunkte aus M eine große Rolle, da sie gewissermaßen Worst-Case-Färbungen darstellen, in denen eine maximale Zahl von Bits flippen muss, um die Färbung wieder zu verlassen. Wir werden zeigen, dass die obere Schranke aus Theorem 15 für Suchpunkte aus M asymptotisch scharf ist und daraus mit der Methode des typischen Laufs (siehe Abschnitt 1.5.4) eine untere Schranke folgern.

Theorem 16. *Gegeben ein vollständiger binärer Baum $G = (V, E)$ mit $n := |V|$ Knoten und Tiefe $d := \log(n+1) - 1$. Die erwartete Laufzeit des (1+1) EA auf $f := \text{Ising}_G$ ist nach unten beschränkt durch $\Omega(2^{-n} \cdot n^{(n+1)/4}) = n^{\Omega(n)}$.*

Beweis. Mit Wahrscheinlichkeit $4 \cdot 2^{-n}$ wird bei der zufälligen Initialisierung ein Suchpunkt $x \in M$ gewählt, bei dem beide Teilbäume der Wurzel einheitlich, aber verschieden gefärbt sind. Wir nehmen im Folgenden an, dass dieses Ereignis eintritt.

Da x in der Fitnessschicht A_{n-2} liegt, werden nur noch Suchpunkte in $A_{n-2} \cup A_{n-1}$ akzeptiert, die also eine Fitness von mindestens $n-2$ haben. Die Zahl dieser Suchpunkte beträgt genau $2n$. Dies ergibt sich wie folgt: Um alle Suchpunkte in A_{n-2} zu erhalten, können wir die Position der einzigen zweifarbigen Kante aus $n-1$ Möglichkeiten wählen und haben dann zwei Möglichkeiten, die beiden Endpunkte dieser Kante zu färben; hinzu kommen noch zwei globale Optima.

Da Suchpunkte $y \notin M$ mit einer Fitness von mindestens $n-2$ zum einen selten und zum anderen weit entfernt sind, schätzen wir die erwartete Optimierungszeit nach unten ab durch die erwartete Zeit, irgendeinen Suchpunkt $y \notin M$ zu erreichen.

Im Beweis von Theorem 15 haben wir bereits gezeigt: Der Hammingabstand von einem Suchpunkt $y \in A_{n-2} \setminus M$ zu allen Suchpunkten aus M ist mindestens $(n+1)/4$. In Theorem 15 ließ sich daraus folgern, dass die Wahrscheinlichkeit, von der Menge $A_{n-2} \setminus M$ in die Menge M überzugehen, $O(n^{-(n+1)/4})$ ist, da die Größe von M konstant ist.

Das gleiche Argument wollen wir nun in umgekehrter Richtung anwenden, um die Wahrscheinlichkeit abzuschätzen, mit der der (1+1) EA von M in die Menge $A_{n-2} \setminus M$ übergeht. Allerdings müssen wir bedenken, dass es in der Zielmenge $A_{n-2} \setminus M$ linear viele Suchpunkte gibt. Wenn wir die gleichen Abschätzungen wie in Theorem 15 verwenden, erhalten wir für die Wahrscheinlichkeit, von der Menge M in die Menge $A_{n-2} \setminus M$ überzugehen, nur eine um Faktor n schwächere untere Schranke von $O(n^{-((n+1)/4-1)})$.

Allerdings lässt sich dieses Problem leicht beheben. Es gibt nur 8 Färbungen in $A_{n-2} \setminus M$, bei denen der verbessernde Teilbaum eine Tiefe von $d-2$ hat. Für diese wenden wir die untere Schranke $(n+1)/4$ für den Hammingabstand an. Es folgt: Die Wahrscheinlichkeit, ausgehend von $x \in M$ einen dieser 8 Suchpunkte zu erreichen, ist $O(n^{-(n+1)/4})$.

Alle anderen Färbungen in $A_{n-2} \setminus M$ haben einen größeren Hammingabstand zu allen Suchpunkten in M: Da der verbessernde Teilbaum in solch einer Färbung höchstens Tiefe $d-3$ und damit $(n+1)/8 - 1$ Knoten haben kann, ist die Zahl der Einsen oder die Zahl der Nullen höchstens $(n+1)/8 - 1$ und der Hammingabstand zu allen Suchpunkten aus M beträgt mindestens

$$\left(\frac{n+1}{2} - 1\right) - \left(\frac{n+1}{8} - 1\right) = \frac{3(n+1)}{8}.$$

Die Wahrscheinlichkeit, ausgehend von $x \in M$ zu einem dieser Suchpunkte zu springen, ist daher $O(n) \cdot O(n^{-3(n+1)/8}) = O(n^{-(3(n+1)/8+1)})$.

Zuletzt haben globale Optima einen Abstand von mindestens $(n+1)/2-1$ zu allen Suchpunkten aus M. Die Wahrscheinlichkeit, direkt zu einem globalen Optimum zu springen, ist daher $O(n^{-((n+1)/2-1)})$.

Die Wahrscheinlichkeit, die Menge M zu verlassen, ergibt sich nun aus der Summe der Wahrscheinlichkeiten, zu einer der drei genannten Mengen von Suchpunkten außerhalb von M überzugehen. Diese Wahrscheinlichkeit ist daher nach oben beschränkt durch

$$O(n^{-(n+1)/4}) + O(n^{-(3(n+1)/8+1)}) + O(n^{-((n+1)/2-1)}) = O(n^{-(n+1)/4}).$$

Die erwartete Optimierungszeit ist daher ausgehend von $x \in M$ nach unten beschränkt durch $\Omega(n^{(n+1)/4})$ und zusammen mit der Wahrscheinlichkeit $\Omega(2^{-n})$, mit einem Suchpunkt $x \in M$ zu starten, folgt eine unbedingte erwartete Optimierungszeit von $\Omega(2^{-n} \cdot n^{(n+1)/4})$. □

Die Wahrscheinlichkeit einer Worst-Case-Färbung

Die untere Schranke ist durch den Faktor 2^{-n} sehr grob geraten. Es ist zu vermuten, dass eine Worst-Case-Färbung $x \in M$ mit sehr viel höherer Wahrscheinlichkeit als $4 \cdot 2^{-n}$ erreicht wird. Konkret stellen wir uns vor, dass wir einen Lauf so lange laufen lassen, bis entweder ein globales Optimum oder eine Worst-Case-Färbung $x \in M$ erreicht wird. Wir vermuten, dass dann die Wahrscheinlichkeit, eine Worst-Case-Färbung zu erreichen, deutlich größer als $4 \cdot 2^{-n}$ ist.

Eine mutige Hypothese wäre, dass diese Wahrscheinlichkeit sogar durch eine Konstante nach unten beschränkt ist; in diesem Fall wäre die Laufzeit des (1+1) EA in der Größenordnung $\Theta(n^{(n+1)/4})$.

Hypothese 1. *Gegeben ein vollständiger binärer Baum $G = (V, E)$. Die Wahrscheinlichkeit, dass der (1+1) EA auf* Ising_G *eine Worst-Case-Färbung mit einheitlich, aber verschieden gefärbten Teilbäumen der Wurzel erreicht, ist* $\Omega(1)$.

Diese Hypothese wollen wir nun näher in Augenschein nehmen und von verschiedenen Seiten beleuchten.

Wenn man den Graphen $G = (V, E)$ modifiziert, indem man eine zur Wurzel inzidente Kante e entfernt, liegen die beiden Teilbäume der Wurzel im resultierenden Graphen $G^* = (V, E \setminus \{e\})$ in verschiedenen Zusammenhangskomponenten. Mit Lemma 3 aus Abschnitt 2.4 lässt sich dann leicht zeigen, dass auf f_{G^*} beide Zusammenhangskomponenten mit Wahrscheinlichkeit 1/2 verschieden gefärbt werden, was übertragen auf den Graphen G einer Färbung aus M entspricht.

Bei der Analyse zweier verbundener Cliquen mit einer Brückenkante in Abschnitt 3.3 haben wir gesehen, dass eine einzelne Kante, die zwei Teilgraphen miteinander verbindet, einen verschwindend kleinen Einfluss auf die Färbung der beiden Teilgraphen hat. Dies wäre ein Indiz dafür, dass die eine zur Wurzel inzidente Kante e, die den Unterschied zwischen G und G^* ausmacht, ebenfalls nur einen kleinen Einfluss hat

und die Wahrscheinlichkeit, dass beide Teilbäume der Wurzel auf f_G verschieden gefärbt werden, ebenfalls durch eine Konstante nach unten beschränkt ist.

Einzuwenden ist allerdings, dass die Graphstrukturen völlig verschieden sind: Bei zwei verbundenen Cliquen ist die Brückenkante nur eine von sehr vielen anderen Kanten und ihr Einfluss wird deshalb vom Einfluss der anderen Kanten übertönt. Auf Bäumen ist jede Kante aber eine von höchstens 3 zu einem Knoten v inzidenten Kanten, so dass der Einfluss einer Kante auf die Endpunkte wesentlich größer ist. Trotzdem können zwei andere zu v inzidente Kanten den Einfluss einer dritten Kante überstimmen, sofern die zwei anderen Kanten einfarbig sind. Durch passende Konstellationen gut gefärbter Knoten können sich also dennoch stabile Teilfärbungen entwickeln, die nur dann aufgebrochen werden können, wenn an anderer Stelle die Fitness (lokal) erhöht wird.

Zweitens müssen wir beachten, dass gegenüber der Analyse von Cliquen die betrachteten Zeiträume ganz verschiedene Größenordnungen haben. Bei Cliquen endet ein Lauf in erwarteter Zeit $O(n \log n)$ in einem globalen Optimum oder einer Worst-Case-Färbung mit verschieden gefärbten Cliquen. Die erwartete Zeit für das analoge Ereignis auf vollständigen binären Bäumen ist exponentiell. Die eine verbindende Kante kann ihren Einfluss also über einen viel größeren Zeitraum ausspielen.

Ein Ansatz, um Hypothese 1 mit theoretischen Mitteln zu untersuchen, wäre, die möglichen Operationen des (1+1) EA genauer zu analysieren. Wir haben bei der Analyse des (1+1) EA und der Betrachtung verbessernder Teilbäume gesehen, dass sich die Wahrscheinlichkeiten verschiedener Fitness verbessernder Operationen sehr stark voneinander unterscheiden können. Man könnte daher die Methode des typischen Laufs anwenden und darauf spekulieren, dass stets eine der wahrscheinlichsten Operationen ausgeführt wird.

Beispielsweise kann man darauf spekulieren, dass ein verbessernder Teilbaum minimaler Tiefe flippt, bevor ein anderer verbessernder Teilbaum größerer Tiefe flippt. Anders ausgedrückt: Wenn ein verbessernder Teilbaum flippt, kann man in einem typischen Lauf davon ausgehen, dass dieser Teilbaum eine minimale Tiefe unter allen verbessernden Teilbäumen hat.

Leider übersieht man bei der Betrachtung verbessernder Teilbäume aber leicht, dass es viele andere Operationen gibt, die den aktuellen Suchpunkt verändern können. So gibt es zusammenhängende Mengen innerer Knoten, die keinen Teilbaum bilden, aber trotzdem die Fitness erhöhen können, wenn genau die Knoten einer solchen Menge flippen. Zwei Beispiele sind in Abbildung 4.7 dargestellt. Die Existenz und das Aussehen solcher Operationen hängt stark von den Färbungen der umgebenden Knoten ab, was die Analyse schwierig macht.

Auch die Reihenfolge bei mehreren gleich wahrscheinlichen Operationen spielt eine Rolle: Oft ergeben sich auf lange Sicht unterschiedliche Färbungen, je nachdem, welche der wahrscheinlichsten Operationen zuerst ausgeführt wird. Ein solches Szenario ist in Abbildung 4.8 skizziert.

Alles in allem scheint eine theoretische Analyse aufgrund der hohen Bandbreite möglicher Operationen schwierig zu sein. Daher bleibt die Frage, ob man die Hypo-

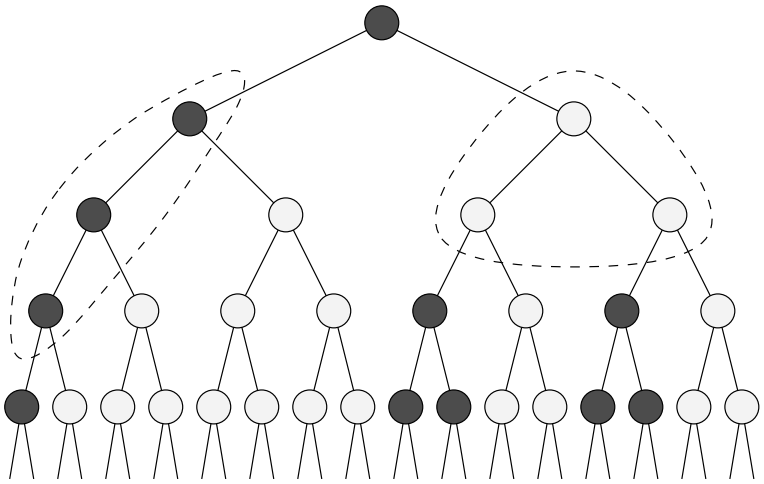


Abbildung 4.7: Zwei Beispiele für Fitness verbessernde Operationen, die ausschließlich innere Knoten betreffen. Flippen genau die Knoten in der linken umrahmten Menge, wird ein Pfad aus dunkel gefärbten Knoten innerhalb einer hell gefärbten Umgebung zerschnitten. Die Knoten der rechten umrahmten Menge bilden eine zusammenhängende Menge hell gefärbter Knoten. Nach oben ist eine solche Menge durch einen Knoten mit dunkel gefärbtem Elter begrenzt und nach unten durch Knoten, die keine zwei hell gefärbten Kinder haben.

these experimentell testen kann. Hier ergibt sich aber das Problem, dass die erwartete Zeit, bis der (1+1) EA ein globales Optimum oder eine Worst-Case-Färbung erreicht, exponentiell ist. Hinzu kommt, dass die Knotenzahl eines vollständigen binären Baums exponentiell in der Tiefe ist und Instanzen mit kleinen Knotenzahlen dementsprechend selten sind.

