Inhaltsverzeichnis

1 Einleitung

1.1 Die Natur, unser Motivator

In der Natur herrscht ein ständiges Fressen und Gefressen Werden. Verschiedene Arten von Lebewesen einerseits und die Individuen innerhalb einer Art andererseits stehen in permanentem Wettbewerb untereinander. Man kann das Prinzip der Natur als 'Variation und Selektion' beschreiben: Je besser ein Individuum an seine Umwelt angepasst ist, desto besser sind seine Überlebenschancen und desto eher ist es demnach in der Lage, sich fortzupflanzen. Dies ist die Selektion (Die besten Individuen überleben, 'Survival of the Fittest'). Pflanzt sich ein Individuum fort, so entsteht dabei i.d.R. ein Individuum, das seinem bzw. seinen Vorfahren relativ ähnlich ist, das aber dennoch gewisse Unterschiede (Variationen) zu den Vorgängern aufweist. Auch diese Nachkommen sind dem natürlichen Wettbewerb ausgesetzt, und es haben diejenigen Nachkommen in der freien Wildbahn die besten Chancen, bei denen die Variationen in eine gute Richtung liefen - diese Individuen pflanzen sich dann eher fort und das Spiel beginnt wieder von vorne. Als Beispiel kann man den Wettkampf Löwen gegen Gazellen angeben. Die Überlebenschancen von Gazellen werden besser, je schneller sie laufen können und je schärfer ihre Sinne zur Früherkennung von Löwen sind. Die Überlebenschancen der Löwen steigen auch, wenn sie schnell laufen können, aber auch, wenn sie mehr Jagdgeschick erlangen (z.B. sich unbemerkt anschleichen können). Die verschiedenen Arten von Lebewesen haben also die Möglichkeit, sich nach mehreren Kriterien zu optimieren. Den Wettbewerb der Gazellen untereinander (Nur die schnellsten und aufmerksamsten Tiere oder gute 'Kombinationskämpfer' überleben) kann man als lokale Optimierung im genetischen Suchraum verstehen, ebenso den Wettbewerb der individuellen Löwen unter sich, da sich die genetischen Baupläne der verschiedenen Gazellen bzw. der verschiedenen Löwen sehr stark ähnlich sind. Den Wettkampf Löwen gegen Gazellen kann man als globale Optimierung sehen, dort wird nach dem 'besten' Lebewesen überhaupt gesucht. Werden die Gazellen so schnell und scharfsinnig, dass die Löwen nichts mehr zu fressen kriegen und sind die Löwen nicht fähig, ein neues Beutetier zu finden, so ist offenbar in der genetischen Umgebung des Gazellengenoms ein besser auf die Umwelt abgestimmtes Lebewesen machbar als mit dem genetischen Material der Löwen, und folglich sterben die Löwen aus und die Gazellen pflanzen sich unbehelligt weiter fort.

1.2 EA - Evolutionäre Algorithmen

Evolutionäre Algorithmen (EA) nehmen das Optimierungsverhalten der Natur auf, um aus einer Vielzahl von Individuen (Möglichen Lösungen eines Problems) die Geeignetsten bzw. das Geeignetste auszuwählen. Bei EA spielt der Zufall eine Rolle in der Art, dass ein Individuum nach dem Zufallsprinzip leicht verändert

wird (Variation) und danach kontrolliert wird, ob die Veränderung vorteilhaft war und das neue Individuum allenfalls in die lebende ('fortpflanzungsfähige') Population aufgenommen werden soll und ob vielleicht auch alte und schlechte Individuen 'sterben' sollen (Selektion). EA suchen innerhalb des sogenannten Suchraumes nach Individuen, die bezüglich eines oder mehrerer Bewertungskriterien möglichst gut abschneiden, bzw. die im sog. Zielraum möglichst gut positioniert sind. EA sind populationsbasierte Algorithmen, d.h. sie halten eine Menge von einem oder mehreren Individuen im Speicher, die sie variieren und gegebenenfalls durch andere, i.d.R. 'bessere' Individuen ersetzen können, welche bei der Variation entstehen. Die Individuen in der gespeicherten Population können dabei auf sehr viele verschiedene Arten variiert werden - z.B. können die 'Gene' verschiedener Individuen zu einem neuen Individuum verschmolzen werden, oder man kann von einem Individuum eine nur leicht veränderte Kopie erstellen, oder man kann für jedes 'Gen' eine individuelle Mutationswahrscheinlichkeit festlegen, ebenso wie man verschiedene Variationstechniken kombinieren kann.

Auch bei der Selektion sind diverse Techniken denkbar, z.B. kann man die Individuen im gespeicherten Archiv bzw. in der Population immer 'sterben lassen' (löschen) und nur die besten Kandidaten aus den variierten Nachkommen neu ins Archiv aufnehmen oder man kann jedes Individuum für eine festgelegte Zahl Variationschritte oder theoretisch auch für immer überleben lassen und es der Konkurrenz mit den 'Kindern' aussetzen usw. Um EA grob zu klassifizieren hat man eine Notation entwickelt. Man kennzeichnet EA als $(\mu + \lambda)$-EA oder als (μ, λ)-EA. μ bezeichnet dabei die Archivgrösse und λ die Anzahl neuer Individuen, die beim Variationsschritt entstehen. Beim $(\mu + \lambda)$-EA wird bei der Selektion aus allen $\mu + \lambda$ Individuen eine Auswahl von μ Individuen getroffen, die Eltern überleben also sozusagen die Geburt ihrer Kinder, beim (μ, λ)-EA wählt man die μ Individuen fürs Archiv nur aus den λ Exemplaren aus, die aus der Variation ihrer μ Vorgänger entstanden sind. Mit diesen Bezeichnungen ist allerdings noch nichts darüber gesagt, wie man die Variation und Selektion vornimmt und auch nichts darüber, nach wie vielen Kriterien man die Individuen beurteilt. Es ist beispielsweise denkbar (und gar nicht immer sinnlos), dass man mit einer gewissen Wahrscheinlichkeit auch Verschlechterungen in Kauf nimmt, denn allenfalls muss man, um das Individuum zu finden, das im Zielraum das globale Maximum erreicht, von einem lokalen Maximum wegkommen, durch ein 'Tal' im Zielraum schreiten, um dann eben auf ein besseres oder gar globales Maximum zu steigen.

EA kommen v.a. dort zum Einsatz, wo keine analytische Funktion für die Abbildung des Suchraumes in den Zielraum zur Verfügung steht, d.h. dort, wo keine analytischen Verfahren zur Suche nach lokalen und globalen Maxima vorhanden sind wie z.B. Funktionsableitungen oder Gradienten. Eine umfassendere Einführung zum Thema EA ist gegebeben in [1] oder [2].

Der Suchraum aller in dieser Arbeit behandelten Probleme, den wir X nen-

nen, ist immer die Menge aller N-dimensionalen Entscheidungsvektoren $\overline{x} = (x_1, x_2, \ldots, x_N)$, wobei alle Entscheidungsvariablen x_i nur die beiden Zustände 0 oder 1 einnehmen können: $X = \mathbb{B}^N$ mit $\mathbb{B} = \{0, 1\}$. X besteht also aus 2^N Vektoren der Länge N. N bezeichnet man dabei als Problemgrösse. X wird über eine Funktion $f : X \to F$ in den Zielraum F abgebildet. F kann sowohl eine Menge von reellen Zahlen als auch eine Menge reellwertiger Vektoren sein, je nachdem, ob wir nur eine oder mehrere Zielfunktionen haben. Es ist also $F \subseteq \mathbb{R}^m$ mit $m \in \mathbb{N}$, wobei m die Anzahl der Zielfunktionen ist. Weiter gilt $|F| \leq |X|$, da f eine eindeutige aber i.A. keine eineindeutige Zuordnung der Elemente aus dem Suchraum X zu Elementen des Zielraumes F ist.

1.3 Optimierprobleme mit einer Zielfunktion

Ist $m = 1$, so haben wir den Fall einer eindimensionalen Zielfunktion. f bildet also alle Elemente von X auf eine reelle Zahl ab: $f : X \to F \subseteq \mathbb{R}$. Ist man auf der Suche nach dem optimalen Vektor $\overline{x}^*$ aus X, so heisst dies, dass man herausfinden will, für welches $\overline{x}$ die Zielfunktion $f(\overline{x})$ ihr globales Maximum f_{max} erreicht. (Will man $f(\overline{x})$ minimieren, so kann man stattdessen $-f(\overline{x})$ maximieren.) Eine solche Optimierung kann wie gesagt z.B. mit einem EA vollzogen werden.

Bei den in dieser Arbeit behandelten einkriteriellen Optimierproblemen kommt ein (1+1)-EA zum Einsatz, der beim Variationsschritt so vorgeht, dass er gleichwahrscheinlich genau eines von den N Bits von $\overline{x}$ im Archiv der Grösse 1 invertiert (d.h. aus 0 wird 1 und umgekehrt an der betreffenden Stelle im Bitstring). Den so neu entstehenden Bitstring nennen wir im Folgenden $\overline{x}'$. Ist sein Zielfunktionswert $f(\overline{x}')$ grösser oder zumindest gleich gross wie $f(\overline{x})$, so ersetzen wir den aktuellen Bitstring bzw. Entscheidungsvektor $\overline{x}$ im Archiv durch den Vektor $\overline{x}'$. Initialisiert wird das Archiv zu Laufzeitbeginn durch einen Zufallsvektor $\overline{x}_0$, bei dem jeweils alle Elemente mit gleicher Wahrscheinlichkeit und voneinander unabhängig auf 0 oder 1 gesetzt sind. In Pseudocode sieht unser $(1 + 1)$-EA folgendermassen aus:

Algorithm 1 $(1 + 1)$-EA

```
1: Wähle gleichwahrscheinlich ein beliebiges x̄_0 aus X aus
2: x̄ := x̄_0
3: loop
4:    Generiere Variation x̄' durch Invertieren eines beliebigen Bits von x̄
5:    if f(x̄') ≥ f(x̄) then
6:       x̄ := x̄'
7:    end if
8: end loop
```

Der oben beschriebene (1+1)-EA kann im Variationsschritt jeweils nur Vek-

toren $\overline{x}'$ erzeugen, die zu den erzeugenden Vektoren $\overline{x}$ eine Hammingdistanz von genau 1 aufweisen. (d.h. genau ein Bit x_i, $i \in \{1, 2, \ldots, N\}$ von $\overline{x}$ ist verschieden von x'_i von $\overline{x}'$.) Deshalb kann unser (1+1)-EA theoretisch auf einem lokalen Maximum im Zielraum stehen bleiben, dann nämlich, wenn alle Hammingnachbarn von $\overline{x}$ mit Distanz 1 einen kleineren Funktionswert liefern als $f(\overline{x})$. Der $(1+1)$-EA ist daher nicht uneingeschränkt brauchbar um globale Maxima zu finden, eignet sich wohl aber zum Auffinden lokaler Maxima in F bzw. der zugehörigen Entscheidungsvektoren aus X. Bei den in dieser Arbeit betrachteten einkriteriellen Problemen (Count Ones, Leading Ones und Binary Value) findet der $(1+1)$-EA die globalen Maxima unabhängig vom Startvektor $\overline{x}_0$. Alle globalen Maxima der Zielfunktion dieser drei Probleme werden mit dem Vektor $\overline{x}^* = (1, 1, 1, \ldots, 1, 1)$ erreicht. Ein Algorithmus, der für alle erdenklichen Zielfunktionen f das globale Maximum finden kann, ist der $(1+1)$-EA$_p$. Dieser geht beim Variationsschritt so vor, dass er für alle N Bits in $\overline{x}$ voneinander unabhängig mit Wahrscheinlichkeit p eine Invertierung vornimmt. Er kann also in einem Schritt ein $\overline{x}'$ erzeugen, dass jede Hammingdistanz von 0 bis N zu $\overline{x}$ haben kann. In Pseudocode präsentiert sich der $(1+1)$-EA$_p$ wie folgt:

Algorithm 2 $(1+1)$-EA$_p$

1: Wähle gleichwahrscheinlich ein beliebiges $\overline{x}_0$ aus X aus
2: $\overline{x} := \overline{x}_0$
3: **loop**
4: Generiere Variation $\overline{x}'$ durch Invertieren jedes Bits von $\overline{x}$ mit Wahrscheinlichkeit p, mit $0 < p < 1$
5: **if** $f(\overline{x}') \geq f(\overline{x})$ **then**
6: $\overline{x} := \overline{x}'$
7: **end if**
8: **end loop**

(1+1)-EA und (1+1)-EA$_p$ haben kein Abbruchkriterium. Wenn wir uns für die Laufzeiten dieser Algorithmen interessieren, so heisst das, dass wir wissen wollen, wieviele Schlaufendurchläufe wir zu erwarten haben, bis ein Entscheidungsvektor $\overline{x}^*$ ins Archiv wandert, für den die Zielfunktion $f(\overline{x})$ ihr globales Maximum $f_{max} = f(\overline{x}^*)$ erreicht. Es kann i.A. mehrere mögliche $\overline{x}^*$ geben. Bei den drei in dieser Arbeit behandelten einkriteriellen Optimierproblemen allerdings existiert wie schon gesagt jeweils nur ein $\overline{x}^*$, nämlich $\overline{x}^* = (1, 1, 1, \ldots, 1, 1)$.

1.4 Optimierprobleme mit mehreren Zielfunktionen

Ist $m > 1$, so haben wir ein mehrkriterielles Optimierproblem, $f(\overline{x})$ ist vektorwertig, also $f(\overline{x}) = (f_1(\overline{x}), f_2(\overline{x}), \ldots, f_m(\overline{x}))$. I.d.R. existiert kein wunderbarer Entscheidungsvektor, für den sämtliche $f_i(\overline{x})$ mit $i \in \{1, 2, \ldots, m\}$ ihr globales Maximum erreichen. Man kann auch nicht mehr so leicht sagen, mit $a, b \in \{1 \ldots 2^N\}$, $a \neq b$ und $\alpha \neq \beta$, ein gewisses $\overline{x}_a$ sei besser als ein $\overline{x}_b$, denn

es ist gut möglich, dass $f_\alpha(\overline{x}_a) > f_\alpha(\overline{x}_b)$, aber $f_\beta(\overline{x}_a) < f_\beta(\overline{x}_b)$, d.h. jeder Entscheidungsvektor kann also seine 'Stärken und Schwächen' haben. Um trotzdem eine Ordnung in die Menge X der Entscheidungsvektoren hineinzubringen, führen wir den Begriff der Dominanz ein. Ein Vektor $\overline{x}_1$ dominiert einen anderen Vektor $\overline{x}_2$, falls $\overline{x}_1$ bezüglich aller Zielfunktionen $f_1(\overline{x})$ bis $f_m(\overline{x})$ mindestens gleich gut abschneidet wie $\overline{x}_2$, in mindestens einer der m Zielfunktionen aber besser ist als $\overline{x}_2$. Siehe dazu auch Abbildung 1. Formal:

Definition 1.1 (Dominanz) *Sei $\overline{x}_1, \overline{x}_2 \in X = \mathbb{B}^N$ und $f : X \to F \subseteq \mathbb{R}^m$ mit $f(\overline{x}) = (f_1(\overline{x}), f_2(\overline{x}), \ldots, f_m(\overline{x}))$. $\overline{x}_1$ dominiert $\overline{x}_2$ bezüglich der Funktion f, geschrieben $\overline{x}_1 \succ \overline{x}_2$, wenn gilt:*

1. $\forall i \in \{1, 2, \ldots, m\} : f_i(\overline{x}_1) \geq f_i(\overline{x}_2)$
2. $\exists j \in \{1, 2, \ldots, m\} : f_j(\overline{x}_1) > f_j(\overline{x}_2)$

Wenn gilt $\overline{x}_1 \succ \overline{x}_2$, so kann man auch davon sprechen, dass der Vektor $f(\overline{x}_1)$ den Vektor $f(\overline{x}_2)$ dominiert ($f(\overline{x}_1) \succ f(\overline{x}_2)$). Wir können nun sagen, dass ein ein Vektor, der von einem andern Vektor aus dem Suchraum dominiert wird, keine Daseinsberechtigung hat, denn er lässt sich durch mindestens einen Vektor aus X ersetzen, der bezüglich aller Bewertungskriterien mindestens gleich gut und in mindestens einem Kriterium sogar besser ist. Es gibt in X mindestens einen, i.d.R. aber mehrere Vektoren, zu denen innerhalb des Suchraumes X kein dominanter Vektor existiert. Die Menge aller solcher nicht dominierter Vektoren nennen wir Paretomenge. Formal:

Definition 1.2 (paretooptimal, Paretomenge, Paretofront) *Ein $\overline{x}^* \in X$ nennen wir paretooptimal bezüglich $f : X \to F \subseteq \mathbb{R}^m$, falls kein anderer Vektor $\overline{x} \in X$ existiert, welcher $\overline{x}^*$ dominiert. Die Menge X^* aller paretooptimaler Entscheidungsvektoren nennen wir Paretomenge:*

$$X^* := \{\overline{x}^* \in X | \nexists\, \overline{x} \in X : \overline{x} \succ \overline{x}^*\}$$

Die Menge $F^* = f(X^*)$, also die Menge der Zielvektoren aller paretooptimaler Suchvektoren nennt man Paretofront.

Für jede Kombination von Suchraum X und eindeutiger Abbildung $f : X \to F \subseteq \mathbb{R}^m$ ist die Paretofront F^* und somit auch X^* eindeutig festgelegt. Es gibt also im konkreten Problemfall nur immer ein mögliches F^*. Eine mögliche Lage einer Paretofront im Zielraum ist durch die Abbildung 2 dargestellt.

Wir können den $(1 + 1)$-EA nun leicht auf Optimierprobleme mit mehreren Zielfunktionen anpassen. $\overline{x}$ muss im mehrkriteriellen Fall nicht mehr mit $\overline{x}'$ ersetzt werden, wenn $f(\overline{x}') \geq f(\overline{x})$, sondern dann, wenn $\overline{x}'$ nicht von $\overline{x}$ dominiert

Algorithm 3 $(1+1)$-MOEA

```
1: Wähle gleichwahrscheinlich ein beliebiges x̄_0 aus X aus
2: x̄ := x̄_0
3: loop
4:    Generiere Variation x̄' durch Invertieren eines beliebigen Bits von x̄
5:    if f(x̄) ⊁ f(x̄') then
6:       x̄ := x̄'
7:    end if
8: end loop
```

wird. Den so entstehenden EA nennen wir $(1+1)$-MOEA (Multi Objective Evolutionary Algorithm). Der zugehörige Pseudocode is in Algorithm 3 angegeben.

Ein Problem mit mehreren Zielfunktionen soll für uns genau dann als optimiert gelten, wenn wir eine Teilmenge der Paretomenge X^* gefunden haben, die bei der Abbildung in den Zielraum F die gesamte Paretofront F^* liefert, d.h. wir müssen zu jedem Element von F^* mindestens ein zugehöriges $\overline{x}^* \in X$ finden. Jedes $\overline{x} \in X \setminus X^*$ wird von mindestens einem $\overline{x}^*$ aus dieser Teilmenge dominiert, und zu keinem Element dieser Teilmenge existiert in X ein dominanter Vektor. Demnach is es i.A. nicht notwendig, die gesamte Paretomenge X^* zu finden, um ein mehrkriterielles Problem zu optimieren, da die Abbildung $f : X \to F$ nicht eineindeutig ist, jedes Element der Paretofront F^* kann durch die jeweilige Abbildung verschiedener $\overline{x}^* \in X^*$ entstehen. (Genausowenig forderten wir auch im einkriteriellen Fall, dass alle $\overline{x}^*$ mit $f(\overline{x}^*) = f_{max}$ gefunden werden müssen, damit das Problem optimiert sei.) Will man ein mehrkriterielles Problem mit einem EA lösen, so genügt ein Archiv mit nur einem Speicherplatz wie bei den zuvor beschriebenen EA i.d.R. nicht mehr (ausser es existiere ein Entscheidungsvektor, der alle m Zielfunktionen gleichzeitig maximiert). Die Archivgrösse muss mindestens $|F^*|$ betragen, denn jedes $\overline{x}^*$ wird durch f eindeutig einem $f(\overline{x}^*) \in F^*$ zugewiesen, somit brauchen wir also eine Population $P \subseteq X^* \subseteq X$, die mindestens $|F^*|$ Elemente enthält, um bei der Abbildung in den Zielraum ganz F^* überdecken zu können. Am flexibelsten sind wir, wenn wir die Archivgrösse variabel belassen.

Ein einfacher EA mit variabler Archivgrösse, der zur Mehrzieloptimierung verwendet werden kann, wurde von Laumanns, Zitzler und Thiele in [3] unter der Bezeichnung SEMO (Simple Evolutionary Multiobjective Optimizer) beschrieben. Der SEMO speichert in seinem Archiv P alle bereits evaluierten $\overline{x}$, die nicht durch einen anderen bereits evaluierten Suchvektor dominiert werden. Beim Variationsschritt geht der SEMO jeweils so vor, dass er mit gleicher Wahrscheinlichkeit ein $\overline{x}$ aus P auswählt und daraus eine Variante $\overline{x}'$ von $\overline{x}$ erzeugt durch Invertierung von genau einem zufälligen, gleichwahrscheinlichen Bit. Bei der Selektion wird dieses $\overline{x}'$ dann in P aufgenommen, falls in P kein Element $\overline{z}$ vorhanden ist, welches $\overline{x}'$ dominiert *und* wenn kein $f(\overline{z})$, $\overline{z} \in P$ mit $f(\overline{x}')$ identisch ist. Werden weiter Elemente aus P durch $\overline{x}'$ dominiert, so werden diese

aus P ausgeschlossen. Der SEMO ist so in der Lage, eine Population $P \subseteq X^*$ zu finden mit $|P| = |F^*|$, die durch f eineindeutig auf F^* abgebildet wird. In Pseudocode präsentiert sich der SEMO folgendermassen:

Algorithm 4 SEMO

1: Wähle gleichwahrscheinlich ein beliebiges $\overline{x}_0$ aus X aus
2: $\overline{x} := \overline{x}_0$
3: $P \leftarrow \{\overline{x}\}$
4: **loop**
5: Wähle gleichwahrscheinlich ein $\overline{x} \in P$ aus
6: Generiere Variation $\overline{x}'$ durch Invertieren eines beliebigen Bits von $\overline{x}$
7: **if** $\nexists\, \overline{z} \in P$ sodass $(\overline{z} \succ \overline{x}' \vee f(\overline{z}) = f(\overline{x}'))$ **then**
8: $P \leftarrow (P \setminus \{\overline{z} \in P | \overline{x}' \succ \overline{z}\}) \cup \{\overline{x}'\}$
9: **end if**
10: **end loop**

Den SEMO kann man als simpelstmöglichen Mehrzieloptimierer sehen, sozusagen die Erweiterung des $(1+1)$-EA auf einen $(\mu+1)$-EA mit variablem μ und adaptiert auf mehrere Zielfunktionen.