In vernünftiger Zeit lassen sich vollständige binäre Bäume nur bis zu einer Tiefe von $d = 4$ untersuchen, was einer Suchraumdimension von $n = 31$ entspricht und bei weitem keine verlässlichen Hinweise auf das asymptotische Verhalten des (1+1) EA gibt. Für vollständige binäre Bäume mit Tiefe $d \in \{2, 3, 4\}$ deuten Experimente jedoch an, dass Hypothese 1 wahr sein könnte.

Wir haben also keinen Erfolg versprechenden Ansatz, um Hypothese 1 zu testen. Dennoch können wir mit der groben unteren Schranke aus Theorem 16 zufrieden sein. Wenn wir n als $2^{\log n}$ schreiben, beträgt die obere Schranke $O(2^{((n+1)/4)\cdot\log n})$ und die untere Schranke $\Omega(2^{-n} \cdot 2^{((n+1)/4)\cdot\log n}) = \Omega(2^{((n+1)/4)\cdot\log n - n})$. Beide Schranken lassen sich durch $2^{\Theta(n \log n)}$ oder $n^{\Theta(n)}$ ausdrücken, sie unterscheiden sich also „nur“ durch die Konstanten im Exponenten.

Eine untere Schranke für beliebige Mutationswahrscheinlichkeiten

Die untere Schranke aus Theorem 16 zeigt, dass der (1+1) EA auf vollständigen binären Bäumen schlecht abschneidet. Im Hinblick auf den Vergleich von Mutation und Rekombination wollen wir uns nun fragen, ob Mutationen auf dieser Klasse

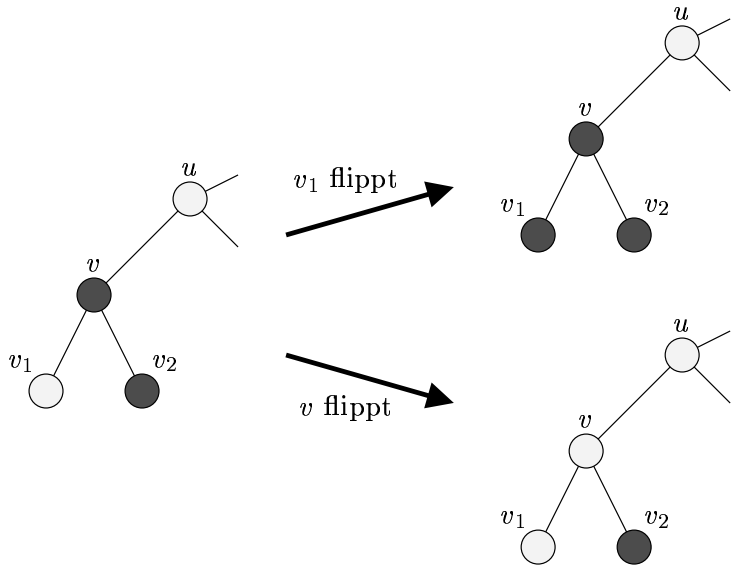


Abbildung 4.8: Eine Teilfärbung mit zwei verbessernden 1-Bit-Mutationen an den Knoten v und v_1. Es entstehen unterschiedliche Färbungen, je nachdem, welche dieser verbessernden 1-Bit-Mutationen ausgeführt wird. Flippt v_1, ist $T(v)$ einheitlich dunkel gefärbt. Flippt hingegen v, ist die Chance groß, dass $T(v)$ später einheitlich hell gefärbt wird. Eine solche Entscheidung kann Auswirkungen auf die Färbung höherer Ebenen des Baumes haben.

von Probleminstanzen generell nicht sinnvoll eingesetzt werden können oder ob wir die voreingestellte Mutationswahrscheinlichkeit $p_m := 1/n$ falsch gewählt haben. Immerhin müssen in einem Lauf oft viele Knoten auf einen Schlag geflippt werden und dabei könnte eine höhere Mutationswahrscheinlichkeit helfen.

Wir unterscheiden zwei Fälle, zum einen den Fall $p_m < 1/32$ und zum anderen den Fall $1/32 \leq p_m \leq 1/2$. Mutationswahrscheinlichkeiten $p_m > 1/2$ sind zum einen semantisch fragwürdig, da weit entfernte Suchpunkte bevorzugt werden. Zum anderen haben sie den gleichen Effekt wie ein Mutationsoperator, der mit Mutationswahrscheinlichkeit $1 - p_m$ arbeitet und nach einer Mutation den kompletten Bitstring invertiert. Aufgrund der Bit-Flip-Symmetrie verhält sich der Algorithmus im Wesentlichen also wie bei einer Mutationswahrscheinlichkeit von $1 - p_m < 1/2$ und es reicht aus, den Fall $p_m \leq 1/2$ zu betrachten.

Falls $p_m < 1/32$, verwenden wir die Ideen aus dem Beweis von Theorem 16. Die Wahrscheinlichkeit, mit einem Suchpunkt $x \in M$ zu starten, ist $4 \cdot 2^{-n}$. Wir erinnern uns daran, dass der (1+1) EA ausgehend von einem beliebigen Suchpunkt $x \in M$ einen von $O(n)$ Suchpunkten in $(A_{n-2} \cup A_{n-1}) \setminus M$ erreichen muss, die alle einen Hammingabstand von mindestens $(n+1)/4$ zu x haben. Um einen bestimmten Suchpunkt $y \in (A_{n-2} \cup A_{n-1}) \setminus M$ zu treffen, müssen also mindestens $(n+1)/4$ ausgewählte Bits flippen. Die Wahrscheinlichkeit hierfür ist höchstens

$$p_m^{(n+1)/4} < 32^{-(n+1)/4} < 2^{-(5/4)n}.$$

Die erwartete Optimierungszeit des (1+1) EA mit einer Mutationswahrscheinlichkeit von $p_m < 1/32$ oder $p_m > 31/32$ ist daher mindestens $4 \cdot 2^{-n} \cdot 2^{(5/4)n}/O(n) = 2^{\Omega(n)}$.

Falls $1/32 \leq p_m \leq 1/2$, haben wir Schwierigkeiten, einen bestimmten Suchpunkt genau zu treffen, da typischerweise zu viele Bits auf einmal flippen. Mit Wahrscheinlichkeit $1 - 2 \cdot 2^{-n}$ wird der Algorithmus mit einem suboptimalen Suchpunkt x gestartet. Um einen bestimmten Suchpunkt $y \in \{0,1\}^n$ zu treffen, muss jedes Bit x_i den Wert y_i annehmen; die Wahrscheinlichkeit hierfür ist p_m, falls $x_i \neq y_i$, und $1 - p_m$, falls $x_i = y_i$. Eine untere Schranke für beide Belegungen von y_i ist der Wert $1 - p_m$, da $p_m \leq 1/2$. Die Wahrscheinlichkeit, von irgendeinem Suchpunkt x aus irgendeinen Suchpunkt y genau zu treffen, ist daher höchstens

$$(1 - p_m)^n \leq \left(\frac{31}{32}\right)^n = 2^{-\Omega(n)}.$$

Die erwartete Zeit, bis eines der beiden Optima getroffen wird, ist daher $2^{\Omega(n)}$; die erwartete Optimierungszeit des (1+1) EA mit Mutationswahrscheinlichkeit $1/32 \leq p_m \leq 31/32$ ist $2^{\Omega(n)}$.

Wir halten also fest:

Korollar 3. *Gegeben ein vollständiger binärer Baum $G = (V, E)$ mit $n := |V|$ Knoten. Die erwartete Optimierungszeit des (1+1) EA auf* Ising_G *mit beliebiger Mutationswahrscheinlichkeit ist nach unten beschränkt durch $2^{\Omega(n)}$.*

4.3 Algorithmen mit Rekombination

Wir haben gesehen, dass mutationsbasierte Algorithmen auf dem Ising-Modell auf vollständigen binären Bäumen chancenlos sind. Daher wollen wir uns nun der Frage widmen, ob genetische Algorithmen mit Rekombination besser abschneiden.

Zwei-Punkt-Kreuzungen

Als Rekombinationsoperator verwenden wir so genannte *Zwei-Punkt-Kreuzungen*, die aus zwei Eltern zwei Nachkommen erzeugen. Der Name rührt daher, dass zwei *Kreuzungspunkte* gewählt werden, mit denen die Bitstrings beider Eltern in Teile zerlegt werden. Die Kinder werden erzeugt, indem Teile aus verschiedenen Eltern wieder zu neuen Bitstrings zusammengesetzt werden.

Definition 15. *Eine* Zwei-Punkt-Kreuzung *von Eltern $x, y \in \{0,1\}^n$ erzeugt zwei Kinder $x', y' \in \{0,1\}^n$ wie folgt. Wir wählen zwei Kreuzungspunkte $a < b$ uniform zufällig aus der Menge zweielementiger Teilmengen von $\{1, \ldots, n\}$. Dann ist $x' = (x'_1, \ldots, x'_n)$ und $y' = (y'_1, \ldots, y'_n)$ mit*

$$x'_i := \begin{cases} x_i, & (i \leq a) \vee (i > b) \\ y_i, & a < i \leq b \end{cases} \qquad y'_i := \begin{cases} x_i, & a < i \leq b \\ y_i, & (i \leq a) \vee (i > b) \end{cases}$$

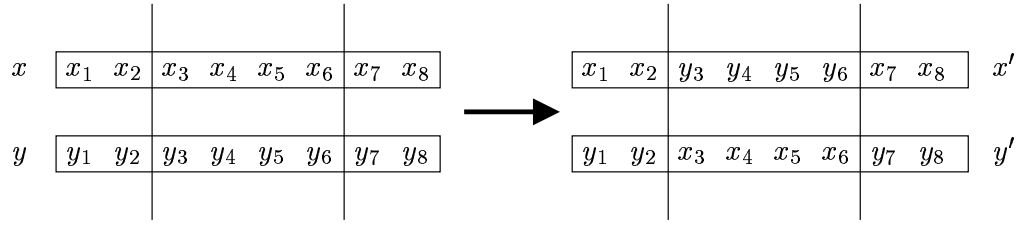


Abbildung 4.9: Eine Visualisierung von Zwei-Punkt-Kreuzungen auf $n = 8$ Bits mit Kreuzungspunkten $a = 2$ und $b = 6$.

Ein Beispiel für eine Zwei-Punkt-Kreuzung ist in Abbildung 4.9 visualisiert.

Abhängig von den beiden Kreuzungspunkten gilt also für eine Bitposition i: $x'_i = x_i$ und $y'_i = y_i$ oder $x'_i = y_i$ und $y_i = x_i$. Alle Belegungen von Bits der Eltern an Position i finden sich in den Belegungen der Kinder an gleicher Position wieder. Es gehen also keine Belegungen verloren, sie werden i. A. nur anders zusammengesetzt.

Eine wichtige Folgerung hieraus ist: Genau dann, wenn $x_i \neq y_i$, ist $x'_i \neq y'_i$. Für die Hammingabstände gilt $H(x', y') = H(x, y)$; eine Zwei-Punkt-Kreuzung bewahrt den Hammingabstand der Eltern.

Nutzen von Rekombination

Wie kann nun Rekombination helfen? Die Building-Block-Hypothese besagt, dass Rekombination einzelne Bausteine guter Lösungen zu einer guten Gesamtlösung kombinieren kann. Als „Bausteine“ können hier die Teilfärbungen aller Teilbäume herhalten. Durch die Codierung gemäß einer Preorder-Nummerierung ist es möglich, dass die Kreuzungspunkte im Bitstring genau die Bits eines Teilbaum $T(v)$ abgrenzen. Wenn x, y die Eltern bei einer solchen Zwei-Punkt-Kreuzung sind, wird aus x ein Nachkomme x' erzeugt, indem die Belegung eines Bausteins, des Teilbaums $T(v)$, durch die Belegung aus dem anderen Elter y ersetzt wird.

Allerdings sollten wir beachten, dass es $\binom{n}{2}$ mögliche Paare von Kreuzungspunkten gibt, jedoch nur n Teilbäume. Die meisten Zwei-Punkt-Kreuzungen grenzen daher irgendeine Knotenmenge ab, die der rekursiven Struktur von Bäumen und Teilbäumen zuwider läuft. Abbildung 4.10 zeigt ein Beispiel für eine solche Operation.

Da die Auswirkungen von Zwei-Punkt-Kreuzungen, die nicht genau einen Teilbaum treffen, nur schwer zu kontrollieren sind, werden wir uns in unseren Analysen auf Zwei-Punkt-Kreuzungen konzentrieren, die genau einen Teilbaum abgrenzen.