1.5 Approximationen

Im einkriteriellen Fall kann man sich auch mit einem Entscheidungsvektor $\overline{x}_\varepsilon$ zufriedengeben, für den $f(\overline{x}_\varepsilon)$ zwar hinter f_{max}, dem globalen Maximum von $f(\overline{x})$, zurückbleibt, der aber zumindest in einem ausreichenden Verhältnis zu f_{max} steht. Man erhofft sich im Gegenzug zur Tolerierung eines suboptimalen Resultats eine deutliche Einsparung an Rechenzeit. Dies ist durchaus berechtigt, denn EA finden in vielen Fällen schnell eine ganz passable Lösung, brauchen aber u.U. sehr lange, um die bestmögliche Lösung zu finden. Als Approximationsmass bzw. Approximationsgüte verwenden wir die Variable ε. Es soll dabei gelten $0 < \varepsilon \leq 1$. Konkret soll $\overline{x}_\varepsilon$ den Vektor $\overline{x}^*$ mit $f(\overline{x}^*) = f_{max}$ ausreichend approximieren, falls $f(\overline{x}_\varepsilon) > \frac{1}{1+\varepsilon} f_{max}$ ist. Durch die Verwendung des Operators $>$ anstelle des oft verwendeten $\geq$ ersparen wir uns in dieser Arbeit die Behandlung von Sonderfällen, dann nämlich, wenn $\frac{f_{max}}{1+\varepsilon}$ eine ganze Zahl ist. Bei den von uns in Kapitel 2 behandelten drei Problemen mit einer Zielfunktion (Count Ones, Leading Ones und Binary Value) spielt es bei der Bestimmung der Problemordnung in Abhängigkeit von N und ε keine Rolle, ob wir $>$ oder $\geq$ verwenden. (Wie wir in Kapitel 2 sehen werden, lassen sich die betreffenden Problemlaufzeiten als Summen erwarteter Anzahlen von Schleifendurchläufen zum Erreichen einzelner kleiner Fortschritte bezüglich $f(\overline{x})$ herleiten. Die erwartete Rechenzeit für diese kleinen Fortschritte ist in allen drei Fällen von geringerer Ordnung als die Problemordnung des jeweils übergeordneten Problems, beim Fall, wo $\frac{f_{max}}{1+\varepsilon}$ ganzzahlig ist, würden wir bei der Verwendung von $\geq$ genau einen solchen Aufwand von geringer Ordnung sparen, was die Problemordnung

des gesamten Problems nicht mindert.)

Bei den einkriteriellen Problemen haben wir uns nun also so festgelegt, dass das Problem dann ε-approximativ gelöst ist, sobald ein $\overline{x}_\varepsilon$ ins Archiv wandert, für das gilt $f(\overline{x}_\varepsilon) > \frac{f_{max}}{1+\varepsilon}$. Um in mehrkriteriellen Problemfällen entsprechend eine approximative Lösungsmenge X_ε festlegen zu können, führen wir das Konzept der ε-Dominanz ein (siehe dazu auch Abbildung 1):

Definition 1.3 (ε-Dominanz) *Ein Vektor $\overline{x}_1 \in X$ ε-dominiert einen Vektor $\overline{x}_2 \in X$ bezüglich der Abbildung $f : X \to F \subseteq \mathbb{R}^m$, notiert als $\overline{x}_1 \succ_\varepsilon \overline{x}_2$, wenn $\forall i \in \{1, 2, \ldots, m\}$ gilt:*

$$(1 + \varepsilon) f_i(\overline{x}_1) \geq f_i(\overline{x}_2)$$

mit $\varepsilon > 0$.

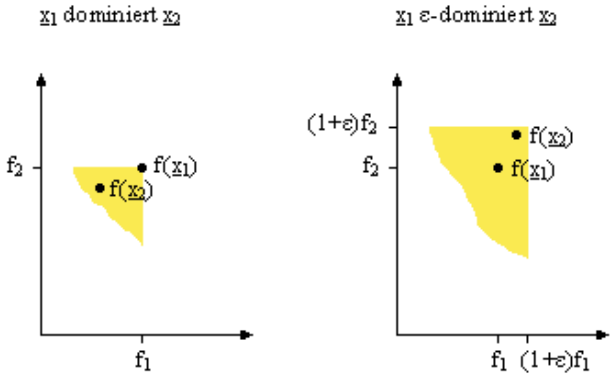

Abbildung 1: Dominanz und ε-Domnanz

Ebenso wie bei der nichtapproximativen Definition der Dominanz lässt sich auch das Konzept der ε-Dominanz in den Zielraum übertragen. Falls $\overline{x}_1 \succ_\varepsilon \overline{x}_2$, so kann man auch schreiben $f(\overline{x}_1) \succ_\varepsilon f(\overline{x}_2)$. Das Konzept der ε-Dominanz wurde schon in [4] vorgeschlagen. Weiter wurde in [4] eine ε-Approximation der Paretofront vorgeschlagen (Siehe auch Abbildung 2):

Definition 1.4 (ε-approximative Paretofront) *Sei $F \subseteq \mathbb{R}^m$ eine Menge von Vektoren. F_ε ist dann eine ε-approximative Paretofront, wenn jedes $\overline{g} \in F$ von mindestens einem $\overline{f} \in F_\varepsilon$ ε-dominiert wird:*

$$\forall \overline{g} \in F : \exists\, \overline{f} \in F_\varepsilon \quad mit \quad \overline{f} \succ_\varepsilon \overline{g}$$

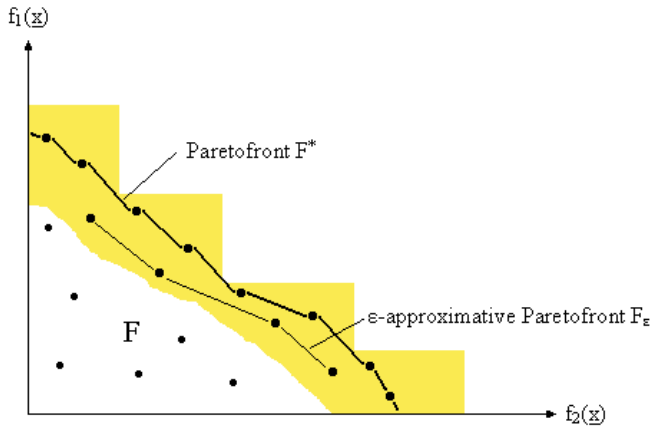

Abbildung 2: Paretofront F^* und ε-approximative Paretofront F_ε

Das 'Salz in der Suppe' bei einer ε-approximativen Paretofront sind natürlich diejenigen Vektoren, die einen Vektor der Paretofront F^* ε-dominieren. Ein Element, das einen Vektor $\overline{f}^*$ aus F^* ε-dominiert, ε-dominiert auch alle Elemente, welche von $\overline{f}^*$ dominiert werden. Nur durch Zugabe eines solchen Elementes kann eine Menge von Vektoren, die keine ε-approximative Paretofront ist zu einer solchen werden. Trotzdem machen wir es für die Elemente einer ε-approximativen Paretofront nicht zur Bedingung, dass sie ein Element von F^* ε-dominieren. Elemente einer ε-approximativen Paretofront, die kein Element von F^* ε-dominieren (wobei sie auch sich selber ε-dominieren können), könnten aus einer ε-approximativen Paretofront ersatzlos gestrichen werden, ebenso wie Elemente, die durch ein anderes Element der ε-approximativen Paretofront dominiert werden. Im Gegensatz zur nichtapproximativen Paretofront F^* kann je nach ε und F auch mehr als ein F_ε existieren. Mindestens ein F_ε existiert immer, nämlich F^* selber.

Übertragen in den Suchraum kann man auch ε-approximative Paretomengen X_ε definieren, Mengen, die bei der Abbildung f zumindest auf ein mögliches F_ε abgebildet werden. Für uns soll von hier an gelten, dass eine ε-approximative Lösung für ein mehrkriterilelles Problem dann vorliegt, wenn wir eine Population X_ε gefunden haben, die bei der Abbildung in den Zielraum mindestens eine der möglichen Mengen F_ε vollständig enthält. D.h. konkret, dass wir eine Menge $X_\varepsilon \subseteq X$ von Suchvektoren finden wollen, für die gelten soll, dass kein möglicher Suchvektor aus X nicht von mindestens einem Vektor aus X_ε ε-dominiert wird. Formal:

Definition 1.5 (ε-approximative Paretomenge) *Sei $X = \mathbb{B}^N$ die Menge aller N-dimensionaler binärer Entscheidungsvektoren. X_ε nennen wir dann eine ε-approximative Paretomenge, wenn jedes $\overline{z} \in X$ von mindestens einem $\overline{x} \in X_\varepsilon$ ε-dominiert wird:*

$$\forall \overline{z} \in X : \exists\, \overline{x} \in X_\varepsilon \quad mit \quad \overline{x} \succ_\varepsilon \overline{z}$$

Bildet man ein X_ε in den Zielraum ab, so entsteht dort eine ε-approximative Paretofront. Mindestens ein X_ε existiert immer, nämlich die Paretomenge X^* selber.

1.6 Zielsetzung

Es existieren Laufzeitanalysen von Optimierproblemen mit mehreren Zielfunktionen, z.B. in [3]. Daneben existieren auch Arbeiten, die sich mit der Laufzeit von Algorithmen bis zur approximativen Optimierung von Problemen befassen. So haben beispielsweise Wegener und Giel in [5] gezeigt, dass der $(1+1)$-EA$_p$ mit $p = \frac{1}{N}$ für das Maximum Matching Problem mit N Kanten in $O\left(N^{2\lceil \frac{1}{\varepsilon} \rceil}\right)$ Schleifendurchläufen eine ε-approximative Lösung findet. Ein Ziel dieser Arbeit soll es sein, aufbauend auf die in Kapitel 2 erarbeiteten erwarteten Laufzeiten für das Auffinden von approximativen Lösungen von Optimierproblemen mit einer Zielfunktion, für Probleme mit mehreren Zielfunktionen so etwas wie eine Approximation der 'First hitting time', des Zeitpunktes des Erreichens des ersten paretooptimalen Suchvektors, zu bestimmen. Dies geschieht in Kapitel 3. Weiter wird in Kapitel 4 für den relativ einfachen populationsbasierten SEMO Algorithmus die erwartete Laufzeit für das Auffinden einer ε-approximierten Paretomenge nach oben und nach unten abgegrenzt. Letzteres kann man als Kombination der Forschungsrichtungen aus [3] und [5] betrachten.

2 Optimierprobleme mit einer Zielfunktion

2.1 Count Ones Optimierproblem

Wir wollen als Einstieg eine relativ simple Optimieraufgabe behandeln, das sog. Count Ones Problem. Dies ist ein Problem mit einer einzigen Zielfunktion $f(\overline{x})$, wie im Titel dieses Kapitels erwähnt. Dabei wird der Zielfunktionswert $f(\overline{x})$ eines Bitstrings $\overline{x} = (x_1, x_2, \ldots, x_N)$ der Länge N als die Summe der auf 1 gesetzten Bits in diesem String berechnet:

Definition 2.1 *Die Count Ones Funktion ist wie folgt definiert:*

$$f(\overline{x}) = \sum_{k=1}^{k=N} x_k$$

mit $x_k \in \mathbb{B} = \{0, 1\}$ *und* $N \in \mathbb{N}$.

Zum besseren Verständnis sei in Abbildung 3 ein Beispiel für $N = 8$ angegeben.

1	0	0	0	**1**	**1**	0	0

$$f(\overline{x}) = \sum_{j=1}^{8} x_j = 3$$

Abbildung 3: Beispiel für Count Ones Problem mit $N = 8$

Wir wollen im Weiteren das Verhalten des in der Einleitung beschriebenen $(1 + 1)$-EA untersuchen.

Es soll als Erstes eine obere Schranke für den Erwartungswert der Anzahl Schritte zur Optimierung des Bitstrings hergeleitet werden. Unter Optimierung versteht sich hier die Anwendung des in der Einleitung beschriebenen $(1 + 1)$-EA solange, bis der Bitstring aus lauter auf 1 gesetzten Bits betsteht, bzw. bis die Funktion $f(\overline{x})$ ihr globales Maximum $f_{max} = N$ erreicht. Da wir eine obere Schranke herleiten wollen, gehen wir davon aus, dass bei der Initialisierung des Bitstrings ein $\overline{x}_0$ resultiert, das aus lauter Bits mit Wert 0 besteht. Dies ist der Worst Case Fall, einen Startvektor mit schlechterem $f(\overline{x}_0)$ existiert hier nicht. ($\overline{x}_0 = \overline{0}$, somit ist $f(\overline{x}_0) = 0$). Wir steigern uns also in genau N 1er Schritten von 0 bis und mit N bezüglich $f(\overline{x})$, wir ersetzen auf unserem Weg zum optimalen Bitstring $\overline{x}^* = (1, 1, 1, \ldots, 1, 1)$ ja jedes der N auf 0

gesetzten Bits Schritt für Schritt mit einer 1. Weiter wissen wir, dass die erwartete Anzahl Iterationsschritte für einen einzelnen Fortschritt (d.h. die erwartete Anzahl Variationen, um von $\overline{x}$ aus ein $\overline{x}'$ zu finden, für das im vorliegenden Falle gilt: $f(\overline{x}') = f(\overline{x}) + 1$) nicht von der Anzahl der benötigten Schritte für vorangehende oder nachfolgende Fortschritte in $f(\overline{x})$ abhängig ist, weil es sich bei unserem $(1+1)$-EA um einen gedächtnislosen Prozess handelt. Damit, und mit der Erkenntnis, dass wir uns ausschliesslich in 1er-Schritten bezüglich $f(\overline{x})$ steigern mit unserem $(1+1)$-EA (Wir können maximal eine 0 durch eine 1 ersetzen pro Variation, und pro geflippte 0 wächst $f(\overline{x})$ um 1), lässt sich sagen, dass wir Erwartungswerte der Schrittanzahl von einem beliebigen $\overline{x}$ aus mittels Summierung der Erwartungswerte für die Anzahl Iterationsschritte, welche für die einzelnen 1er-Schritte in $f(\overline{x})$ resultieren, ausrechnen können. Wir gehen wie gesagt von Worst Case aus, indem wir mit dem Nullvektor starten. Damit ergibt sich als Resultat eine obere Schranke für unseren Erwartungswert.

Nun müssen wir also der Frage nachgehen, wieviele Iterationsschritte wir durchschnittlich brauchen, um in $f(\overline{x})$ einen Schritt voranzukommen, falls die Problemgrösse N ist und falls $f(\overline{x}) = k$ ist, mit $0 \leq k < N$, was soviel bedeutet, dass bereits k Bits im Bitstring auf 1 gesetzt sind. Es lässt sich leicht die Überlegung anstellen, dass die Wahrscheinlichkeit, den Bitstring bei gegebenem N und k durch Invertierung eines zufälligen Bits derart zu variieren, dass $f(\overline{x}') = k+1 > f(\overline{x}) = k$ gleich $\frac{N-k}{N}$ ist, was dem Verhältnis der Anzahl zu 0 gesetzter Bits zu der gesammten Bitanzahl N entspricht. Folglich ist die Wahrscheinlichkeit, den Bitstring durch eine Variation zu verschlechtern $1 - \frac{N-k}{N} = \frac{k}{N}$. Mit $\alpha = \frac{N-k}{N}$ ergibt sich der Erwartungswert der Anzahl benötigter Variationen bis zum Auffinden einer dominanten (d.h. besseren) Lösung gegeben N, k, den wir $E_{step,CO}(N,k)$ nennen, folgendermassen:

$$E_{step,CO}(N,k) = \sum_{j=0}^{\infty} \alpha(1-\alpha)^j(j+1) = \frac{\alpha}{1-\alpha} \sum_{j=1}^{\infty} j(1-\alpha)^j \qquad (1)$$

$E_{step,CO}(N,k)$ berechnet sich somit nach obiger Formel als unendliche Summe. Dabei beschreibt der Term $\alpha(1-\alpha)^j$ die Wahrscheinlichkeit, dass eine dominante Lösung bei der $(j+1)$-ten Variation gefunden wird. Zur Berechnung der erwarteten Anzahl Iterationsschritte müssen diese einzelnen Wahrscheinlichkeiten mit den jeweils dazugehörigen Schrittanzahlen $(j+1)$ multipliziert und dann summiert werden. Setzen wir $\beta = 1 - \alpha$ und führen wir die Variable s ein, so können wir schreiben:

$$s = \sum_{j=1}^{\infty} j\beta^j = \beta + 2\beta^2 + 3\beta^3 + 4\beta^4 + \cdots \Rightarrow$$

$$\beta s = \beta^2 + 2\beta^3 + 3\beta^4 + \cdots = s - \sum_{j=1}^{\infty} \beta^j = s - \left(\frac{1}{1-\beta} - 1\right) \Rightarrow$$

$$(\beta - 1)s = 1 - \frac{1}{1-\beta} \Rightarrow$$

$$s = \frac{1}{\beta - 1} - \frac{1}{(1-\beta)(\beta-1)} \tag{2}$$

Setzen wir nun wieder $\beta = 1 - \alpha$ in (2) ein, so können wir unseren Erwartungswert wie folgt schreiben:

$$E_{step,CO}(N,k) = \left(\frac{1}{(1-\alpha)-1} - \frac{1}{(1-(1-\alpha))((1-\alpha)-1)}\right)\left(\frac{\alpha}{1-\alpha}\right)$$

$$= \left(\frac{1}{-\alpha} + \frac{1}{\alpha^2}\right)\left(\frac{\alpha}{1-\alpha}\right) = \frac{1}{\alpha - 1} + \frac{1}{(1-\alpha)\alpha} = \frac{-\alpha}{(1-\alpha)\alpha} + \frac{1}{(1-\alpha)\alpha}$$

$$= \frac{1-\alpha}{(1-\alpha)\alpha} = \frac{1}{\alpha} = \frac{N}{N-k} \tag{3}$$

Dieses Resultat ist gültig für $k \in \{0, \ldots, N-1\}$. Man nennt die in (1) angetroffene Wahrscheinlichkeitsverteilung geometrisch. Wir werden wiederholt von der Tatsache Gebrauch machen, dass man bei unabhängigen Experimenten, bei denen ein bestimmtes Ereignis mit der Wahrscheinlichkeit $p \leq 1$ eintritt im Schnitt $\frac{1}{p}$ Experimente abwarten muss bis zum erstmaligen Auftreten dieses Ereignisses. Wie oben erwähnt, erhalten wir eine obere Schranke für den Erwartungswert für die Rechenzeit $T_{CO}(N)$ des gesamten Problems mittels Summation der einzelnen Erwartungswerte $E_{step,CO}(N,k)$ für $k \in \{0, \ldots, N-1\}$:

$$T_{CO}(N) \leq \sum_{k=0}^{N-1} E_{step,CO}(N,k) = \sum_{k=0}^{N-1} \frac{N}{N-k} \tag{4}$$

Wir können die Summe in (4) leicht umschreiben, indem wir einfach die Summationsreihenfolge der zu summierenden Brüche vertauschen:

$$T_{CO}(N) \leq \sum_{k=0}^{N-1} \frac{N}{N-k} = \sum_{i=1}^{N} \frac{N}{i} = N \sum_{i=1}^{N} \frac{1}{i} \tag{5}$$

Ein Blick in eine geeignete Formelsammlung ergibt, dass sich der Term $\sum_{i=1}^{N} \frac{1}{i}$ in etwa wie $\log N$ verhält. Ganz genau gilt folgende Ungleichung:

$$\ln N < H_N < \ln N + 1 \tag{6}$$

Mit $H_N = \sum_{i=1}^{N} \frac{1}{i}$. Man nennt H_N die N-te harmonische Zahl. Aus Ungleichung (6) lässt sich im Weiteren ableiten mit $\tau \in \mathbb{R}$:

$$\begin{aligned} T_{CO}(N) &\leq N \sum_{i=1}^{N} \frac{1}{i} = NH_N < N \ln N + N \\ &= \tau N \log N + O(N) = O(N \log N) \end{aligned} \tag{7}$$

Somit können wir folgenden Satz aufstellen:

Satz 2.1 *Die erwartete Laufzeit des* $(1+1)$*-EA angewendet auf die Count Ones Optimieraufgabe mit Problemgrösse* N *ist* $O(N \log N)$.

Wollte man den exakten Erwartungswert für die Laufzeit des Count Ones Problems bestimmen, so müsste man alle möglichen (hier immer gleich wahrscheinlichen) $\overline{x}_0$ bzw. die jeweils daraus resultierenden zu erwartenden Laufzeiten bis zum Erreichen des globalen Maximums f_{max} bei $\overline{x}^* = (1, 1, 1, \ldots, 1, 1)$ berücksichtigen. Man kommt auf folgenden (komplizierten) Ausdruck:

$$T_{CO}(N) = \sum_{s=0}^{N-1} \left(\frac{1}{2^N} \binom{N}{s} \sum_{k=s}^{N-1} \frac{N}{N-k} \right) \tag{8}$$

Wobei $s = f(\overline{x}_0)$ ist und der Term $\frac{1}{2^N}\binom{N}{s}$ die Wahrscheinlichkeit beschreibt, dass $f(\overline{x}_0) = s$ ist. $\sum_{k=s}^{N-1} \frac{N}{N-k}$ ist die erwartete Laufzeit, falls $f(\overline{x}_0) = s$.

Diesen Term wollen wir aber nicht weiterbearbeiten, stattdessen wollen wir dafür eine untere Schranke bestimmen. Wir machen hierzu eine Fallunterscheidung:

Fall 1: N ungerade, $N \geq 3$

Da $f(\overline{x}_0)$ im Intervall $\{0, 1, \ldots, N\}$ binomial verteilt ist, gilt für genau die Hälfte der $\overline{x}_0$, dass $f(\overline{x}_0) \leq \frac{N-1}{2}$, wobei auch $\overline{x}_0$ existieren, für die gilt $f(\overline{x}_0) < \frac{N-1}{2}$. Für solche $\overline{x}_0$ ist die erwartete Zahl der Iterationsschritte bis zur Optimierung von $\overline{x}$ jeweils mindestens $\sum_{k=\frac{N-1}{2}}^{N-1} \frac{N}{N-k}$ und grösser als $\sum_{k=\frac{N-1}{2}}^{N-1} \frac{N}{N-k}$, falls $f(\overline{x}_0) < \frac{N-1}{2}$. Da diese Schranke die Hälfte der $\overline{x}_0$ betrifft und man bei einer nicht verschwindenden Anzahl $\overline{x}_0$ auch mehr benötigte Variationen zu erwarten hat wie im Falle, wenn $f(\overline{x}_0) = \frac{N-1}{2}$, lässt sich schreiben:

$$T_{CO}(N) > \frac{1}{2}\sum_{k=\frac{N-1}{2}}^{N-1}\frac{N}{N-k} = \frac{N}{2}\sum_{k=\frac{N-1}{2}}^{N-1}\frac{1}{N-k} = \frac{N}{2}\sum_{i=1}^{\frac{N+1}{2}}\frac{1}{i} \tag{9}$$

Durch die Verdoppelung der oberen Summationsgrenze in (9) kann man $\frac{N+1}{2}$ Summandenpaare bilden, nämlich mit $\frac{1}{j}$ und $\frac{1}{j+\frac{N+1}{2}}$ für $j \in \{1,\ldots,\frac{N+1}{2}\}$. Es ist leicht zu sehen, dass gilt $\frac{1}{j} \geq \frac{2}{3}\left(\frac{1}{j}+\frac{1}{j+\frac{N+1}{2}}\right)$, wobei auch das Zeichen '>' vorkommt, da $N \geq 3$. Deshalb wird aus dem $\frac{N}{2}$ vor dem Summenzeichen in (9) das nachfolgende $\frac{N}{3}$. Notfalls käme man aber auch ohne diese Raffinesse aus. Wir schreiben somit:

$$T_{CO}(N) > \frac{N}{3}\sum_{i=1}^{N+1}\frac{1}{i} = \frac{N}{3}\sum_{i=1}^{N}\frac{1}{i} + \underbrace{\frac{N}{3(N+1)}}_{<\frac{1}{3}} = \frac{N}{3}\sum_{i=1}^{N}\frac{1}{i} + O(1) \tag{10}$$

Unter Anwendung von (6) kann geschrieben werden mit $\tau \in \mathbb{R}$:

$$\begin{aligned} T_{CO}(N) &> \frac{N}{3}\sum_{i=1}^{N}\frac{1}{i} + O(1) = \frac{N}{3}H_N + O(1) \\ &> \frac{N}{3}\ln N + O(1) = \tau\frac{N}{3}\log N + O(1) \end{aligned} \tag{11}$$

Also ist $T_{CO}(N)$ von $\Omega(N\log N)$, falls N ungerade ist.