Solche Operationen sind besonders dann hilfreich, wenn ein Teilbaum $T(v)$ ausgewählt wird, der in beiden Eltern x und y verbessernd ist, und für den die Wurzel v in x und y komplementär gefärbt ist: $x_v \neq y_v$. In beiden Nachkommen ist dann die zuvor zweifarbige Kante von v zu seinem Elter u einfarbig gefärbt. Für die summierten Fitnesswerte von Eltern und Kindern gilt

$$f(x') + f(y') = f(x) + f(y) + 2,$$

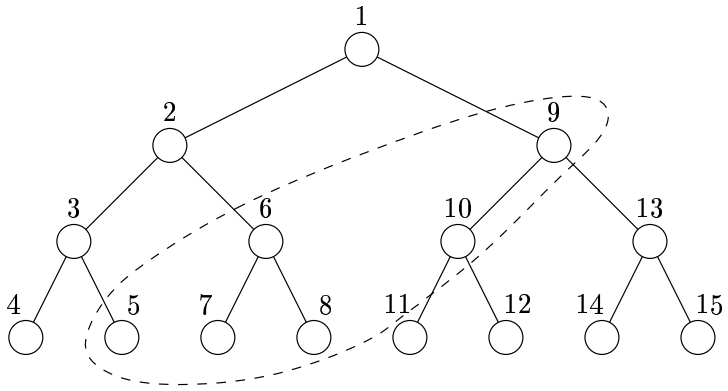


Abbildung 4.10: Eine Skizze einer durch eine Zwei-Punkt-Kreuzung abgegrenzten Knotenmenge, die keinen Teilbaum darstellt. Die Knoten sind mit ihrer Preorder-Nummerierung beschriftet, die Kreuzungspunkte sind $a = 4$ und $b = 10$.

da die Teilfärbungen von Kanten innerhalb von $T(v)$ lediglich zwischen x' und y' ausgetauscht werden. Alle in x und y einfarbigen Kanten gehen auch in die Summe $f(x') + f(y')$ ein und die nun in x' und y' einfarbige Kante $\{u, v\}$ liefert einen zusätzlichen Beitrag von 2.

Wenn die aus einer solchen günstigen Zwei-Punkt-Kreuzung entstandenen Nachkommen x', y' ihre Eltern x, y in der Population ersetzen, steigt die durchschnittliche Fitness der Individuen in der Population. Auf diese Art und Weise können Zwei-Punkt-Kreuzungen zum Optimierungsprozess beitragen.

Allerdings ist eine Voraussetzung für eine solche Operation, dass die Wurzel v (und damit auch ihr Elter u) in beiden Elter-Suchpunkten komplementär gefärbt sind. Man kann sich leicht überlegen, dass alle anderen Kombinationen von Färbungen für v und u in x und y zur Folge haben, dass $f(x') + f(y') \leq f(x) + f(y)$. (In solch einer Situation kann Rekombination nur dann die durchschnittliche Fitness der Individuen in der Population erhöhen, wenn die Selektion zur Übernahme beliebige Kombinationen von Eltern und Nachkommen zulässt.)

Für den profitablen Einsatz von Zwei-Punkt-Kreuzungen ist also wichtig, dass die Population eine ausreichend große Diversität bereit stellt. Hier sei auf Abschnitt 1.2.1 verwiesen, in dem wir bereits die Notwendigkeit von Diversitätserhaltung erläutert haben. Um Rekombination sinnvoll einsetzen zu können, wollen wir im Folgenden ebenfalls Maßnahmen zur Diversitätserhaltung treffen.

4.3.1 Die Laufzeit des (1+1) GIGA

Eine Population $P = \{x, y\}$ aus zwei Individuen erreicht eine maximale Diversität, wenn $y = \overline{x}$ ist. In einem solchen Szenario sind alle in x zweifarbigen Kanten $\{u, v\}$ auch in $y = \overline{x}$ zweifarbig und die Endpunkte u, v sind in x und y jeweils komplementär gefärbt. Jede Zwei-Punkt-Kreuzung von x und $y = \overline{x}$, die genau einen

verbessernden Teilbaum abgrenzt, erhöht sowohl die Fitness von x als auch die Fitness von $y = \overline{x}$ um jeweils 1.

Als ersten Algorithmus mit Rekombination wollen wir einen speziellen Algorithmus betrachten, der stets eine maximale Diversität garantiert. Der so genannte *(1+1) GIGA* wurde bereits von Fischer [5] und Fischer und Wegener [7] auf dem Ising-Modell auf dem Ring untersucht. Seine Strategie zur Diversitätserhaltung besteht ganz einfach darin, dass er stets eine Population aus einem Suchpunkt $x \in \{0,1\}^n$ und seinem bitweisen Komplement $\overline{x}$ verwaltet.

Die einzigen Operationen, die der (1+1) GIGA ausführt, sind Zwei-Punkt-Kreuzungen von x und $\overline{x}$. Da Zwei-Punkt-Kreuzungen den Hammingabstand der Eltern bewahren, sind die beiden erzeugten Nachkommen ebenfalls komplementär zueinander.

Algorithmus 4 ((1+1) GIGA).

1. *Wähle einen Suchpunkt $x \in \{0,1\}^n$ uniform zufällig. Setze $P := \{x, \overline{x}\}$.*
2. *$(x', \overline{x'}) :=$ Zwei-Punkt-Kreuzung$(x, \overline{x})$.*
3. *Falls $f(x') \geq f(x)$, setze $P := \{x', \overline{x'}\}$.*
4. *Weiter bei 2.*

Der (1+1) GIGA ist speziell an die Bit-Flip-Symmetrie der Ising-Funktion angepasst; das Selektionskriterium hängt nur von x und x' ab und ignoriert die beiden Komplemente $\overline{x}$ und $\overline{x'}$. Daher ist ein vernünftiges Verhalten des (1+1) GIGA nur auf bit-flip-symmetrischen Fitnessfunktionen garantiert.

Im Grunde genommen ist es auch fragwürdig, von einer „Population“ $P = \{x, \overline{x}\}$ zu sprechen, da in Wahrheit nur ein Suchpunkt x vorliegt; das Komplement $\overline{x}$ ist redundant. Eine andere Sichtweise des (1+1) GIGA ist der eines mutationsbasierten Algorithmus mit Populationsgröße 1 und einem speziellen Mutationsoperator: Dieser wählt zwei Kreuzungspunkte und invertiert die Bits, die durch die Kreuzungspunkte abgesteckt werden.

Wir wollen den (1+1) GIGA im Folgenden als eine Art idealisierten genetischen Algorithmus ansehen, der stets eine maximal mögliche Diversität garantiert und damit ideale Bedingungen für Rekombination bereit stellt. Die Analyse des (1+1) GIGA soll uns Aufschluss über die Auswirkungen und den Nutzen von Zwei-Punkt-Kreuzungen geben, so dass wir uns danach mit den neu gewonnenen Erkenntnissen an die Analyse „realistischerer“ genetischer Algorithmen wagen können.

Theorem 17. *Gegeben ein beliebiger Baum $G = (V, E)$ mit $n := |V|$ Knoten. Die erwartete Optimierungszeit des (1+1) GIGA auf $f := \mathrm{Ising}_G$ ist beschränkt durch $\Omega(n^2)$ und $O(n^2 \log n)$.*

Beweis. Mit der Sichtweise des (1+1) GIGA als mutationsbasierten Algorithmus mit Populationsgröße 1 können wir die Methode der Fitnessschichten anwenden. Dazu wählen wir die kanonische Einteilung in Schichten $A_i := \{x \mid f(x) = i\}$.

Wie bereits gesehen, wird die Fitness beider Suchpunkte x und $\overline{x}$ um jeweils 1 erhöht, wenn eine Zwei-Punkt-Kreuzung von x und $\overline{x}$ einen verbessernden Teilbaum in x abgrenzt. Ein Beispiel für eine Färbung x und drei verbessernde Teilbäume ist in Abbildung 4.11 dargestellt.

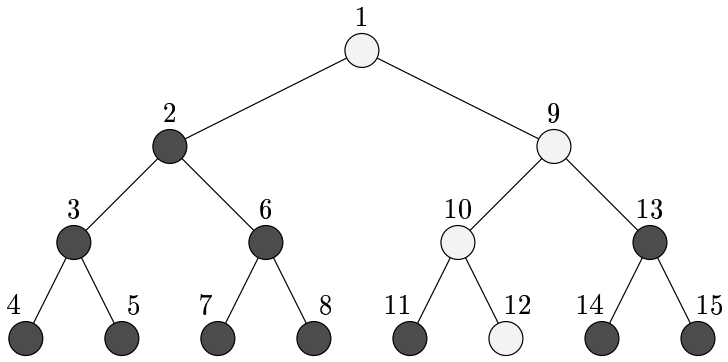


Abbildung 4.11: Ein Suchpunkt x mit drei dunkel gefärbten verbessernden Teilbäumen. Zwei-Punkt-Kreuzungen von x und $\overline{x}$ mit Kreuzungspunkten $(1, 8), (10, 11)$ oder $(12, 15)$ ersetzen die Teilfärbungen der jeweiligen verbessernden Teilbäume durch hell gefärbte Teilfärbungen aus $\overline{x}$.

Die Wahrscheinlichkeit für eine Zwei-Punkt-Kreuzung, die einen bestimmten verbessernden Teilbaum $T(v)$ abgrenzt, ist $1/\binom{n}{2}$. Bei einer Fitness von $f(x) = i$ gibt es in x genau $|E| - i = n - i - 1$ zweifarbige Kanten und aufgrund der eineindeutigen Beziehung zwischen zweifarbigen Kanten und verbessernden Teilbäumen gibt es in x auch $n - i - 1$ verbessernde Teilbäume.

Mit der Methode der Fitnessschichten erhalten wir eine obere Schranke für die erwartete Optimierungszeit von

$$1 + \sum_{i=0}^{n-2} \frac{1}{n-i-1} \cdot \binom{n}{2} \;=\; 1 + \binom{n}{2} \cdot \sum_{i=1}^{n-1} \frac{1}{i} \;=\; O(n^2 \log n).$$

Die untere Schranke folgt mit der Methode des typischen Laufs. Mit Wahrscheinlichkeit $1 - 2 \cdot 2^{-n}$ starten wir mit suboptimalen Suchpunkten $x, \overline{x} \notin \{0^n, 1^n\}$. Für die untere Schranke interessiert uns nur, wie lange wir auf den allerletzten Schritt warten, der aus zwei suboptimalen Suchpunkten $x, \overline{x}$ das eindeutige Paar $0^n, 1^n$ globaler Optima erzeugt.

Bei zwei komplementären Eltern $x, \overline{x}$ erzeugen zwei verschiedene Paare von Kreuzungspunkten verschiedene Paare von Nachkommen. Daher gibt es nur höchstens ein Paar von Kreuzungspunkten, das aus beliebigen Eltern $x, \overline{x}$ die beiden globalen Optima erzeugt. (Ein Paar von Kreuzungspunkten gibt es nur, sofern $0^n, 1^n$ aus $x, \overline{x}$ durch eine Zwei-Punkt-Kreuzung erzeugt werden können.)

Die erwartete Zeit, bis aus suboptimalen Suchpunkten $x, \overline{x}$ die beiden globalen Optima erzeugt werden, ist daher nach unten beschränkt durch $\Omega(n^2)$. □

Es sei angemerkt, dass Theorem 17 für beliebige Bäume gilt und nicht auf vollständige binäre Bäume beschränkt ist.

Die untere Schranke ist vermutlich nicht scharf, was unter anderem daran liegt, dass die Effekte von Zwei-Punkt-Kreuzungen, die nicht genau einen Teilbaum abgrenzen, nur schwer zu analysieren sind. Abbildung 4.10 lässt erahnen, dass eine Zwei-Punkt-Kreuzung im Extremfall $\Theta(\log n)$ Kanten auf einen Schlag umfärben kann, was eine genauere Analyse schwierig macht.

Die obere Schranke $O(n^2 \log n)$ ist erstaunlich klein, wenn man bedenkt, dass wir auf eine spezielle Zwei-Punkt-Kreuzung im Erwartungswert $\Theta(n^2)$ Schritte warten müssen. Auf jeden Fall zeigt Theorem 17, dass Rekombination mit einem speziellen Algorithmus zu einer polynomiellen erwarteten Optimierungszeit führen kann.

Dennoch sollten wir das Ergebnis nicht überbewerten, da der (1+1) GIGA ein sehr spezieller Algorithmus ist und durch die Ausnutzung der Bit-Flip-Symmetrie nicht zur Klasse der allgemeinen Suchheuristiken gezählt werden kann.

4.3.2 Die Laufzeit eines GAs mit Fitness Sharing

Interessanter ist die Frage, ob wir ähnliche Ergebnisse auch mit Standardmethoden zur Diversitätserhaltung erzielen können, die nicht speziell an die Bit-Flip-Symmetrie der Ising-Funktion angepasst sind.

Fitness Sharing

In Abschnitt 1.2.1 haben wir bereits das so genannte *Fitness Sharing* kennen gelernt. Ähnliche Suchpunkte müssen die Fitness unter sich aufteilen; „Klumpen“ von Suchpunkten werden so mit Strafkosten belegt und erhalten einen niedrigeren effektiven Fitnesswert. Die Selektion tut ein Übriges und sorgt dafür, dass solche Klumpen aufgelöst werden, indem ähnliche Suchpunkte nach und nach aus der Population verdrängt werden. Hat diese Strategie Erfolg, werden mit der Zeit die Suchpunkte der Population über große Teile des Suchraums verteilt.