Fall 2: N gerade, $N \geq 2$

$f(\overline{x}_0)$ ist binomialverteilt. Für mehr als die Hälfte der möglichen $\overline{x}_0$ gilt: $f(\overline{x}_0) \leq \frac{N}{2}$, wobei $\overline{x}_0$ existieren, für die gilt $f(\overline{x}_0) < \frac{N}{2}$. Für alle diese $\overline{x}_0$ ist die zu erwartende Zahl Variationen im Minimum $\sum_{k=\frac{N}{2}}^{N-1}\frac{N}{N-k}$. Wir können also schreiben:

$$\begin{aligned} T_{CO}(N) &> \frac{1}{2}\sum_{k=\frac{N}{2}}^{N-1}\frac{N}{N-k} = \frac{N}{2}\sum_{k=\frac{N}{2}}^{N-1}\frac{1}{N-k} = \frac{N}{2}\sum_{i+1}^{\frac{N}{2}}\frac{1}{i} \\ &> \frac{N}{3}\sum_{i=1}^{N}\frac{1}{i} = \frac{N}{3}H_N > \frac{N}{3}\ln N \end{aligned} \tag{12}$$

Also ist $T_{CO}(N)$ von $\Omega(N \log N)$ sowohl für gerade wie auch für ungerade N. Die zuvor hergeleitete obere Schranke war von gleicher Ordnung wie die vorliegende untere Schranke. Daraus können wir folgern:

Satz 2.2 *Die erwartete Laufzeit des* $(1+1)$*-EA angewendet auf die Count Ones Optimieraufgabe der Problemgrösse* N *ist* $\Theta(N \log N)$.

Nun wollen wir das Count Ones Problem nur approximativ lösen, d.h. wir brechen die Iterationen ab, sobald wir irgendeine Lösung $\overline{x}_\varepsilon$ gefunden haben, die uns einen Wert für $f(\overline{x}_\varepsilon)$ liefert, der in einem ausreichenden Verhältnis zum globalen Maximum f_{max} von $f(\overline{x})$ für alle $\overline{x} \in X = \mathbb{B}^N$ steht. Die Approximationsgüte wird dabei durch ε bestimmt, wobei $\varepsilon > 0$. Wir begnügen uns konkret mit einer Lösung $\overline{x}_\varepsilon$, falls gilt $f(\overline{x}_\varepsilon) > \frac{1}{1+\varepsilon} f_{max}$. Um eine obere Schranke $E_{CO,\varepsilon}(N, \varepsilon)$ für den Erwartungswert zu bestimmen, muss man wiederum wie oben die $E_{step,CO}(N, k)$ summieren, allerdings nur für $k \in \{0 \ldots k_{\varepsilon 2}\}$ mit $k_{\varepsilon 2} = \lfloor \frac{N}{1+\varepsilon} \rfloor$. Zur Vereinfachung können wir annehmen, dass $\frac{N}{1+\varepsilon}$ ganzzahlig ist, bzw. wir können $\frac{N}{1+\varepsilon}$ auf die nächste ganze Zahl abrunden (was i.d.R. einer leichten Variation (Vergrösserung) von ε gleichkommt). Damit stehen wir bezüglich der Berechnung einer oberen Schranke gerade noch auf der sicheren Seite, denn weil gelten muss $f(\overline{x}_\varepsilon) > \frac{1}{1+\varepsilon} f_{max}$, können wir die Iterationen abbrechen, wenn gilt $f(\overline{x}_\varepsilon) = \lfloor \frac{N}{1+\varepsilon} \rfloor + 1$. Der Mehraufwand R an Rechenschritten, welchen die konsequente Aufrundung (im Gegensatz zur verwendeten Abrundung) der oberen Summationsgrenze k_ε höchstens mit sich bringen würde, lässt sich folgendermassen nach oben abgrenzen:

$$R < \frac{N}{N - \left(\frac{N}{1+\varepsilon} + 1\right)} = \frac{N(1+\varepsilon)}{N(1+\varepsilon) - N - (1+\varepsilon)} = \frac{N(1+\varepsilon)}{N\varepsilon - (1+\varepsilon)} \tag{13}$$

Der Term $\frac{N}{N - \left(\frac{N}{1+\varepsilon} + 1\right)}$ in (13) ist schlicht eine obere Schranke für das Summationsglied, welches bei der Aufrundung von $\frac{N}{1+\varepsilon}$ unnötigerweise in die Berechnung der Oberschranke einfliessen würde. Für $N \gg \frac{1}{\varepsilon}$ wird die Obergrenze für R ungefähr zu $\frac{1+\varepsilon}{\varepsilon}$, löst sich also von ihrer Abhängigkeit von N. Da $\varepsilon > 0$ ist, gilt auch $\lfloor \frac{N}{1+\varepsilon} \rfloor < N$, was uns erlaubt, R gleichzeitig mit der plumperen Schranke N nach oben abzuschätzen:

$$R \leq N = O(N) \tag{14}$$

Mit der abgerundeten Obergrenze des Summationsindex, welche wir als $k_{\varepsilon 2}$ bezeichnen, wobei $\varepsilon 2$ so zu wählen ist, dass $k_{\varepsilon 2} = \frac{N}{1+\varepsilon 2}$ dem auf die nächste

ganze Zahl abgerundeten Wert von $\frac{N}{1+\varepsilon}$ entspricht, lässt sich eine Obergrenze für die Ordnung des so adaptierten Problemes bestimmen zu:

$$E_{CO,\varepsilon 2}(N,\varepsilon 2) = \sum_{k=0}^{k_{\varepsilon 2}} E_{step,CO}(N,k)$$

$$= \sum_{k=0}^{\frac{1}{1+\varepsilon 2}N} \frac{N}{N-k} = \sum_{k=0}^{N-1} \frac{N}{N-k} - \sum_{k=\frac{1}{1+\varepsilon 2}N+1}^{N-1} \frac{N}{N-k} \tag{15}$$

Falls $\frac{N}{1+\varepsilon 2} = N-1$ ist, so ist $\sum_{k=0}^{N-1} \frac{N}{N-k} - \sum_{k=\frac{1}{1+\varepsilon 2}N+1}^{N-1} \frac{N}{N-k}$ die leere Summe und wir haben den Fall des nichtapproximativen Count Ones Problems. Fahren wir nun mit obigem Term (15) weiter:

$$E_{CO,\varepsilon 2}(N,\varepsilon 2) = \sum_{k=0}^{N-1} \frac{N}{N-k} - \sum_{k=\frac{1}{1+\varepsilon 2}N+1}^{N-1} \frac{N}{N-k}$$

$$= N\left(\sum_{k=0}^{N-1} \frac{1}{N-k} - \sum_{k=\frac{1}{1+\varepsilon 2}N+1}^{N-1} \frac{1}{N-k}\right) = N\left(\sum_{i=1}^{N} \frac{1}{i} - \sum_{i=1}^{N-\frac{1}{1+\varepsilon 2}N-1} \frac{1}{i}\right) \tag{16}$$

Den Term $N - \frac{1}{1+\varepsilon 2}N - 1$ in (16) kann man vereinfachen zu $\frac{\varepsilon 2}{1+\varepsilon 2}N - 1$. Somit lässt sich schreiben:

$$E_{CO,\varepsilon 2}(N,\varepsilon 2) = N\left(\sum_{i=1}^{N} \frac{1}{i} - \sum_{i=1}^{\frac{\varepsilon 2}{1+\varepsilon 2}N-1} \frac{1}{i}\right)$$

$$= N\left(\sum_{i=1}^{N} \frac{1}{i} - \sum_{i=1}^{\frac{\varepsilon 2}{1+\varepsilon 2}N} \frac{1}{i}\right) + \frac{N}{\frac{\varepsilon 2}{1+\varepsilon 2}N}$$

$$= N\left(\sum_{i=1}^{N} \frac{1}{i} - \sum_{i=1}^{\frac{\varepsilon 2}{1+\varepsilon 2}N} \frac{1}{i}\right) + \frac{1+\varepsilon 2}{\varepsilon 2} \tag{17}$$

Aufgrund der Eigenschaften von $\varepsilon 2$ gilt: $\frac{N}{1+\varepsilon 2}$ ist genau wie N ganzzahlig. Weiter gilt:

$$\frac{1}{1+\varepsilon 2}N \leq N-1 \Rightarrow N\left(1 - \frac{1}{1+\varepsilon 2}\right) = N\left(\frac{\varepsilon 2}{1+\varepsilon 2}\right) \geq 1$$

$$\Rightarrow \frac{\varepsilon 2}{1+\varepsilon 2} \geq \frac{1}{N} \Rightarrow \frac{1+\varepsilon 2}{\varepsilon 2} \leq N = O(N)$$

$$\Rightarrow E_{CO,\varepsilon 2}(N, \varepsilon 2) = N \left(\sum_{i=1}^{N} \frac{1}{i} - \sum_{i=1}^{\frac{\varepsilon 2}{1+\varepsilon 2}N} \frac{1}{i} \right) + O(N) \tag{18}$$

Wir wenden nun Ungleichung (6) auf (18) an. Dabei ist zu beachten, dass der Summenterm mit positivem Vorzeichen durch seine obere Begrenzung nach Ungleichung (6) ersetzt werden muss, der subtrahierte Summenterm hingegen durch die entsprechende untere Grenze, wollen wir eine obere Schranke für $E_{CO,\varepsilon 2}(N, \varepsilon 2)$ herleiten. Wir können also schreiben:

$$E_{CO,\varepsilon 2}(N, \varepsilon 2) < N \left(\ln N + 1 - \ln \left(\frac{\varepsilon 2}{1+\varepsilon 2} N \right) \right) + O(N)$$

$$= N \ln \left(\frac{N}{\frac{\varepsilon 2}{1+\varepsilon 2} N} \right) + \underbrace{N + O(N)}_{O(N)} = N \ln \left(\frac{N}{\frac{\varepsilon 2}{1+\varepsilon 2} N} \right) + O(N)$$

$$= N \ln \left(\frac{1}{\frac{\varepsilon 2}{1+\varepsilon 2}} \right) + O(N) = N \ln \left(\frac{1+\varepsilon 2}{\varepsilon 2} \right) + O(N)$$

$$= \tau N \log \left(\frac{1+\varepsilon 2}{\varepsilon 2} \right) + O(N) \tag{19}$$

Mit $\tau \in \mathbb{R}$. Wir formulieren somit den:

Satz 2.3 *Die erwartete Laufzeit des* $(1+1)$*-EA für das Auffinden einer Lösung des Count Ones Problems mit Problemgrösse* N *und relativer Approximationsgüte* $\varepsilon > 0$ *ist* $O\left(N \log\left(\frac{1+\varepsilon 2}{\varepsilon 2}\right)\right)$*, wobei* $\varepsilon 2$ *so zu wählen ist, dass* $\frac{N}{1+\varepsilon 2} = \lfloor \frac{N}{1+\varepsilon} \rfloor$.

$\frac{N}{1+\varepsilon 2} = \lfloor \frac{N}{1+\varepsilon} \rfloor$ wird maximal $N-1$, weil $\varepsilon > 0$. Setzt man $\frac{N}{1+\varepsilon 2} = N-1$, so wird $\varepsilon 2 = \frac{1}{N-1}$. Setzt man dieses $\varepsilon 2$ zur Kontrolle in $N \log\left(\frac{1+\varepsilon 2}{\varepsilon 2}\right)$ ein, so erhält man $N \log N$, genau wie bei der nichtapproximativen Laufzeitberechnung. Damit wird unser Resultat plausibel.

Ganz streng genommen kann man auch sagen:

$$E_{CO,\varepsilon 2}(N, \varepsilon 2) < N \left(\ln(N) + 1 - \ln \left(\frac{\varepsilon 2}{1+\varepsilon 2} N \right) \right) + \underbrace{\frac{1+\varepsilon 2}{\varepsilon 2}}_{\leq N}$$

$$\leq N\left(\ln N - \ln\left(\frac{\varepsilon 2}{1+\varepsilon 2}N\right)\right) + 2N = N\left(\ln\left(\frac{1+\varepsilon 2}{\varepsilon 2}\right) + 2\right) \qquad (20)$$

Mit (20) erhält man eine leicht modifizierte obere Schranke. Da der konstante Term in der Klammer gegenüber dem logarithmischen Term vernachlässigbar wird mit $\varepsilon 2 \to 0$, wollen wir ihn auch vernachlässigen, da wir ja hier v.a. am Einfluss der Approximationsgüte ε bzw. $\varepsilon 2$ auf die Laufzeit interessiert sind.

$\varepsilon 2$ ist ja i.d.R. grösser als ε, jedenfalls mindestens gleich gross. Würde man die obere Summationsgrenze aufrunden zu $\lceil\frac{N}{1+\varepsilon}\rceil$, kann man dies gleichsetzen mit der Einführung eines $\varepsilon 2'$, sodass $\frac{N}{1+\varepsilon 2'} = \lceil\frac{N}{1+\varepsilon}\rceil$. Es gilt:

$$\varepsilon 2 \geq \varepsilon \geq \varepsilon 2' \Rightarrow \frac{1+\varepsilon 2}{\varepsilon 2} \leq \frac{1+\varepsilon}{\varepsilon} \leq \frac{1+\varepsilon 2'}{\varepsilon 2'}$$

$$\Rightarrow N\log\left(\frac{1+\varepsilon 2}{\varepsilon 2}\right) + 2N \leq N\log\left(\frac{1+\varepsilon}{\varepsilon}\right) + 2N \leq N\log\left(\frac{1+\varepsilon 2'}{\varepsilon 2'}\right) + 2N \quad (21)$$

Weiter oben haben wir den Fehler R hergeleitet, der bei der konsequenten Aufrundung der oberen Summationsgrenze entstehen würde. Dieser wurde berechnet, indem man die Maximalgrösse des durch die Aufrundung der oberen Summationsgrenze überflüssigen Summanden bestimmte. Die Überlegungen führten zum Resultat $R \leq N$. $N\log\left(\frac{1+\varepsilon 2'}{\varepsilon 2'}\right) + 2N$ aus Formel (21) ist durch diesen Maximalfehler R nach oben abgegrenzt. Es gilt nämlich:

$$N\log\left(\frac{1+\varepsilon 2'}{\varepsilon 2'}\right) + 2N \leq N\log\left(\frac{1+\varepsilon 2}{\varepsilon 2}\right) + 2N + R \leq N\log\left(\frac{1+\varepsilon 2}{\varepsilon 2}\right) + 3N \qquad (22)$$

Wobei wir hier den Fall ausschliessen, wo $\frac{N}{1+\varepsilon 2} = N-1$ bzw. $\varepsilon 2' = 0$. Also gilt:

$$N\log\left(\frac{1+\varepsilon 2}{\varepsilon 2}\right) + 2N \leq N\log\left(\frac{1+\varepsilon}{\varepsilon}\right) + 2N \leq N\log\left(\frac{1+\varepsilon 2}{\varepsilon 2}\right) + 3N \quad (23)$$

Wieder interessieren uns im konkreten Fall die Terme, die einzig von N abhängen, nicht. $N\log\left(\frac{1+\varepsilon}{\varepsilon}\right)$ ist sozusagen von gleicher Ordnung wie $N\log\left(\frac{1+\varepsilon 2}{\varepsilon 2}\right)$. Wollen wir den Gebrauch der Hilfsvariablen $\varepsilon 2$ vermeiden, so können wir sagen:

Satz 2.4 *Für $\frac{N}{1+\varepsilon} < N-1$ ist die erwartete Laufzeit des $(1+1)$-EA bis zum Auffinden einer ε-approximierten Lösung des Count Ones Problems mit Problemgrösse N von $O\left(N\log\left(\frac{1+\varepsilon}{\varepsilon}\right)\right)$, sonst $O(N\log N)$.*

Wollen wir die genaue Ordnung Θ des approximativen Count Ones Problems ermitteln, so erreichen wir dies, wenn wir eine untere Schranke für die Laufzeit $T_{CO,\varepsilon}(N)$ finden, die von gleicher Ordnung wie eine gültige obere Schranke ist. Eine untere Schranke können wir z.B. finden wie bei der Herleitung des Θ beim nichtapproximativen Count Ones Problems, wo wir eine minimale Laufzeiterwartung herleiten konnten, die für mindestens 50% aller möglichen $\overline{x}_0$ gilt. Dazu machen wir wieder unsere Fallunterscheidung:

Fall 1: N gerade

In mehr als der Hälfte der Fälle ist $f(\overline{x}_0) \leq \frac{N}{2}$. Ist $f(\overline{x}_0) \leq \frac{N}{2}$, so ist die Mindestlaufzeit bis zum Auffinden einer approximativen Lösung gegeben durch $\sum_{k=\frac{N}{2}}^{\frac{N}{1+\varepsilon 2}} \frac{N}{N-k}$. Wir können für die Laufzeit $T_{CO,\varepsilon}(N)$ bis zur approximativen Optimierung von $\overline{x}$ folgende Unterschranke angeben:

$$
\begin{aligned}
T_{CO,\varepsilon}(N) &> \frac{1}{2} \sum_{k=\frac{N}{2}}^{\frac{N}{1+\varepsilon 2}} \frac{N}{N-k} = \frac{N}{2} \sum_{k=\frac{N}{2}}^{\frac{N}{1+\varepsilon 2}} \frac{1}{N-k} \\
&= \frac{N}{2} \sum_{i=\frac{\varepsilon 2}{1+\varepsilon 2}N}^{\frac{N}{2}} \frac{1}{i} = \frac{N}{2} \left(\sum_{i=1}^{\frac{N}{2}} \frac{1}{i} - \sum_{i=1}^{\frac{\varepsilon 2}{1+\varepsilon 2}N-1} \frac{1}{i} \right) \\
&= \frac{N}{2} \left(\sum_{i=1}^{\frac{N}{2}} \frac{1}{i} - \sum_{i=1}^{\frac{\varepsilon 2}{1+\varepsilon 2}N} \frac{1}{i} \right) + \frac{1+\varepsilon 2}{2\varepsilon 2} \qquad (24)
\end{aligned}
$$

Da $\varepsilon 2$ mindestens $\frac{1}{N-1}$ ist (im Falle, wenn $\frac{N}{1+\varepsilon 2} = N-1$), ist $\frac{1+\varepsilon 2}{2\varepsilon 2}$ maximal $\frac{N}{2}$, also $O(N)$. Es ist zu beachten, dass wir hier den additiven Summenterm durch seine Untergrenze gemäss (6) ersetzen müssen und den subtrahierten Summenterm durch seine Obergrenze, denn wir sind auf der Suche nach einer unteren Laufzeitschranke. Bei der Herleitung der Obergrenze war dies genau umgekehrt. Unter Anwendung von Ungleichung (6) mit $\tau \in \mathbb{R}$ schreiben wir:

$$
\begin{aligned}
T_{CO,\varepsilon}(N) &> \frac{N}{2} \left(\sum_{i=1}^{\frac{N}{2}} \frac{1}{i} - \sum_{i=1}^{\frac{\varepsilon 2}{1+\varepsilon 2}N} \frac{1}{i} \right) + O(N) \\
&> \frac{N}{2} \left(\ln\left(\frac{N}{2}\right) - \ln\left(\frac{\varepsilon 2}{1+\varepsilon 2}N\right) - 1 \right) + O(N) \\
&= \frac{N}{2} \left(\tau \log\left(\frac{N}{2}\right) - \tau \log\left(\frac{\varepsilon 2}{1+\varepsilon 2}N\right) - 1 \right) + O(N)
\end{aligned}
$$

$$= \frac{\tau N}{2}\left(\log N - \log\left(\frac{\varepsilon 2}{1+\varepsilon 2}N\right)\right) + \underbrace{\tau \log\left(\frac{1}{2}\right)\frac{N}{2} - \frac{N}{2} + O(N)}_{O(N)}$$

$$= \frac{\tau N}{2}\log\left(\frac{1+\varepsilon 2}{\varepsilon 2}\right) + O(N) \tag{25}$$

Den Term O(N) vernachlässigen wir. Somit ist $T_{CO,\varepsilon}(N)$ von $\Omega\left(N \log\left(\frac{1+\varepsilon 2}{\varepsilon 2}\right)\right)$ für gerade N.

Fall 2: N ungerade

Bei mehr als der Hälfte der möglichen (und gleich wahrscheinlichen) $\overline{x}_0$, bei denen noch keine Lösungsapproximation gegeben ist, gilt $f(\overline{x}_0) \leq \frac{N-1}{2}$. Für solche $\overline{x}_0$ ist die erwartete Zahl Iterationen bis zur approximativen Optimierung jeweils mindestens $\sum_{k=\frac{N-1}{2}}^{\frac{N}{1+\varepsilon 2}} \frac{N}{N-k}$

Als untere Schranke für $T_{CO,\varepsilon}(N)$ können wir schreiben:

$$T_{CO,\varepsilon}(N) > \frac{1}{2}\sum_{k=\frac{N-1}{2}}^{\frac{N}{1+\varepsilon 2}} \frac{N}{N-k} = \frac{N}{2}\sum_{k=\frac{N-1}{2}}^{\frac{N}{1+\varepsilon 2}} \frac{1}{N-k}$$

$$= \frac{N}{2}\sum_{i=\frac{\varepsilon 2}{1+\varepsilon 2}N}^{\frac{N+1}{2}} \frac{1}{i} = \frac{N}{2}\left(\sum_{i=1}^{\frac{N+1}{2}} \frac{1}{i} - \sum_{i=1}^{\frac{\varepsilon 2}{1+\varepsilon 2}N-1} \frac{1}{i}\right)$$

$$= \frac{N}{2}\left(\sum_{i=1}^{\frac{N+1}{2}} \frac{1}{i} - \sum_{i=1}^{\frac{\varepsilon 2}{1+\varepsilon 2}N} \frac{1}{i}\right) + \frac{\frac{N}{2}}{\frac{\varepsilon 2}{1+\varepsilon 2}N}$$

$$= \frac{N}{2}\left(\sum_{i=1}^{\frac{N+1}{2}} \frac{1}{i} - \sum_{i=1}^{\frac{\varepsilon 2}{1+\varepsilon 2}N} \frac{1}{i}\right) + \underbrace{\frac{1+\varepsilon 2}{2\varepsilon 2}}_{\leq \frac{N}{2}}$$

$$\Rightarrow T_{CO,\varepsilon}(N) > \frac{N}{2}\left(\sum_{i=1}^{\frac{N+1}{2}} \frac{1}{i} - \sum_{i=1}^{\frac{\varepsilon 2}{1+\varepsilon 2}N} \frac{1}{i}\right) + O(N)$$

$$\Rightarrow T_{CO,\varepsilon}(N) > \frac{N}{2}\left(\tau \log\left(\frac{N+1}{2}\right) - \tau \log\left(\frac{\varepsilon 2}{1+\varepsilon 2}N\right) - 1\right) + O(N)$$

$$= \frac{\tau N}{2}\left(\log(N+1) - \log\left(\frac{\varepsilon 2}{1+\varepsilon 2}N\right)\right) + \underbrace{\frac{N}{2}\tau \log\left(\frac{1}{2}\right) - \frac{N}{2} + O(N)}_{O(N)} \tag{26}$$

Es gilt $\log(N+1) > \log N$, deshalb kann man anstelle von (26) auch schreiben:

$$\begin{aligned} T_{CO,\varepsilon}(N) &> \frac{\tau N}{2}\left(\log N - \log\left(\frac{\varepsilon 2}{1+\varepsilon 2}N\right)\right) + O(N) \\ &= \frac{\tau N}{2}\log\left(\frac{1+\varepsilon 2}{\varepsilon 2}\right) + O(N) \end{aligned} \tag{27}$$

Auch für ungerade N ist $T_{CO,\varepsilon}(N)$ also $\Omega\left(N\log\left(\frac{1+\varepsilon 2}{\varepsilon 2}\right)\right)$. Untere und obere Grenze für die Laufzeit $T_{CO,\varepsilon}(N)$ sind also von gleicher Ordnung, was uns erlaubt, zu sagen:

Satz 2.5 *Die erwartete Laufzeit des* (1+1)*-EA zum Auffinden einer* ε*-approximativen Lösung des Count Ones Problems mit Problemgrösse* N *ist* $\Theta\left(N\log\left(\frac{1+\varepsilon 2}{\varepsilon 2}\right)\right)$*, wobei* $\varepsilon 2$ *so zu wählen ist, dass* $\frac{N}{1+\varepsilon 2} = \lfloor\frac{N}{1+\varepsilon}\rfloor$ *mit* $\varepsilon > 0$*.*

Aus ähnlichen Überlegungen wie bei Satz 2.4 könnte man auch in Satz 2.5 $\Theta\left(N\log\left(\frac{1+\varepsilon}{\varepsilon}\right)\right)$ verwenden.