Wir haben bereits in Abschnitt 1.2.1 angeführt, dass die Fitness eines Suchpunkts x auf die Population P bezogen wird und sich diese Fitness als Quotient aus der „echten“ Fitness f und der *Sharing-Funktion* $Sh(x, P)$ ergibt, die ein Maß für die Ähnlichkeit von x zu den Individuen der Population P darstellt:

$$f(x, P) := \frac{f(x)}{Sh(x, P)}.$$

Es ist üblich, für die Sharing-Funktion folgende Formel zu verwenden:

$$Sh(x, P) := \sum_{y \in P} \max\{1 - d(x, y)/\sigma, 0\}$$

Dabei ist $d(x,y)$ ein vernünftiges Distanzmaß für die Distanz zwischen x und y und σ ist ein Schwellenwert, der angibt, bis zu welcher Distanz Suchpunkte ihre Fitness teilen sollen. Im Fall $d(x,y) > \sigma$ ist $1 - d(x,y)/\sigma < 0$ und durch den max-Operator wird der Wert 0 angerechnet.

In dieser Arbeit wählen wir das Distanzmaß $d(x,y)$ als Hammingabstand $H(x,y)$. Den Schwellenwert σ belegen wir mit $\sigma := n$, da wir große Hammingabstände erreichen wollen und bei $\sigma := n$ Individuen immer ihre Fitness teilen. Ein Nebeneffekt dieser Wahl ist, dass $1 - H(x,y)/n$ stets nicht negativ ist und wir damit den max-Operator weglassen können. Damit erhalten wir folgende einfachere Formel für die Sharing-Funktion.

Definition 16. *Gegeben eine Population $P \subseteq \{0,1\}^n$. Wir definieren die Sharing-Funktion eines Individuums $x \in P$ bezüglich der Population P als*

$$Sh(x,P) := \sum_{y \in P} (1 - H(x,y)/n)$$

und die Fitness von $x \in P$ mit Fitness Sharing bezogen auf die Population P als

$$f(x,P) := \frac{f(x)}{Sh(x,P)}.$$

Die Fitness der Population P ist dann

$$f(P) := \sum_{x \in P} f(x,P).$$

Es sei angemerkt, dass wegen $x \in P$ die Summe in der Sharing-Funktion stets den Summanden $1 - H(x,x)/n = 1$ enthält, so dass $Sh(x,P)$ immer mindestens 1 beträgt. Der Quotient aus $f(x)$ und $Sh(x,P)$ ist daher stets wohldefiniert.

Ein einfacher genetischer Algorithmus mit Fitness Sharing

Mit diesen Notationen wollen wir nun einen einfachen genetischen Algorithmus definieren, der mit Fitness Sharing arbeitet. Als Populationsgröße wählen wir wie beim (1+1) GIGA die minimal mögliche Größe von 2 Individuen; die Population P besteht nur aus zwei Suchpunkten $P = \{x,y\}$.

In jedem Schritt führt der Algorithmus entweder eine Zwei-Punkt-Kreuzung aus oder eine Mutation. Dabei verwenden wir die Notation „Mutation(x)“ als eine abkürzende Schreibweise für die bekannte Mutation, die jedes Bit in x unabhängig von den anderen mit Mutationswahrscheinlichkeit $p_m := 1/n$ flippt.

Die Selektion zur Übernahme vergleicht die Population der Nachkommen als Ganzes mit der Population der Eltern und übernimmt die Nachkommen, falls die Fitness der Nachkommenpopulation nicht schlechter ist.

Algorithmus 5 ((2+2) GA mit Fitness Sharing).

1. *Wähle zwei Suchpunkte* $x, y \in \{0,1\}^n$ *uniform zufällig. Setze* $P := \{x, y\}$.
2. *Mit Wahrscheinlichkeit* $1/2$ *führe Schritt* $3a$ *aus, ansonsten Schritt* $3b$.

$3a$ $(x', y') :=$ *Zwei-Punkt-Kreuzung*(x, y). *Setze* $P' := \{x', y'\}$.

$3b$ $x' :=$ *Mutation*(x), $y' :=$ *Mutation*(y). *Setze* $P' := \{x', y'\}$.

4. *Falls* $f(P') \geq f(P)$, *setze* $P := P'$.
5. *Weiter bei 2.*

Der Selektionsoperator ist recht einfach strukturiert, da er die Populationen von Eltern und Nachkommen als Ganze miteinander vergleicht und somit stets ganze Populationen optimiert. Dies entspricht nicht der üblichen Vorgehensweise; meist wird zur Selektion eine Gesamtpopulation $\tilde{P} := P \cup P'$ aus Eltern und Nachkommen erstellt und die Fitness von Suchpunkten mit Fitness Sharing auf diese Gesamtpopulation $\tilde{P}$ bezogen. Dadurch können allerdings Verfälschungen auftreten: Es kann vorkommen, dass ein Elter x gemessen an der Elternpopulation P eine hohe Fitness $f(x, P)$ hat, aber in der Gesamtpopulation $\tilde{P}$ eine niedrige Fitness $f(x, \tilde{P})$ hat, falls die beiden Nachkommen, die neu ins Spiel kommen, dem Elter x ähnlich sind. Gegenüber solchen Selektionsstrategien, hat die hier gewählte Lösung den Vorteil, dass sowohl die Eltern als auch die Nachkommen immer in ihren jeweiligen Kontexten P bzw. P' erfasst werden.

Nachteil der gewählten Selektionsstrategie ist, dass es nicht möglich ist, eine neue Population zu erstellen, die sowohl Eltern als auch Nachkommen enthält; wir haben nur die Wahl zwischen beiden Eltern und beiden Nachkommen. Die folgende Analyse wird jedoch zeigen, dass dieser Nachteil nicht wesentlich ist.

Unser Algorithmus weicht in einem zweiten Punkt von einem üblichen Form genetischer Algorithmen ab. Und zwar ist es üblich, nach einer Rekombination die beiden Kinder anschließend zu mutieren. Wir verzichten hier darauf, da die folgende Analyse des Algorithmus auch ohne Mutation der erzeugten Kinder auskommt. Es wäre jedoch problemlos möglich, in Schritt $3a$ zusätzliche Mutationsoperatoren einzufügen. Die Analyse würde dadurch nicht wesentlich beeinträchtigt, da mit einer Wahrscheinlichkeit von ungefähr e^{-2} bei einer Mutation beider Kinder nach der Rekombination in beiden Kindern kein Bit geflippt wird.

Dies ist auch eine Rechtfertigung dafür, warum wir in Schritt $3b$ stets beide Suchpunkte der Population mutieren. Wenn der Algorithmus einen Schritt ausführen soll, bei dem nur ein Suchpunkt durch Mutation verändert wird, können wir darauf warten, dass eine passende Mutation dieses Suchpunkts eintritt und gleichzeitig im anderen Suchpunkt kein Bit flippt.

Vorarbeit zur Analyse des (2+2) GAs

Die Formeln aus Definition 16 lassen sich vereinfachen, wenn wir ausnutzen, dass die Population des (2+2) GAs nur aus zwei Individuen besteht.

Bei einer Population $P = \{x, y\}$ mit nur zwei Individuen gilt

$$\begin{aligned} Sh(x,P) &= \sum_{z\in\{x,y\}} (1 - H(x,z)/n) \\ &= (1 - H(x,x)/n) + (1 - H(x,y)/n) \\ &= 2 - H(x,y)/n. \end{aligned}$$

Für $Sh(y,P)$ erhalten wir analog ebenfalls den Term $2-H(x,y)/n$. Damit lässt sich die Fitness der Population $P = \{x, y\}$ wie folgt vereinfachen:

$$\begin{aligned} f(P) &= \frac{f(x)}{Sh(x,P)} + \frac{f(y)}{Sh(y,P)} \\ &= \frac{f(x)}{2 - H(x,y)/n} + \frac{f(y)}{2 - H(x,y)/n} \\ &= \frac{f(x) + f(y)}{2 - H(x,y)/n}. \end{aligned}$$

Für die Analyse des (2+2) GAs ist essenziell, dass die Fitness $f(P)$ der jeweils aktuellen Population über die Zeit monoton steigend ist, was leicht aus dem Selektionsoperator ersichtlich wird. Die maximale Fitness der Population ist der Wert $f(P) = 2n - 2$, der nur für die Population $P = \{0^n, 1^n\}$ angenommen wird. Die erwartete Zeit, bis die aktuelle Population einen global optimalen Suchpunkt enthält, ist nach oben beschränkt durch die erwartete Zeit, bis der (2+2) GA die Population $P = \{0^n, 1^n\}$ erreicht.

Um eine obere Schranke für die erwartete Optimierungszeit des (2+2) GAs zu zeigen, wollen wir im Folgenden Operationen ausfindig machen, die helfen können, die Fitness der jeweils aktuellen Population zu erhöhen. Dabei werden sich auf natürliche Weise unterschiedliche Szenarien ergeben, in denen unterschiedliche Operationen helfen können. Wichtig ist jedoch nur, dass es in allen Szenarien hilfreiche Operationen gibt.

Angenommen, eine Population P mit $f(P) < 2n-2$ liegt vor. Dann kann die Fitness der aktuellen Population erhöht werden, indem die Summe der „echten“ Fitnesswerte $f(x) + f(y)$ erhöht wird, oder indem der Hammingabstand $H(x,y)$ erhöht wird. Im Grunde liegt hier ein Optimierungsproblem mit den beiden Komponenten $f(x)+f(y)$ und $H(x,y)$ vor.

Im Allgemeinen können diese beiden Komponenten nicht unabhängig voneinander behandelt werden. In vielen Populationen gibt es keine effiziente Möglichkeit, beide Komponenten gleichzeitig zu verbessern. Oft müssen wir daher in Kauf nehmen, dass der Wert einer Komponente sinkt, damit der Wert der anderen Komponente steigt.

Besonders wahrscheinlich sind in diesem Fall 1-Bit-Mutationen, die eine Komponente verbessern und die andere Komponente eventuell verschlechtern. Eine Klasse von 1-Bit-Mutationen sind solche, die die Fitnesskomponente $f(x) + f(y)$ erhöhen, was

unter Umständen auf Kosten des Hammingabstands geht. Da jedoch nur ein Bit geflippt wird, ändert sich der Hammingabstand $H(x, y)$ nur um ± 1.

Die Frage, ob und in welchen Szenarien solche 1-Bit-Mutationen die Fitness der Population erhöhen können, werden wir später klären. Zunächst wollen wir herausfinden, wann 1-Bit-Mutationen existieren, die die Fitness-Komponente $f(x) + f(y)$ erhöhen können. Dazu formulieren wir folgendes Lemma.

Lemma 11. *Sei $G = (V, E)$ ein vollständiger binärer Baum mit $n := |V|$ Knoten und $x \in \{0,1\}^n$ eine Färbung mit $f(x) < (3n-3)/4$. Dann gibt es eine 1-Bit-Mutation, die aus x einen Suchpunkt mit höherer Fitness erzeugt.*

Beweis. Wir zeigen die Behauptung durch Kontraposition: Wir nehmen an, dass es keine verbessernde 1-Bit-Mutation ausgehend von x gibt. Eine verbessernde 1-Bit-Mutation gibt es genau dann, wenn es einen in x schlecht gefärbten Knoten gibt. Die Annahme, die wir zum Widerspruch führen wollen, ist daher, dass alle Knoten in x nicht schlecht gefärbt sind; für alle Knoten v gilt $\deg_x^+(v) \geq \deg(v)/2$. Aus dieser Annahme folgern wir dann $f(x) \geq (3n-3)/4$.

Die Wurzel, die Blätter und die übrigen inneren Knoten haben verschiedene Grade. Die Wurzel r hat Grad 2 und damit gilt $\deg_x^+(r) \geq 1$, alle Blätter w haben Grad 1 und damit gilt $\deg_x^+(w) = 1$. Schließlich haben alle übrigen Knoten v, nämlich alle inneren Knoten mit Ausnahme der Wurzel, Grad 3 und es gilt $\deg_x^+(v) \geq 2$.

Um die Zahl einfarbiger Kanten abzuschätzen, summieren wir die Werte $\deg_x^+(\cdot)$ für alle Knoten des Baums. Die Zahl der Blätter in einem vollständigen binären Baum mit n Knoten beträgt $(n+1)/2$, demnach gibt es neben der Wurzel noch $(n-3)/2$ andere innere Knoten. Insgesamt erhalten wir

$$\sum_{v \in V} \deg_x^+(v) \;\geq\; 1 \cdot 1 + \frac{n+1}{2} \cdot 1 + \frac{n-3}{2} \cdot 2 \;=\; \frac{3n-3}{2}.$$

Die Summe zählt alle einfarbigen Kanten genau zwei Mal, da eine einfarbige Kante $\{u, v\}$ in $\deg_x^+(u)$ und $\deg_x^+(v)$ eingeht. Daher ist $f(x) \geq (3n-3)/4$, Widerspruch zur Annahme $f(x) < (3n-3)/4$. □

Andererseits kann es auch 1-Bit-Mutationen geben, die den Hammingabstand um 1 erhöhen, was dann eventuell auf Kosten der Fitness geht.

Wir haben in Abschnitt 4.1.2 gesehen, dass die Ising-Funktion auf vollständigen binären Bäumen exponentiell viele lokale Optima hat. Daher ist es plausibel, dass Populationen erreicht werden, bei denen beide Suchpunkte auf der Fitnesslandschaft der Ising-Funktion (also die „echte“ Fitness ohne Fitness Sharing) lokale Optima darstellen. Eine interessante Frage ist daher: Ist der Effekt von Fitness Sharing groß genug, so dass ein lokales Optimum x effizient zu Gunsten eines weiter von y entfernten Suchpunkts verlassen werden kann?

Das folgende Theorem gibt eine Antwort auf diese Frage.

Theorem 18. *Gegeben ein beliebiger Baum $G = (V, E)$ mit $n := |V|$ Knoten und Tiefe $d := \log(n+1)-1$. Der (2+2) GA mit Fitness Sharing erreicht auf $f := \text{Ising}_G$ die global optimale Population $P = \{0^n, 1^n\}$ in erwarteter Zeit $O(n^3)$.*

Beweis. Wir nutzen aus, dass die Fitness der Population $f(P)$ über die Zeit monoton ist und suchen im Folgenden – analog zur Methode der Fitnessschichten – nach Operationen, die mit ausreichend großer Wahrscheinlichkeit auftreten und $f(P)$ um eine positive Konstante erhöhen.

Zuerst wollen wir ausrechnen, wie sich die Fitness der Population verändert, wenn sich beim Übergang von $P = \{x, y\}$ zu $P' = \{x', y'\}$ die Fitnesskomponente um $\Delta f := (f(x')+f(y'))-(f(x)+f(y))$ und der Hammingabstand um $\Delta H := H(x', y')-H(x, y)$ verändert.

Zur besseren Lesbarkeit schreiben wir kurz H für $H(x, y)$. Damit gilt

$$\begin{aligned}
& f(P') - f(P) \\
\overset{\text{Def.}}{=} \quad & \frac{f(x') + f(y')}{2 - H(x', y')/n} - \frac{f(x) + f(y)}{2 - H(x, y)/n} \\
= \quad & \frac{f(x) + f(y) + \Delta f}{2 - (H + \Delta H)/n} - \frac{f(x) + f(y)}{2 - H/n} \\
= \quad & \frac{(f(x) + f(y) + \Delta f)(2 - H/n)}{(2 - (H + \Delta H)/n)(2 - H/n)} - \frac{(f(x) + f(y))(2 - (H + \Delta H)/n)}{(2 - (H + \Delta H)/n)(2 - H/n)} \\
= \quad & \frac{2\Delta f - \Delta f \cdot H/n + \Delta H/n \cdot (f(x) + f(y))}{(2 - (H + \Delta H)/n)(2 - H/n)}.
\end{aligned}$$

Da die Hammingabstände $H(x, y) = H$ und $H(x', y') = H + \Delta H$ durch 0 nach unten beschränkt sind, gilt

$$(2 - (H + \Delta H)/n)(2 - H/n) \leq 4.$$

Falls der Zähler $2\Delta f - \Delta f \cdot H/n + \Delta H/n \cdot (f(x) + f(y))$ nicht negativ ist, gilt daher

$$\begin{aligned}
f(P') - f(P) \quad & \geq \quad \frac{2\Delta f - \Delta f \cdot H/n + \Delta H/n \cdot (f(x) + f(y))}{4} \\
& = \quad \frac{2n \cdot \Delta f - \Delta f \cdot H(x, y) + \Delta H \cdot (f(x) + f(y))}{4n} \\
& = \quad \frac{\Delta f \cdot (2n - H(x, y)) + \Delta H \cdot (f(x) + f(y))}{4n} \qquad (*)
\end{aligned}$$

In der Formel $(*)$ finden sich die beiden Komponenten $H(x, y)$ und $f(x) + f(y)$ wieder. Wenn die Differenz $f(P') - f(P)$ nicht negativ ist, ergibt sich der $f(P)$-Gewinn damit additiv aus den Termen der beiden Komponenten.

Es wird auch deutlich, dass die Erhöhung einer Komponente um Δf bzw. ΔH durch den Wert der jeweils anderen Komponente bestimmt wird. Je höher die Fitnesskomponente $f(x) + f(y)$, desto größer der Gewinn, wenn der Hammingabstand um $\Delta H > 0$ vergrößert wird. Bei der Erhöhung der Fitnesskomponente ist es genau

umgekehrt: Je kleiner der Hammingabstand $H(x,y)$, um so größer der Gewinn bei einer Erhöhung der Fitnesskomponente um $\Delta f > 0$.

In unterschiedlichen Szenarien für die beiden Komponenten $H(x,y)$ und $f(x)+f(y)$ können unterschiedliche Operationen helfen. Wir werden daher eine Fallunterscheidung durchführen und in jedem Fall effiziente Operationen ausfindig machen, die $f(P)$ erhöhen können.

Die erwartete Zeit, bis die Zielpopulation $P = \{0^n, 1^n\}$ erreicht ist, schätzen wir ab, indem wir die erwarteten Laufzeiten in den einzelnen Fällen abschätzen und diese Schranken am Ende addieren. Diese Methode ist auch als *Buchhaltermethode* bekannt.

Die erwartete Laufzeit innerhalb eines Falles schätzen wir durch die erwartete Zeit ab, die wir benötigen, um $f(P)$ um bis zu $2n-2$ Werte zu erhöhen. Anschaulich gehen wir damit in jedem Fall pessimistischerweise davon aus, dass dieser Fall die gesamte Arbeit allein leisten muss (bzw. den größtmöglichen Anteil daran leistet, falls dieser Fall nicht im gesamten Optimierungsprozess eintreten kann).

Fall 1: $H(x,y) = n$.

In diesem Fall haben wir wie beim (1+1) GIGA eine Population aus zwei komplementären Suchpunkten. Aus der Analyse des (1+1) GIGA wissen wir bereits, dass eine passende Zwei-Punkt-Kreuzung von x und $y = \overline{x}$ Nachkommen erzeugen kann, die eine um jeweils 1 höhere Fitness als ihre Eltern haben; es gilt $f(x') + f(y') = f(x) + f(y) + 2$.

Zweitens wissen wir, dass eine Zwei-Punkt-Kreuzung den Hammingabstand der Eltern bewahrt. Die beiden Komponenten ändern sich also um $\Delta f := 2$ und $\Delta H := 0$.

Wenn wir dies in Formel $(*)$ einsetzen, erhalten wir

$$\begin{aligned} f(P') - f(P) &\overset{(*)}{\geq} \frac{\Delta f \cdot (2n - H(x,y)) + \Delta H \cdot (f(x) + f(y))}{4n} \\ &= \frac{2 \cdot (2n-n) + 0 \cdot (f(x) + f(y))}{4n} = \frac{1}{2}. \end{aligned}$$

(Der Form halber sei angemerkt: Die Voraussetzung für die Anwendung von $(*)$ bereitet uns wenig Sorgen; in einer Ungleichungskette

$$f(P') - f(P) \overset{(*)}{\geq} \frac{\Delta f \cdot (2n - H(x,y)) + \Delta H \cdot (f(x) + f(y))}{4n} \overset{(**)}{\geq} 0$$

beweist Ungleichung $(**)$, dass $2\Delta f - \Delta f \cdot H(x,y)/n + \Delta H/n \cdot (f(x) + f(y)) \geq 0$ und damit die Voraussetzung für Ungleichung $(*)$ erfüllt ist.)

Die Wahrscheinlichkeit für einen Schritt, der $f(P)$ um eine positive Konstante erhöht, ist bei i zweifarbigen Kanten $1/2 \cdot i \cdot 1/\binom{n}{2}$. Der Faktor $1/2$ kommt daher, dass wir einen Schritt mit Zwei-Punkt-Kreuzung ausführen müssen. Die erwartete Zahl von Schritten die wir in Fall 1 zubringen, ist damit beschränkt durch

$$1 + \sum_{i=1}^{n-1} 2 \cdot \frac{1}{i} \cdot \binom{n}{2} = O(n^2) \cdot \sum_{i=1}^{n-1} \frac{1}{i} = O(n^2 \log n).$$

Fall 2: $H(x,y) < n$ und $H(x,y) + f(x) + f(y) < (3n-3)/2$.

In einem solchen Fall ist die Fitnesskomponente typischerweise klein, so dass eine Verringerung des Hammingabstands keine all zu großen negativen Auswirkungen auf die Fitness der Population hat. Daher warten wir auf eine 1-Bit-Mutation, die die Fitness auf Kosten des Hammingabstands erhöht.

Da $H(x,y) \geq 0$, folgt $f(x) + f(y) < (3n-3)/2$. Für einen der beiden Suchpunkte, o. B. d. A. für x, folgt $f(x) < (3n-3)/4$ und Lemma 11 besagt, dass es in x eine verbessernde 1-Bit-Mutation gibt.

Wenn nun Schritt $3b$ ausgeführt wird, in x eine verbessernde 1-Bit-Mutation stattfindet und in y kein Bit geflippt wird, erhöht sich die Fitnesskomponente um $\Delta f \geq 1$. Der Hammingabstand $H(x,y)$ verändert sich um $\Delta H = -1$ oder $\Delta H = 1$. Damit gilt

$$\begin{aligned} f(P') - f(P) &\overset{(*)}{\geq} \frac{\Delta f \cdot (2n - H(x,y)) + \Delta H \cdot (f(x) + f(y))}{4n} \\ &\geq \frac{2n - H(x,y) - f(x) - f(y)}{4n} \\ &\overset{\text{Vor.}}{>} \frac{2n - (3n-3)/2}{4n} > \frac{1}{8}. \end{aligned}$$

Die Wahrscheinlichkeit für einen solchen Schritt ist $\Omega(1/n)$ und da $f(P)$ nur $O(n)$ Mal um einen solchen Wert erhöht werden muss, ist die erwartete Zahl von Schritten in Fall 2 beschränkt durch $O(n^2)$.

Fall 3: $H(x,y) < n$ und $H(x,y) + f(x) + f(y) > (33/16) \cdot n$.

Im Vergleich zu Fall 2 haben wir hier den umgekehrten Fall: Die Fitnesskomponente ist typischerweise groß, so dass eine Erhöhung des Hammingabstands einen großen Gewinn bringt. Wir warten daher auf eine 1-Bit-Mutation, die den Hammingabstand auf Kosten der Fitness erhöht.

Allerdings darf die Fitness nicht zu stark absinken; wir möchten sicher stellen, dass die Fitnessveränderung $\Delta f \geq -1$ beträgt. Wir suchen daher einen Knoten v, der folgende Eigenschaften hat. Zum einen soll v in x und y gleich gefärbt sein, $x_v = y_v$, damit eine Mutation von v in einem der beiden Suchpunkte den Hammingabstand $H(x,y)$ vergrößern kann. Zum anderen soll die lokale Fitnessveränderung an v (die Veränderung der Fitness, wenn nur das Bit des Knotens v flippt) in einem der beiden Suchpunkte mindestens -1 betragen: $\ell(x, x_v) \geq -1$ oder $\ell(y, y_v) \geq -1$.

Da $H(x,y) < n$, gibt es einen Knoten u mit $x_u = y_u$. Falls u ein Blatt ist oder ein innerer Knoten, an dem in x oder y mindestens eine Kante zweifarbig ist, wählen wir $v := u$ und sind fertig. Ansonsten haben beide Kinder von u die gleiche Farbe wie u. Wir wählen ein Kind w von u, für das dann gilt: $x_w = x_u = y_u = y_w$ und führen unsere Betrachtungen analog fort für w statt u. Spätestens wenn auf diese Weise ein Blatt erreicht wird, finden wir einen passenden Knoten v.

Sei v also ein Knoten mit $x_v = y_v$ und o. B. d. A. in x einer lokalen Fitnessveränderung von $\ell(x, x_v) \geq -1$. Wenn Schritt $3b$ ausgeführt wird, eine 1-Bit-Mutation in x das Bit x_v flippt und kein Bit in y geflippt wird, gilt $\Delta f \geq -1$ und $\Delta H = 1$. Die Fitness der Population ändert sich um

$$\begin{aligned} f(P') - f(P) &\overset{(*)}{\geq} \frac{\Delta f \cdot (2n - H(x,y)) + \Delta H \cdot (f(x) + f(y))}{4n} \\ &\geq \frac{-2n + H(x,y) + f(x) + f(y)}{4n} \\ &\overset{\text{Vor.}}{>} \frac{-2n + (33/16 \cdot n)}{4n} = \frac{1}{64}. \end{aligned}$$

Dies beantwortet (zumindest teilweise) die Frage, ob mit Fitness Sharing lokale Optima effizient wieder verlassen werden können.

Die Wahrscheinlichkeit für einen solchen Schritt ist $\Omega(1/n)$ und da $f(P)$ nur $O(n)$ Mal um einen solchen Wert erhöht werden muss, ist die erwartete Zahl von Schritten in Fall 3 beschränkt durch $O(n^2)$.

Fall 4: $H(x,y) < n$ und $(3n-3)/2 \leq H(x,y) + f(x) + f(y) \leq (33/16) \cdot n$.