2.2 Leading Ones Optimierproblem

Beim Leading Ones Problem errechnet sich die Zielfunktion $f(\overline{x})$ eines Bitstrings $\overline{x} = (x_1, x_2, \ldots, x_N)$ der Länge N nach der Anzahl der auf 1 gesetzten Bits, die den Bitstring anführen, ohne dass diese durch eine 0 unterbrochen werden:

Definition 2.2 *Die Leading Ones Funktion ist wie folgt definiert:*

$$f(\overline{x}) = \sum_{k=1}^{N}\prod_{i=1}^{k} x_i$$

mit $x_i \in \mathbb{B}$ *und* $N \in \mathbb{N}$

Zum besseren Verständnis seien in Abbildung 4 drei Beispiele gegeben.

Wir wollen nun das Problem mit unserem $(1+1)$-EA optimieren, also indem wir zufällig ein Bit auswählen, es invertieren und unseren Bitstring $\overline{x}$ durch den variierten String $\overline{x}'$ ersetzen, sofern gilt $f(\overline{x}') \geq f(\overline{x})$.

Wir wollen zuerst eine obere Schranke für die Laufzeit der Problemoptimierung bestimmen. Dazu starten wir mit $\overline{x}_0 = \overline{0}$. Wir bewegen uns nur in

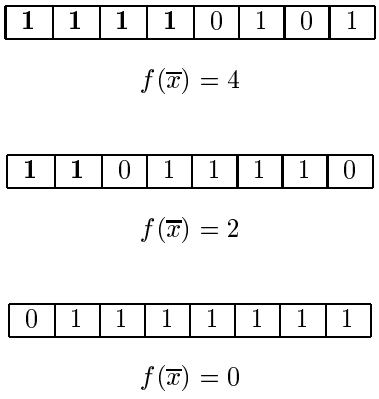

Abbildung 4: Beispiele für Leading Ones Problem mit $N = 8$

1er-Schritten bezüglich $f(\overline{x})$ voran, ganz im Sinne unseres Worst Case Szenarios. Es sei hier kurz bemerkt, dass in der Praxis natürlich auch Schritte grösser als 1 vorkommen, wir schliessen dies hier jedoch aus, damit wir bezüglich der Berechnung einer oberen Schranke auf der sicheren Seite sind. Solch ein 1er-Schritt gelingt uns nur dann, wenn wir jeweils die erste 0 im Bitstring zum Invertieren auswählen. Die Chance dafür beträgt in jedem Iterationsschritt $\frac{1}{N}$, denn es existiert nur eine einzige solche erste 0 in $\overline{x}$. Aus der Behandlung des Count Ones Problems wissen wir, dass die erwartete Anzahl Iterationsschritte (Anzahl Variationen) bis zur Auswahl genau dieses bestimmten Bits N beträgt, denn die Bitstringvariationen erfolgen unabhängig von den vorangegangenen.

Um das Maximum von $f(\overline{x})$ zu erreichen, müssen wir also genau N Schritte mit Erwartungswert N pro Schritt vollziehen, also N^2 Schritte. Wir wissen wiederum, dass die einzelnen Erwartungswerte für die Iterationsanzahl, um in $f(\overline{x})$ einen Schritt voranzukommen, unabhängig sind von der Anzahl gebrauchter Iterationsschritte für vorangehende und nachfolgende Fortschritte in $f(\overline{x})$, was uns erlaubt, die einzelnen Erwartungswerte aufzusummieren. Der $(1+1)$-EA ist wie schon erwähnt ohne Gedächtnis. Daraus können wir folgern:

Satz 2.6 *Die erwartete Laufzeit des* $(1+1)$*-EA angewendet auf die Leading Ones Optimieraufgabe der Problemgrösse* N *ist* $O(N^2)$.

Wir haben also eine obere Schranke für das Leading Ones Problem hergeleitet, indem wir angenommen haben, dass wir jedes der N Bits im Bitstring $\overline{x}$ invertieren mussten, was uns jeweils im Schnitt N Variationsschritte kostete, erwartungshalber insgesamt also genau N^2 Schritte. Um die genaue Ordnung der erwarteten Laufzeit des Leading Ones Problems zu bestimmen, gehen wir nun davon aus, dass wir nur jedes zweite Bit tatsächlich invertieren müssen, jeweils auch mit Aufwand N. Diese Annahme ist zulässig, denn wir starten

unseren $(1+1)$-EA mit einem Bitstring $\overline{x}_0$, in dem jedes einzelne Bit von den anderen Bits unabhängig mit Wahrscheinlichkeit $\frac{1}{2}$ bereits auf 1 gesetzt ist und sämtliche Bits, welche auf die erste Null in einem Bitstring folgen, sind auch mit Wahrscheinlichkeit $\frac{1}{2}$ bereits auf 1 gesetzt, ebenfalls unabhängig von den anderen Bits und vom Zeitpunkt. Desweiteren braucht die Invertierung eines auf 0 gesetzten Bits im Durchschnitt immer N Schritte, unabhängig davon, in welchen Zuständen sich die restlichen Bits befinden. Somit müssen wir tatsächlich im Mittel nur bei jedem zweiten Bit die durchschnittlich N Schritte bis zur Invertierung abwarten, in der anderen Hälfte der Fälle ist das kritische Bit bereits auf 1 gesetzt, und wir haben an der betreffenden Stelle im Bitstring keinen Zeitaufwand. Formal können wir für den erwarteten Totalaufwand $T_{LO}(N)$ sagen:

$$T_{LO}(N) = \sum_{k=0}^{N-1} E_{step,LO}(N,k) p_{k+1} \tag{28}$$

Mit $E_{step,LO}(N,k)$, der erwarteten Anzahl Variationen, bis das Bit x_{k+1} invertiert wird und p_{k+1}, der Wahrscheinlichkeit, dass das Bit x_{k+1} auf 0 gesetzt ist (und somit invertiert werden muss). p_{k+1} ist zu jeder Zeit und für jedes N und k immer gleich $\frac{1}{2}$ und $E_{step,LO}(N,k)$ ist immer gleich N. Somit beträgt die zu erwartende Anzahl Variationen zum Erreichen des globalen Maximums beim Leading Ones Problems exakt:

$$T_{LO}(N) = \sum_{k=0}^{N-1} E_{step,LO}(N,k) p_{k+1} = \sum_{k=0}^{N-1} N\frac{1}{2} = \frac{N^2}{2} \tag{29}$$

Gegenüber dem Worst Case Szenario mit genau N^2 Variationen sparen wir also im zu erwartenden Falle 50% der Rechenzeit. Wir formulieren somit folgenden:

Satz 2.7 *Die erwartete Laufzeit des* $(1+1)$*-EA angewendet auf die Leading Ones Optimieraufgabe der Problemgrösse* N *ist* $\frac{N^2}{2} = \Theta(N^2)$.

Gibt man sich indes mit einer approximativen Lösung zufrieden, indem man die Iterationen abbricht, sobald gilt $f(\overline{x}) > \frac{1}{1+\varepsilon} f(\overline{x}^*) = \frac{1}{1+\varepsilon} f_{max}$, so muss man nur $\frac{N}{1+\varepsilon}$ mal einen Fortschritt in $f(\overline{x})$ erreichen, womit für dieses Problem der folgende Satz aufgestellt werden kann:

Satz 2.8 *Die erwartete Laufzeit des* $(1+1)$*-EA für das Auffinden einer Lösung des Leading Ones Problems mit Problemgrösse* N *und relativer Approximationsgüte* ε *ist* $O\left(\frac{N^2}{1+\varepsilon}\right)$.

Setzt man zur Kontrolle $\varepsilon = 0$, so wird $\frac{N^2}{1+\varepsilon}$ wieder zu N^2, womit unser Resultat plausibel ist.

Die Überlegung, welche uns auf die genaue Ordnung Θ des nichtapproximativen Leading Ones Problems geführt hat, ist auch im approximativen Fall gültig, denn auch hier müssen wir die einzelnen Bits mit jeweiligem Aufwand N lediglich jeweils mit Wahrscheinlichkeit $\frac{1}{2}$ invertieren, was uns unseren zu erwartenden Rechenaufwand gegenüber dem Worst Case Szenario bei der Herleitung der oberen Schranke um die Hälfte reduziert. Um ein approximatives Maximum des Leading Ones Problems zu erreichen, muss man also im Durchschnitt mit einem Aufwand von genau $\frac{N^2}{2(1+\varepsilon)}$ Variationen rechnen:

Satz 2.9 *Die erwartete Laufzeit des* $(1+1)$*-EA für das Auffinden einer Lösung des Leading Ones Problems mit Problemgrösse* N *und relativer Approximationsgüte* ε *ist* $\Theta\left(\frac{N^2}{1+\varepsilon}\right)$.

Auch hier ergibt das Einsetzen von $\varepsilon = 0$ ein sinnvolles Resultat, nämlich genau das Θ des nichtapproximativen Leading Ones Problems.

2.3 $(1+1)$-EA$_p$ angewendet auf das Leading Ones Problem

In der Einleitung haben wir alternativ zu unserem $(1+1)$-EA auch den $(1+1)$-EA$_p$ festgelegt, welcher beim Variationsschritt so vorgeht, dass jedes Bit mit derselben Wahrscheinlichkeit p unabhängig von den anderen Bits invertiert wird. Wir wollen für den $(1+1)$-EA$_p$ wiederum eine obere Schranke für das Leading Ones Problem bestimmen, d.h. wir starten mit dem schlechtestmöglichen Bitstring $\overline{x}_0 = \overline{0} = (0, 0, 0, \ldots, 0, 0)$ und ignorieren die Tatsache, dass man bezüglich $f(\overline{x})$ durch Variation von $\overline{x}$ auch Fortschritte machen kann, die grösser als 1 sind. Die Fortschrittswahrscheinlichkeit bez. $f(\overline{x})$ bei $f(\overline{x}) = k$, also wenn die ersten k Bits auf 1 gesetzt sind und das Bit x_{k+1} 0 ist, nennen wir $p_{advance}(N, p, k)$:

$$p_{advance}(N, p, k) = (1-p)^k p \tag{30}$$

Dabei beschreibt der Term $(1-p)^k$ die Wahrscheinlichkeit, dass keines der Leading Ones auf 0 zurückgesetzt (invertiert) wird und p ist die Wahrscheinlichkeit, dass das Bit x_{k+1} von 0 auf 1 flippt. Wie wir schon aus der Behandlung des Count Ones Problems wissen, verhalten sich die Wahrscheinlichkeit eines Ereignisses und die durchschnittliche Anzahl Versuche bis zum ersten Auftreten dieses Ereignisses reziprok zueinander. Daher kann die Anzahl Variationen $T_{advance}(N, p, k)$, die man erwartet um in $f(\overline{x})$ einen Fortschritt zu machen, folgendermassen ausgedrückt werden:

$$T_{advance}(N,p,k) = \frac{1}{p_{advance}(N,p,k)} = \frac{1}{p(1-p)^k} \tag{31}$$

Bemerkenswert scheint hier die Tatsache, dass $T_{advance}(N,p,k)$ nicht von N abhängig ist. Eine obere Schranke für den Erwartungswert der Problemlaufzeit $T_{total}(N,p)$ erhalten wir durch Summation der $T_{advance}(N,p,k)$ für $0 \leq k < N$:

$$T_{total}(N,p) \leq \sum_{k=0}^{N-1} T_{advance}(N,p,k) = \sum_{k=0}^{N-1} \frac{1}{p(1-p)^k} = \frac{1}{p} \sum_{k=0}^{N-1} \left(\frac{1}{1-p}\right)^k \tag{32}$$