Die Rechnungen aus Fall 2 und Fall 3 zeigen: Je kleiner $H(x,y) + f(x) + f(y)$, desto größer der $f(P)$-Gewinn durch Mutationen, die die Fitnesskomponente um 1 erhöhen und den Hammingabstand um 1 verringern. Je größer $H(x,y)+f(x)+f(y)$, desto größer der $f(P)$-Gewinn durch Mutationen, die den Hammingabstand um 1 erhöhen und die Fitnesskomponente um 1 senken. Beide Typen von Mutationen sind gewissermaßen komplementär zueinander; es kann stets nur ein Typ die Fitness der Population erhöhen.

Bei „mittleren“ Werten für $H(x,y) + f(x) + f(y)$ bringen beide Typen von Mutationen nur einen geringen $f(P)$-Gewinn; bei $H(x,y) + f(x) + f(y) = 2n$ wird mit $f(P') - f(P) = 0$ sogar eine Nullstelle für den $f(P)$-Gewinn beider Typen von Mutationen erreicht. Im Fall $H(x,y) + f(x) + f(y) = 2n$ können also Mutationen, die eine Komponente um 1 erhöhen und die andere Komponente um 1 senken, nicht weiterhelfen.

Daher wollen wir nun andere, komplexere Operationen finden, die die Fitness der Population erhöhen können. Dies können zum einen Operationen sein, mit denen eine Komponente erhöht und die andere Komponente nicht gesenkt wird. Da in Fall 4 gilt: $f(x) + f(y) = \Omega(n)$, sind in Ungleichung $(*)$ beide Δ-Werte im Zähler mit linear großen Termen gewichtet, so dass der $f(P)$-Gewinn durch solch eine Operation stets $\Omega(1)$ beträgt. Zum anderen können Operationen, mit denen eine Komponente um mehr als 1 erhöht wird, in bestimmten Szenarien einen Verlust von 1 in der anderen Komponente ausgleichen, so dass wir insgesamt wieder einen $f(P)$-Gewinn von $\Omega(1)$ erhalten.

Es können folgende Szenarien eintreten, bei denen günstige Operationen $f(P)$ jeweils um eine positive Konstante erhöhen.

Szenario 1: Es gibt einen Knoten v, bei dem eine 1-Bit-Mutation in x oder y die Fitness um mindestens 2 verbessert: $\ell(x, x_v) \geq 2$ oder $\ell(y, y_v) \geq 2$.

O. B. d. A. sei $\ell(x, x_v) \geq 2$. Wenn Schritt $3b$ ausgeführt wird, eine 1-Bit-Mutation x_v flippt und in y kein Bit flippt, erhöht sich die Fitness um $\Delta f \geq 2$. Der Hammingabstand $H(x, y)$ verändert sich um $\Delta H = -1$ oder $\Delta H = 1$. Damit gilt

$$\begin{aligned} f(P') - f(P) &\overset{(*)}{\geq} \frac{\Delta f \cdot (2n - H(x,y)) + \Delta H \cdot (f(x) + f(y))}{4n} \\ &\geq \frac{4n - 2H(x,y) - f(x) - f(y)}{4n} \\ &= \frac{4n - (H(x,y) + f(x) + f(y)) - H(x,y)}{4n} \\ &\overset{\text{Vor.}}{\geq} \frac{4n - (33/16) \cdot n - H(x,y)}{4n} > \frac{15}{64}. \end{aligned}$$

Die Wahrscheinlichkeit für eine bestimmte Operation dieser Art ist $\Omega(1/n)$.

Szenario 2: Es gibt einen Teilbaum $T(v)$, der sowohl in x als auch in y ein verbessernder Teilbaum ist und dessen Wurzel v in x und y komplementär gefärbt ist: $x_v \neq y_v$.

Dann warten wir auf eine Zwei-Punkt-Kreuzung, die genau den verbessernden Teilbaum $T(v)$ abgrenzt. Wir haben bereits in den Vorüberlegungen zu diesem Abschnitt gesehen, dass dadurch die zuvor zweifarbige Kante von v zu seinem Elter in beiden Nachkommen jeweils einfarbig gefärbt wird und insgesamt $f(x') + f(y') = f(x) + f(y) + 2$ gilt (die einfarbigen Kanten innerhalb von $T(v)$ werden lediglich zwischen den beiden Suchpunkten ausgetauscht). Weiter bewahrt eine Zwei-Punkt-Kreuzung den Hammingabstand. Damit ändert sich die Fitness um $\Delta f = 2$ und der Hammingabstand um $\Delta H = 0$, somit erhöht sich $f(P)$ um einen Wert $\Omega(1)$.

Die Wahrscheinlichkeit für eine bestimmte Operation dieser Art ist $\Omega(1/n^2)$.

Szenario 3: Es gibt einen Teilbaum $T(u)$ der Höhe 1 (also zwei Blätter mit ihrem gemeinsamen Elter) mit Wurzel u und Kindern v, w, der nicht in x und y einheitlich und mit verschiedenen Farben gefärbt ist, für den also gilt:

$$\neg(x_u = x_v = x_w \neq y_u = y_v = y_w).$$

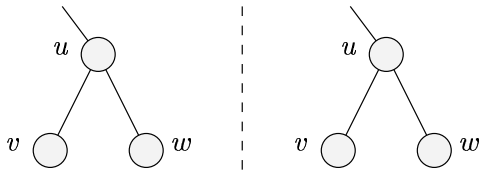
Eine solche Belegung lässt Raum für Verbesserung, da eine Mutation die Bits x_u, x_v, x_w und y_u, y_v, y_w in eine Belegung überführen kann, bei der die Knoten in $T(u)$ in x' und y' jeweils einheitlich und mit verschiedenen Farben gefärbt sind. Wenn wir uns nur auf $T(u)$ beschränken, maximiert eine solche Belegung mit $x_u = x_v = x_w \neq y_u = y_v = y_w$ sowohl die Fitnesskomponente als auch den Hammingabstand innerhalb von $T(u)$. Daher bezeichnen wir eine solche Belegung als lokal optimale Belegung.

Es gibt zwei zueinander komplementäre lokal optimale Belegungen der Bits x_u, x_v, x_w und y_u, y_v, y_w. Ausgehend von einer beliebigen Belegung dieser Bits

ist der minimale Hammingabstand zu einer der beiden optimalen Belegungen höchstens 3. Es genügen also Mutationen von höchstens 3 Bits (in x und y zusammen), um aus einer beliebigen nicht lokal optimalen Belegung eine der beiden lokal optimalen Belegungen zu erzeugen.

Es bleibt zu zeigen, dass sich $f(P)$ um einen Wert $\Omega(1)$ erhöht, wenn eine Mutation eine beliebige nicht lokal optimale Belegung in eine lokal optimale Belegung überführt. Wir unterscheiden mehrere Fälle von Belegungen, die wir durch Mutationen von insgesamt höchstens 3 Bits auf direktem Wege in eine lokal optimale Belegung überführen wollen. Dabei machen wir keine Aussage über die Färbung der Kante von u zum Elter von u; wir nehmen stets pessimistischerweise an, dass diese Kante in einem Suchpunkt $z \in \{x, y\}$ zweifarbig gefärbt wird, wenn z_u geflippt wird.

Szenario $3a$**:** Der Teilbaum $T(u)$ ist in x und y einheitlich mit der gleichen Farbe gefärbt, wie in folgender Skizze dargestellt:

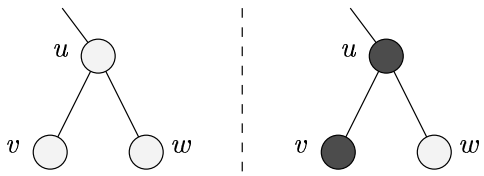


Färbung in x Färbung in y

Eine Mutation, die in x und y zusammen genau die Bits x_u, x_v und x_w flippt, erhöht den Hammingabstand um $\Delta H = 3$, während die Fitnesskomponente sich um $\Delta f \geq -1$ verändert (die Kante zum Elter von u kann in x zweifarbig werden). Es gilt

$$\begin{aligned} f(P') - f(P) &\overset{(*)}{\geq} \frac{\Delta f \cdot (2n - H(x,y)) + \Delta H \cdot (f(x) + f(y))}{4n} \\ &\geq \frac{-2n + H(x,y) + 3(f(x) + f(y))}{4n} \\ &= \frac{-2n + 3(H(x,y) + f(x) + f(x)) - 2H(x,y)}{4n} \\ &\overset{\text{Vor.}}{\geq} \frac{-2n + 3(3n-3)/2 - 2H(x,y)}{4n} \\ &= \frac{(5/2)n - 9/2 - 2H(x,y)}{4n} \geq \frac{1}{8} - O(n^{-1}). \end{aligned}$$

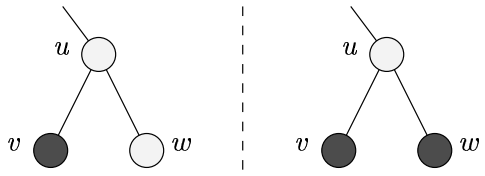
Szenario $3b$**:** Der Teilbaum $T(u)$ ist o. B. d. A. in x einheitlich gefärbt, in y jedoch nicht. Folgende Skizze zeigt ein Beispiel für dieses Szenario, in dem eine Mutation von y_w den Wert $f(P)$ erhöht.



Färbung in x Färbung in y

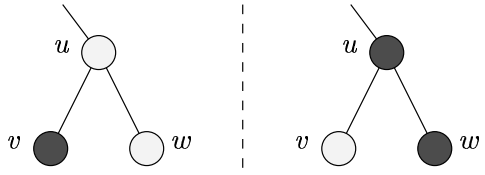
In y ist mindestens eine Kante in $T(u)$ zweifarbig und der Hammingabstand der Bitstrings (x_u, x_v, x_w) und (y_u, y_v, y_w) ist kleiner als 3, da ansonsten eine lokal optimale Belegung vorläge. Daher gibt es eine Mutation von höchstens 2 Bits in y, die den Teilbaum $T(u)$ in y' einheitlich färbt. Da es vorher in y eine zweifarbige Kante in $T(u)$ gab, wird die Fitnesskomponente nicht gesenkt, selbst wenn die Kante von u zum Elter in y' zweifarbig wird. Der Hammingabstand jedenfalls wird um mindestens 1 erhöht; es gilt $\Delta f \geq 0$ und $\Delta H \geq 1$ und insgesamt steigt $f(P)$ um $\Omega(1)$.

Szenario $3c$**:** Der Teilbaum $T(u)$ ist in x und y nicht einheitlich gefärbt. Falls der Hammingabstand der Bitstrings (x_u, x_v, x_w) und (y_u, y_v, y_w) kleiner als 3 ist, gibt es wie in Szenario $3b$ eine Mutation, die die Fitness nicht senkt, den Hammingabstand aber erhöht. Hier ein Beispiel für eine solche Konstellation, bei der eine Mutation von x_v und y_u den Wert $f(P)$ erhöht.



Färbung in x Färbung in y

Falls der Hammingabstand der Bitstrings (x_u, x_v, x_w) und (y_u, y_v, y_w) genau 3 beträgt, muss der Algorithmus den Hammingabstand bewahren, um zu einer lokal optimale Belegung zu gelangen. Wir betrachten daher nur Mutationen, bei denen alle flippenden Bits in x und y synchron flippen, d. h. genau dann, wenn x_i flippt, muss auch y_i flippen, $i \in \{u, v, w\}$. Der Hammingabstand bleibt bei einer synchronen Mutation konstant. Ein Beispiel für eine solche Konstellation, bei dem x_v und y_v flippen müssen, sieht aus wie folgt:



Färbung in x Färbung in y

Wenn alle flippenden Bits synchron flippen, ist die Zahl flippender Bits gerade. Da wir nur Mutationen von höchstens 3 Bits betrachten, kommen nur noch drei Möglichkeiten in Frage, wie in diesem Fall die nächste lokal optimale Färbung auf direktem Wege erreicht werden kann. Und zwar wird die nächste lokal optimale Färbung erzeugt, indem genau eine der folgenden Mengen von Bits geflippt wird: $\{x_u, y_u\}$, $\{x_v, y_v\}$ und $\{x_w, y_w\}$. Man kann leicht überprüfen, dass sich in diesen drei Unterfällen die Fitnesskomponente um mindestens 2 erhöht. Flippen die Bits eines Blattes, werden zweifarbige Kanten innerhalb von $T(u)$ einfarbig gefärbt (die Färbung von u wird nicht angetastet, daher bleibt die Kante zum Elter

von u unberührt). Flippen die Bits der Wurzel u, gibt es 4 zweifarbige Kanten innerhalb von $T(u)$, die durch eine Mutation von $\{x_u, y_u\}$ einfarbig werden. Selbst wenn die Kante von u zum Elter in beiden Suchpunkten zweifarbig wird, erhöht sich die Fitnesskomponente um 2.

Da der Hammingabstand konstant bleibt, gilt $\Delta f \geq 2$ und $\Delta H = 0$ und $f(P)$ erhöht sich um $\Omega(1)$.

Insgesamt haben wir nun gezeigt, dass $f(P)$ in Szenario 3 um eine positive Konstante erhöht werden kann, wenn eine nicht lokal optimale Färbung eines Teilbaums der Höhe 1 in eine lokal optimale Färbung überführt wird.

Die Wahrscheinlichkeit für eine bestimmte Operation dieser Art ist $\Omega(1/n^3)$; der worst case wird in Szenario $3a$ angenommen, in dem wir auf eine 3-Bit-Mutation warten.

Mit den Szenarien 1 – 3 haben wir drei beispielhafte Szenarien betrachtet, in denen verschiedene Operationen helfen können, $f(P)$ zu erhöhen. Solche Operationen bezeichnen wir im Folgenden als verbessernde Operationen. Um diese Erkenntnisse zum Beweis einer oberen Schranke für die erwartete Zeit in Fall 4 einzusetzen, müssen wir nun zeigen, dass solche verbessernden Operationen in einer gegebenen Population tatsächlich existieren.

Die drei Szenarien präsentieren uns nur einen Ausschnitt aus der Liste aller möglichen verbessernden Operationen. Bei der Vielfalt der Szenarien 1 – 3 ist es plausibel, dass stets einige dieser Szenarien zutreffen und es somit stets einige verbessernde Operationen gibt. Zumindest gilt dies, so lange die Summe der Komponenten $H(x,y)+f(x)+f(y)$ nicht zu groß wird, denn je größer die Werte der Komponenten werden, um so stärker werden die Möglichkeiten zur Verbesserung eingeschränkt.

Wir zeigen nun, dass es in Fall 4 in mindestens einem der Szenarien 1 – 3 mindestens $\Omega(n)$ verschiedene verbessernde Operationen gibt. Diesen Beweis führen wir durch Kontraposition: Wir nehmen an, dass es in allen drei Szenarien höchstens $o(n)$ verbessernde Operationen gibt und zeigen dann, dass dann die Summe der Komponenten $H(x,y)+f(x)+f(y)$ größer als $(33/16)\cdot n$ wird, Widerspruch zur Bedingung von Fall 4.

Die Ideen hinter der folgenden Argumentation sind einfach. Wenn es in Szenario 3 nur wenige nicht lokal optimale Teilbäume der Höhe 1 gibt, gibt es viele bereits lokal optimal gefärbte Teilbäume der Höhe 1. Dies lässt auf einen hohen Hammingabstand und eine hohe Zahl einfarbiger Kanten auf den untersten beiden Ebenen schließen. Hinzu kommt: Wenn es in den Szenarien 1 und 2 nur wenige verbessernde Operationen gibt, können auch auf den höheren Ebenen nicht alle Kanten zweifarbig sein. Diese Überlegungen wollen wir nun formalisieren.

Es gibt insgesamt $(n+1)/4$ Teilbäume der Höhe 1. Wenn es in Szenario 3 nur $o(n)$ verbessernde Operationen gibt, gibt es mindestens $(n+1)/4 - o(n)$ lokal optimal gefärbte Teilbäume der Höhe 1. Da jeder lokal optimal gefärbte Teilbaum den Wert

3 zum Hammingabstand $H(x, y)$ beiträgt, gilt

$$H(x,y) \geq 3 \cdot \left(\frac{n+1}{4} - o(n) \right) = \frac{3}{4} \cdot n - o(n).$$

Die Fitnesskomponente schätzen wir wie folgt nach unten ab. Sei $E_{i,j} \subseteq E$ für $i < j$ die Menge der Kanten $\{u, v\} \in E$, für die sowohl u als auch v auf Ebenen im Intervall $[i, j]$ liegen. Dann lässt sich die Kantenmenge E partitionieren in die disjunkten Teilmengen $E_{d-1,d}, E_{d-2,d-1}$ und $E_{0,d-2}$ wobei d die Tiefe des Baumes ist. Die Fitnesskomponente ergibt sich dann als Summe der Fitnessbeiträge der einzelnen Kantenmengen in x und y.

Die Kanten in $E_{d-1,d}$, die in lokal optimal gefärbten Teilbäumen verlaufen, sind in x und y einfarbig. Pro Teilbaum sind dies 2 Kanten; die Zahl der einfarbigen Kanten in $E_{d-1,d}$ ist demnach in beiden Suchpunkten jeweils $(n+1)/2 - o(n)$. Es sei daran erinnert, dass wir mit $f(x, E')$ die Zahl in x einfarbiger Kanten in der Menge E' notieren. Damit gilt

$$f(x, E_{d-1,d}) + f(y, E_{d-1,d}) \;\geq\; 2 \cdot \left(\frac{n+1}{2} - o(n) \right) \;=\; n - o(n).$$

Hinzu kommt Folgendes: Gegeben eine Kante $\{u, v\} \in E_{d-2,d-1}$, deren unterer Endpunkt v die Wurzel eines lokal optimal gefärbten Teilbaums der Höhe 1 ist. Falls die Kante $\{u, v\}$ sowohl in x als auch in y zweifarbig ist, haben wir mit $T(v)$ einen Teilbaum aus Szenario 2 gefunden, da $T(v)$ in x und y ein verbessernder Teilbaum ist und die Wurzel v in x und y jeweils komplementär gefärbt ist. Ein Beispiel für eine solche Situation ist in Abbildung 4.12 dargestellt. Ist $T(v)$ kein Teilbaum aus Szenario 2, muss die Kante $\{u, v\}$ daher in x oder in y einfarbig sein.

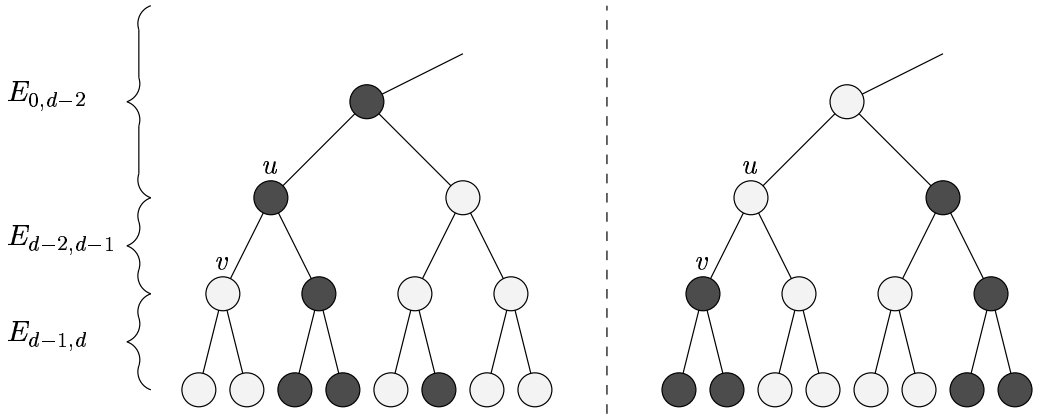


Abbildung 4.12: Eine Situation, in der eine Kante $\{u, v\}$, die zu einem lokal optimal gefärbten Teilbaum $T(v)$ führt, in beiden Suchpunkten zweifarbig ist.

Die Zahl der Kanten in $E_{d-2,d-1}$, deren untere Endpunkte Wurzeln von lokal optimal gefärbten Teilbäumen der Höhe 1 sind, ist gleich der Zahl lokal optimal gefärbter

Teilbäume der Höhe 1, hier also $(n+1)/4-o(n)$. Wenn es in Szenario 2 höchstens $o(n)$ verbessernde Operationen geben darf, gilt für $(n+1)/4-o(n)-o(n) = n/4-o(n)$ Kanten in $E_{d-2,d-1}$, dass sie in x oder in y einfarbig sein müssen. Wir erhalten

$$f(x, E_{d-2,d-1}) + f(y, E_{d-2,d-1}) \geq \frac{n}{4} - o(n).$$

Es bleiben noch die Kanten auf den höher gelegenen Ebenen, der Menge $E_{0,d-2}$. Jeder Knoten, der nur zu Kanten aus $E_{0,d-2}$ inzident ist, sprich jeder Knoten auf den Ebenen $0, \ldots, d-3$, hat einen Grad von mindestens 2. Falls ein solcher Knoten v in einem Suchpunkt $z \in \{x, y\}$ zu keiner einfarbigen Kante adjazent ist, gilt $\ell(z, z_v) \geq 2$ und wir haben eine verbessernde Operation aus Szenario 1 gefunden. Da es nach Annahme nur $o(n)$ verbessernde Operationen in Szenario 1 gibt, dürfen höchstens $o(n)$ dieser Knoten zu keiner einfarbigen Kante adjazent sein.

Sei V' die Menge der Knoten v auf den Ebenen $0, \ldots, d-3$, die zu mindestens einer in $z \in \{x, y\}$ einfarbigen Kante inzident sind, sprich für die $\deg_z^+(v) \geq 1$ gilt. Da es $(n+1)/2$ Blätter, $(n+1)/4$ Knoten auf Ebene $d-1$ und $(n+1)/8$ Knoten auf Ebene $d-2$ gibt, ist die Zahl der Knoten in V'

$$|V'| = \left(n - \frac{n+1}{2} - \frac{n+1}{4} - \frac{n+1}{8}\right) - o(n) = \frac{n}{8} - o(n).$$

Damit gilt

$$\sum_{v \in V'} \deg_z^+(v) \geq \frac{n}{8} - o(n).$$

Da jede in z einfarbige Kante höchstens 2 zu dieser Summe beiträgt, ist die Zahl einfarbiger Kanten in z in der Menge $E_{0,d-2}$ mindestens $|V'|/2 = n/16 - o(n)$. In x und y zusammen sind auf den genannten Ebenen also mindestens $n/8 - o(n)$ Kanten einfarbig:

$$f(x, E_{0,d-2}) + f(y, E_{0,d-2}) \geq \frac{n}{8} - o(n).$$

Wenn wir die Fitnessbeiträge aus den drei Kantenmengen $E_{d-1,d}, E_{d-2,d-1}$ und $E_{0,d-2}$ zusammenrechnen, erhalten wir folgende untere Schranke für die Fitnesskomponente $f(x) + f(y)$:

$$\begin{aligned} f(x) + f(y) &= \sum_{z \in \{x,y\}} (f(z, E_{0,d-2}) + f(z, E_{d-2,d-1}) + f(z, E_{d-1,d})) \\ &\geq n - o(n) + \frac{n}{4} - o(n) + \frac{n}{8} - o(n) \\ &= \frac{11}{8} \cdot n - o(n). \end{aligned}$$

Insgesamt gilt für die Summe beider Komponenten

$$H(x, y) + f(x) + f(y) \geq \frac{3}{4} \cdot n - o(n) + \frac{11}{8} \cdot n - o(n) = \frac{17}{8} \cdot n - o(n)$$

und für ausreichend große n ist dies größer als $(33/16)\cdot n$, Widerspruch zur Bedingung von Fall 4.

Wir haben also gezeigt, dass es in mindestens einem Szenario $\Omega(n)$ verbessernde Operationen geben muss. Eine bestimmte verbessernde Operation tritt mit Wahrscheinlichkeit $\Omega(1/n^3)$ auf; die Wahrscheinlichkeit, dass irgendeine verbessernde Operation auftritt, ist daher $\Omega(1/n^2)$. Da der Wert $f(P)$ durch eine verbessernde Operation um eine positive Konstante erhöht wird und dies nur $O(n)$ Mal geschehen muss, ist die erwartete Zeit in Fall 4 beschränkt durch $O(n^3)$.

Damit ist die Analyse von Fall 4 beendet. Wir addieren nun die erwarteten Zeiten, die wir in den vier Fällen zubringen, bevor die Population $P = \{0^n, 1^n\}$ erreicht wird:

$$O(n^2 \log n) + O(n^2) + O(n^2) + O(n^3) = O(n^3).$$

Daher erreichen wir in erwarteter Zeit $O(n^3)$ die Population $P = \{0^n, 1^n\}$. □

Ergänzungen und Diskussion der Ergebnisse

Der Beweis gibt keine Auskunft darüber, zu welcher Zeit eines Laufs welcher der vier Fälle typischerweise gerade aktuell ist. Experimente zeigen, dass der (2+2) GA typischerweise am Anfang eines Laufs die Fitness erhöht, auch wenn dies den Hammingabstand senkt (Fall 2). Irgendwann geht der Algorithmus dazu über, den Hammingabstand der beiden Individuen zu vergrößern; hier kann man in manchen Läufen beobachten, dass die Fitness dabei deutlich absinkt (Fall 3). Wenn einmal eine Population mit komplementären Suchpunkten erreicht wird, bleibt der Algorithmus typischerweise bei komplementären Suchpunkten und wir warten auf passende Zwei-Punkt-Kreuzungen, um die optimale Population zu erreichen (Fall 1).

Den Effekt, dass Fall 3 erst nach Fall 2 eintritt, können wir auch rechnerisch belegen. Ausgangspunkt ist die Ungleichung $H(x,y)+f(x)+f(y) \leq 2n$, also der Fall, dass die Summe der Komponenten, mit der die Fälle 2 – 4 unterschieden werden, unter dem Wert $2n$ liegt. Wir erinnern uns daran, dass der Wert $2n$ eine Nullstelle für $f(P)$-Gewinne bei einfachen 1-Bit-Mutationen war, die eine Komponente um 1 erhöhen und die andere Komponente um 1 senken.

Aus der Definition von $f(P)$ folgt

$$f(P) = \frac{f(x)+f(y)}{2 - H(x,y)/n} \;\Leftrightarrow\; f(x)+f(y) = f(P)\cdot(2 - H(x,y)/n).$$

Damit gilt

$$\begin{aligned} H(x,y) + f(x) + f(y) &\leq 2n \\ \Leftrightarrow H(x,y) + f(P)\cdot(2 - H(x,y)/n) &\leq 2n \\ \Leftrightarrow f(P)\cdot(2 - H(x,y)/n) &\leq 2n - H(x,y) \\ \Leftrightarrow f(P) &\leq \frac{2n - H(x,y)}{2 - H(x,y)/n} \\ \Leftrightarrow f(P) &\leq n. \end{aligned}$$

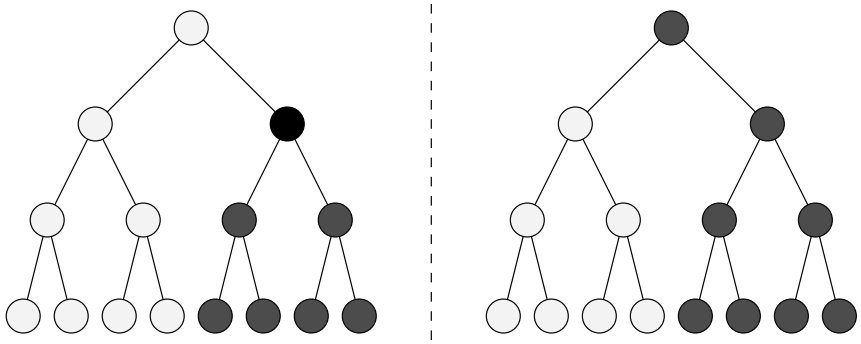
Da $f(P)$ über die Zeit monoton ist, tritt also Fall 2 aufgrund der Voraussetzung $H(x,y)+f(x)+f(y) < (3n-3)/2 \leq 2n$ stets vor Fall 3 mit Voraussetzung $H(x,y)+f(x)+f(y) > (33/16)\cdot n > 2n$ ein.

Der komplizierte Fall 4 mit mittleren Werten für $H(x,y)+f(x)+f(y)$ lässt sich allerdings nicht so leicht einordnen, da die Summe der Komponenten i. A. nicht monoton in $f(P)$ ist; 1-Bit-Mutationen, die eine Komponente um 1 erhöhen und die andere Komponente um 1 senken, können $f(P)$ erhöhen, während die Summe der Komponenten konstant bleibt.

Eine andere Frage ist die, ob die Schranke $O(n^3)$ asymptotisch scharf ist. Experimente zeigen, dass der (2+2) GA typischerweise recht schnell eine Population mit komplementären Suchpunkten erreicht und dann den größten Teil der Laufzeit dafür aufbringt, in Fall 1 auf die passenden Zwei-Punkt-Kreuzungen zu warten. Dies lässt vermuten, dass die Schritte, die wir in Fall 1 zubringen, die erwartete Laufzeit dominieren. Wenn dies der Fall ist, wäre die erwartete Laufzeit sogar durch $O(n^2 \log n)$ beschränkt statt durch $O(n^3)$.

Hypothese 2. *Gegeben ein beliebiger Baum $G=(V,E)$ mit $n := |V|$ Knoten. Die erwartete Zeit, bis der (2+2) GA mit Fitness Sharing auf* Ising_G *die global optimale Population $P = \{0^n, 1^n\}$ erreicht, ist nach oben beschränkt durch $O(n^2 \log n)$.*

Der kubische Term in unserer Schranke rührt daher, dass wir in Fall 4 im Szenario $3a$ auf 3-Bit-Mutationen warten und die Wahrscheinlichkeit einer Fitnesserhöhung dadurch vergleichsweise klein wird. In bestimmten Szenarien ist es jedoch nötig, auf solche 3-Bit-Mutationen zurückzugreifen. Abbildung 4.13 zeigt ein Beispiel für eine solche Population $P = \{x, y\}$. In x sind beide Teilbäume der Wurzel einheitlich und mit verschiedenen Farben gefärbt; der Suchpunkt y ergibt sich aus x, indem nur die Wurzel geflippt wird.



Färbung in x Färbung in y

Abbildung 4.13: Ein Beispiel für eine Population $P = \{x, y\}$, bei der die wahrscheinlichste Operation zur Erhöhung von $f(P)$ eine Operation aus Szenario $3a$ ist, bei dem ein Teilbaum der Höhe 1 komplett flippt.

In dieser Beispielpopulation liegt die Summe der Komponenten mit $H(x,y)+f(x)+f(y) = 2n-3$ knapp unter der „kritischen“ Schranke $2n$. Damit können 1-Bit-Mutationen, die den Hammingabstand um 1 erhöhen und die Fitnesskomponente

senken, hier nicht helfen. Andere 1-Bit-Mutationen, die die Fitness erhöhen, existieren in dieser Situation nicht.

Auch Zwei-Punkt-Kreuzungen können hier nichts ausrichten. In Bitstring-Darstellung geschrieben, sehen die Färbungen von x und y aus wie folgt. Mit der Preorder-Codierung steht an erster Stelle die Farbe der Wurzel r, danach folgen die Farben der $(n-1)/2$ Knoten des linken Teilbaums von r, gefolgt von den Farben der $(n-1)/2$ Knoten des rechten Teilbaums von r. Es ergeben sich folgende Darstellungen:

$$\begin{array}{cccc} 0 & 0\ldots0 & 1\ldots1 \\ 1 & \underbrace{0\ldots0}_{(n-1)/2} & \underbrace{1\ldots1}_{(n-1)/2} \end{array}$$

Man kann leicht sehen, dass bei allen $\binom{n}{2}$ Wahlmöglichkeiten für die beiden Kreuzungspunkte stets wieder die gleiche Population erzeugt wird, $P' = P$. Daher sind Zwei-Punkt-Kreuzungen in dieser Situation wirkungslos.

In der Population aus Abbildung 4.13 besteht die wahrscheinlichste verbessernde Operation darin, ganze Teilbäume der Höhe 1 zu flippen. Zumindest in einer Situation ist es daher für den Beweis einer guten oberen Schranke notwendig, auf 3-Bit-Mutationen zu warten. Offen bleibt die Frage, wie oft solche Situationen auftreten können. Im Beweis von Theorem 18 nehmen wir implizit an, dass solche Situationen linear oft auftreten können. Ein möglicher Ansatz zur Verbesserung der oberen Schranke wäre, zu zeigen, dass solche Situationen sehr viel seltener auftreten.

Nebenbei sei bemerkt: Das Beispiel aus Abbildung 4.13 zeigt, dass es notwendig ist, Mutationen mehrerer Bits zuzulassen. Wenn wir den Mutationsoperator einschränken, so dass in x und y jeweils nur höchstens ein Bit geflippt werden kann, kann die Population aus Abbildung 4.13 nicht mehr verändert werden: Es gibt keine 0- oder 1-Bit-Mutationen von x und y, die zu einer anderen akzeptierten Population führen, und Zwei-Punkt-Kreuzungen sind hier, wie gesehen, nutzlos.

Fazit

Auch wenn die Schranke $O(n^3)$ für den (2+2) GA vermutlich nicht scharf ist, liegt sie dennoch überraschend nah an der oberen Schranke für den (1+1) GIGA. Anders als befürchtet, hat der (2+2) GA keinerlei Probleme mit Populationen, die zwei suboptimale lokale Optima enthalten. Entweder es gibt Zwei-Punkt-Kreuzungen, die diese Färbungen verbessern können, oder durch Mutationen und den Einfluss von Fitness Sharing wird ein lokal optimaler Suchpunkt x durch einen anderen Suchpunkt ersetzt, der einen größeren Hammingabstand von y hat.

Letztere Mutationen können sogar dann akzeptiert werden, wenn der Funktionswert $f(x)$ dadurch gesenkt wird; dies ist bei der gegebenen Fitnessfunktion essenziell, da die Ising-Funktion auf vollständigen binären Bäumen exponentiell viele lokale Optima enthält und es daher fast unvermeidlich ist, dass der Algorithmus auf lokale Optima trifft.

Wenn wir das Resultat von Theorem 18 in den Kontext der Diskussion um Mutation versus Rekombination und die Resultate von Wegener [13] und Storch und Wegener [19] einordnen, haben wir eine neue Funktion gefunden, für die Rekombination beweisbar essenziell ist. Ein einfacher genetischer Algorithmus mit Zwei-Punkt-Kreuzung hat eine polynomielle erwartete Optimierungszeit, während die erwartete Optimierungszeit des (1+1) EA bei jeder Mutationswahrscheinlichkeit exponentiell ist.

Die Ising-Funktion auf vollständigen binären Bäumen lässt sich wie die Funktion von Storch und Wegener durch genetische Algorithmen mit konstanter Populationsgröße effizient optimieren, so dass ein einigermaßen fairer Vergleich von Mutation und Rekombination gegeben ist.

Einzuwenden ist allerdings, dass dem (2+2) GA mit einer Populationsgröße von 2 und Fitness Sharing mächtigere Werkzeuge zugestanden werden als dem (1+1) EA. Eine interessante offene Frage wäre daher, wie sich ein mutationsbasierter Algorithmus mit einer Populationsgröße von mindestens 2 und Fitness Sharing auf der Ising-Funktion auf vollständigen binären Bäumen schlägt.

Literaturverzeichnis

[1] P. Briest et al. (2004). *The Ising model: simple evolutionary algorithms as adaptation schemes.* In *Proceedings of the Eighth Conference on Parallel Problem Solving from Nature (PPSN VIII)*, Band 3242 von *Lecture Notes in Computer Science*, S. 31–40. Springer-Verlag.

[2] M. Dietzfelbinger, B. Naudts, C. van Hoyweghen und I. Wegener (2003). *The analysis of a recombinative hill-climber on H-IFF.* In *IEEE Transactions on Evolutionary Computation*, 7: 417–423.

[3] S. Droste, T. Jansen und I. Wegener (2002). *On the analysis of the (1+1) evolutionary algorithm.* In *Theoretical Computer Science*, 276: 51–81.

[4] W. Feller (1971). *An Introduction to Probability Theory and Its Applications*, Band 2. Wiley, New York.

[5] S. Fischer (2003). *Evolutionäre Algorithmen für verallgemeinerte Ising-Modelle.* Diplomarbeit, Universität Dortmund.

[6] S. Fischer (2004). *A polynomial upper bound for a mutation-based algorithm on the two-dimensional Ising model.* In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) 2004*, Band 3102 von *Lecture Notes in Computer Science*, S. 1100–1112. Springer-Verlag.

[7] S. Fischer und I. Wegener (2004). *The Ising model on the ring: Mutation versus recombination.* In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) 2004*, Band 3102 von *Lecture Notes in Computer Science*, S. 1113–1124. Springer-Verlag.

[8] D. E. Goldberg, C. Van Hoyweghen und B. Naudts (2001). *Building block supeority, multimodality and synchronization problems.* In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) 2001*, S. 694–701. Morgan Kaufmann.

[9] D. E. Goldberg, C. Van Hoyweghen und B. Naudts (2002). *From twomax to the Ising model: Easy and hard symmetrical problems.* In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) 2002*, S. 626–633. Morgan Kaufmann.

[10] D. E. Goldberg, C. Van Hoyweghen und B. Naudts (2002). *Spin-flip symmetry and synchronization*. In *Evolutionary Computation*, 10: 317–344.

[11] J. H. Holland (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press.

[12] E. Ising (1925). *Beitrag zur Theorie des Ferromagnetismus*. In *Z. Physik*, 31: 235–288.

[13] T. Jansen und I. Wegener (2001). *Real royal road functions – functions where crossover provably is essential*. In L. Spector, E. D. Goodman, A. Wu et al. (Hg.), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001)*, S. 375–382. Morgan Kaufmann.

[14] S. Kobe (2004). *Das Ising-Modell – gestern und heute*. WWW Seite. `http://www.physik.tu-dresden.de/itp/members/kobe/isingphbl/`.

[15] M. Mitchell, S. Forrest und J. H. Holland (1992). *The royal road for genetic algorithms: Fitness landscapes and GA performance*. In F. J. Varela und P. Bourgine (Hg.), *Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life, 1991*, S. 245–254. A Bradford book, The MIT Press, Paris.

[16] M. Mitchell, J. H. Holland und S. Forrest (1994). *When will a genetic algorithm outperform hill climbing*. In J. D. Cowan, G. Tesauro und J. Alspector (Hg.), *Advances in Neural Information Processing Systems*, Band 6, S. 51–58. Morgan Kaufmann Publishers, Inc.

[17] B. Naudts und J. Naudts (1998). *The effect of spin-flip symmetry on the performance of the simple GA*. In *Proceedings of the Fifth Conference on Parallel Problem Solving from Nature (PPSN V)*, Band 1488 von *Lecture Notes in Computer Science*, S. 317–344. Morgan Kaufmann.

[18] R. Motwani und P. Raghavan (1995). *Randomized Algorithms*. Cambridge University Press, 1. Auflage.

[19] T. Storch und I. Wegener (2004). *Real royal road functions for constant population size*. In *Theoretical Computer Science*, 320: 123–134.

[20] I. Wegener (2002). *Methods for the analysis of evolutionary algorithms on pseudo–boolean functions*. In R. Sarker, M. Mohammadian und X. Yao (Hg.), *Evolutionary Optimization*, Kapitel 14, S. 349–369. Kluwer Academic Publishers.

[21] I. Wegener (SS 2002). *Evolutionäre Algorithmen*. Skript zur gleichnamigen Vorlesung.

[22] I. Wegener und C. Witt (2004). *On the analysis of a simple evolutionary algorithm on quadratic pseudo-boolean functions*. Erscheint in Journal of Discrete Algorithms.

Index