Ein Blick in eine geeignete Formelsammlung liefert die folgende Gleichung für $c \neq 1$:

$$\sum_{i=0}^{n} c^i = \frac{c^{n+1}-1}{c-1} \tag{33}$$

(33) angewendet auf (32) ergibt mit $c = \frac{1}{1-p}$, mit $1 > p > 0 \Rightarrow c > 1$:

$$T_{total}(N,p) \leq \frac{1}{p} \frac{\left(\frac{1}{1-p}\right)^N - 1}{\left(\frac{1}{1-p}\right) - 1} = \frac{1}{p} \frac{\left(\frac{1}{1-p}\right)^N - 1}{\left(\frac{1}{1-p}\right) - 1} \left(\frac{1-p}{1-p}\right)$$

$$= \frac{1}{p} \frac{\left(\frac{1}{1-p}\right)^{N-1} - (1-p)}{1-(1-p)} = \frac{\left(\frac{1}{1-p}\right)^{N-1} - 1 + p}{p^2} \tag{34}$$

Setzen wir nun $p = \frac{1}{N}$ in das obige Ergebnis (34) ein, d.h. wir wählen dieselbe 'Mutationsfreudigkeit' bzw. Mutationsstärke (die erwartete Anzahl geflippter Bits pro Variationsschritt) wie beim $(1+1)$-EA, so bekommen wir als zu erwartenden Aufwand:

$$T_{total}\left(N, p = \frac{1}{N}\right) \leq \frac{\left(\frac{1}{1-\frac{1}{N}}\right)^{N-1} - 1 + \frac{1}{N}}{\frac{1}{N^2}}$$

$$= \frac{\left(\frac{N}{N-1}\right)^{N-1}}{\frac{1}{N^2}} - N^2 + N = N^2 \left(\frac{N}{N-1}\right)^{N-1} - N^2 + O(N) \tag{35}$$

Mit der Substitution $N = j + 1$ kann man schreiben:

$$T_{total}\left(N, p = \frac{1}{N}\right) \leq N^2 \left(\frac{j+1}{j}\right)^j - N^2 + O(N) \tag{36}$$

Aus der Formelsammlung erhalten wir die Ungleichung für die Eulersche Zahl e:

$$\left(\frac{j+1}{j}\right)^j < e < \left(\frac{j+1}{j}\right)^{j+1} \tag{37}$$

Mit dieser Euler-Ungleichung (37) können wir folgern:

$$N^2 \left(\left(\frac{j+1}{j}\right)^j - 1\right) + O(N) < N^2(e-1) + O(N) = O(N^2) \tag{38}$$

Wir stellen somit fest, dass unser $(1 + 1)$-EA$_p$ genau wie der $(1 + 1)$-EA höchstens quadratische Laufzeit in N aufweist, sofern man $p = \frac{1}{N}$ setzt. Wollen wir eine untere Schranke für den Erwartungswert der Problemlaufzeit bestimmen, so können wir uns überlegen, dass wir jedes Bit nur mit Wahrscheinlichkeit $\frac{1}{2}$ unter Zeitaufwand zu invertieren haben, bzw. dass bei allen 2^N gleich wahrscheinlichen Abfolgen von N jeweils zu invertierenden oder eben nicht zu invertierenden Bits jedes Bit gleich viele Male zu invertieren ist wie nicht. Dies ist nicht weiter erstaunlich, denn weder bei der Initialisierung des Bitstrings, noch bei der Variation werden entweder auf 0 oder auf 1 gesetzte Bits bevorzugt. Gegenüber dem Worst Case, wo alle N Bits invertiert werden müssen (vergleiche (32)), sparen wir also in Wirklichkeit 50% an Aufwand:

$$T_{total}(N, p) = \frac{1}{2} \sum_{k=0}^{N-1} T_{advance}(N, p, k) \tag{39}$$

mit Hilfe von (35) können wir direkt schreiben:

$$\begin{aligned} T_{total}(N, p) &= \frac{1}{2} N^2 \left(\frac{N}{N-1}\right)^{N-1} - \frac{N^2}{2} + \frac{N}{2} \\ &= \frac{1}{2} N^2 \frac{N-1}{N} \left(\frac{N}{N-1}\right)^N - \frac{N^2}{2} + \frac{N}{2} \end{aligned} \tag{40}$$

Wieder verwenden wir die Substitution $N = j + 1$:

$$T_{total}(N,p) = \frac{1}{2}N^2\frac{N-1}{N}\underbrace{\left(\frac{j+1}{j}\right)^{j+1}}_{Euler:>e} - \frac{N^2}{2} + \frac{N}{2}$$

$$\Rightarrow T_{total} > \frac{1}{2}N^2 e - \frac{1}{2}Ne - \frac{N^2}{2} + \frac{N}{2} \Rightarrow T_{total}(N, p = \frac{1}{N}) = \Omega(N^2) \qquad (41)$$

Obere und untere Schranke sind also von gleicher Ordnung. Daraus folgt:

Satz 2.10 *Mit $p = \frac{1}{N}$ ist die erwartete Laufzeit des $(1+1)$-EA_p angewendet auf das Leading Ones Problem $\Theta(N^2)$.*

Setzt man $p = \frac{1}{2}$, so wird die Laufzeit unseres $(1+1)$-EA_p exponentiell in N:

$$T_{total}\left(N, p = \frac{1}{2}\right) = \frac{1}{2}\frac{\left(\frac{1}{1-\frac{1}{2}}\right)^{N-1} - 1 + \frac{1}{2}}{\frac{1}{2^2}} = \frac{2^{N-1}}{\frac{1}{2}} - \frac{1}{\frac{1}{2}} + \frac{\frac{1}{2}}{\frac{1}{2}}$$
$$= 2^N - 1 = O(\exp N) = O\left(2^N\right) \qquad (42)$$

Dieses Verhalten ist nicht weiter erstaunlich, denn mit $p = \frac{1}{2}$ verhält sich unser $(1+1)$-EA_p genau so wie ein Random-Search Algorithmus, der mit gleicher Wahrscheinlichkeit einen Bitstring von 2^N möglichen auswählt und den besten aller bisher eruierten Bitstrings im Archiv speichert. Die Chance, dass dieser Random-Search Algorithmus das globale Maximum der Leading Ones Funktion herauspickt ist $\frac{1}{2^N}$, die erwartete Laufzeit des Random-Search ist daher 2^N, d.h. exponentiell.

Wir können auch für das Leading Ones Problem eine approximative Lösung finden mit unserem $(1+1)$-EA_p. Setzen wir $N_\varepsilon = \lfloor \frac{N}{1+\varepsilon} \rfloor$, so lässt sie die erwartete Schrittanzahl bis zum Erreichen der ε-relativ approximativen Lösung $T_{total,\varepsilon}(N,p,\varepsilon)$ mit folgendem Ausdruck genau bestimmen (Vgl. (34) und (39)):

$$T_{total,\varepsilon}(N,p,\varepsilon) = \frac{1}{2}\frac{\left(\frac{1}{1-p}\right)^{N_\varepsilon} - 1 + p}{p^2} = \frac{1}{2}\frac{\left(\frac{1}{1-p}\right)^{N_\varepsilon}}{p^2} + \frac{p-1}{2p^2} \qquad (43)$$

Setzen wir nun wiederum $p = \frac{1}{N}$ ein, so erhalten wir:

$$T_{total,\varepsilon}\left(N, p = \frac{1}{N}, \varepsilon\right) = \frac{1}{2}\frac{\left(\frac{1}{1-\frac{1}{N}}\right)^{N_\varepsilon} - 1 + \frac{1}{N}}{\frac{1}{N^2}}$$

$$= \frac{1}{2}\frac{\left(\frac{1}{1-\frac{1}{N}}\right)^{N_\varepsilon}}{\frac{1}{N^2}} + \frac{1}{2}\frac{\frac{1}{N}-1}{\frac{1}{N^2}} = \frac{1}{2}\left(\frac{1}{1-\frac{1}{N}}\right)^{N_\varepsilon} N^2 - \frac{1}{2}N^2 + \frac{1}{2}N$$

$$= \frac{1}{2}\left(\left(\frac{N}{N-1}\right)^{N_\varepsilon} - 1\right)N^2 + \frac{N}{2} \tag{44}$$

Eigentlich könnten wir ja mit dem exakten Resultat in (44) zufrieden sein. Um aber dieses Resultat besser mit den Resultaten im nichtapproximativen Fall vergleichen zu können, wollen wir (44) nun doch nach oben und unten abgrenzen. Wegen $\frac{N}{N-1} > 1$ und $\lfloor\frac{N}{1+\varepsilon}\rfloor > \frac{N}{1+\varepsilon} - 1$ gilt:

$$T_{total,\varepsilon} = \frac{1}{2}\left(\left(\frac{N}{N-1}\right)^{N_\varepsilon} - 1\right)N^2 + \frac{N}{2}$$

$$> \frac{1}{2}\left(\frac{N-1}{N}\right)\left(\frac{N}{N-1}\right)^{\frac{N}{1+\varepsilon}} N^2 - \frac{N^2}{2} + \frac{N}{2} \tag{45}$$

Substitution $N = j + 1$ ergibt:

$$T_{total,\varepsilon}(N, p = \frac{1}{N}, \varepsilon) > \frac{1}{2}\left(\frac{N-1}{N}\right)\left(\sqrt[1+\varepsilon]{\underbrace{\left(\frac{j+1}{j}\right)^{j+1}}_{Euler:>e}}\right)N^2 - \frac{N^2}{2} + \frac{N}{2}$$

$$> \frac{1}{2}\left(\frac{N-1}{N}\right)\sqrt[1+\varepsilon]{e}N^2 - \frac{N^2}{2} + \frac{N}{2}$$

$$= \frac{1}{2}\sqrt[1+\varepsilon]{e}N^2 - \frac{1}{2}\sqrt[1+\varepsilon]{e}N - \frac{N^2}{2} + \frac{N}{2} \tag{46}$$

Für die Terme $-\frac{1}{2}\sqrt[1+\varepsilon]{e}N$ und $\frac{N}{2}$ interessieren wir uns nicht, da sie mit grossen N marginalisiert werden gegenüber $\frac{1}{2}\sqrt[1+\varepsilon]{e}N^2$ und $-\frac{N^2}{2}$. Das Problem ist somit $\Omega\left(\left(\sqrt[1+\varepsilon]{e} - 1\right)N^2\right)$.

Für eine obere Schranke rechnen wir wie folgt:

$$T_{total,\varepsilon} = \frac{1}{2}\left(\left(\frac{N}{N-1}\right)^{N_\varepsilon} - 1\right)N^2 + \frac{N}{2}$$

$$\leq \frac{1}{2}\left(\frac{N}{N-1}\right)^{\frac{N}{1+\varepsilon}} N^2 - \frac{N^2}{2} + \frac{N}{2}$$

$$< \frac{1}{2}\left(\frac{N}{N-1}\right)\left(\frac{N}{N-1}\right)^{\frac{N-1}{1+\varepsilon}} N^2 - \frac{N^2}{2} + \frac{N}{2} \tag{47}$$

Wieder gebrauchen wir die Substitution $N = j + 1$:

$$T_{total,\varepsilon}(N, p = \frac{1}{N}, \varepsilon) < \frac{1}{2}\left(\frac{N}{N-1}\right) \sqrt[1+\varepsilon]{\underbrace{\left(\frac{j+1}{j}\right)^j}_{<e}} N^2 - \frac{N^2}{2} + \frac{N}{2}$$

$$< \frac{1}{2}\frac{N}{N-1}\sqrt[1+\varepsilon]{e}N^2 - \frac{N^2}{2} + \frac{N}{2} \tag{48}$$

Für den Term $\frac{N}{2}$ interessieren wir uns wiederum nicht. Das Problem ist daher $O\left(\left(\frac{N}{N-1}\sqrt[1+\varepsilon]{e} - 1\right)N^2\right)$. Somit scheint sich der erwartete Aufwand, um mit dem $(1+1)$-EA$_p$ mit $p = \frac{1}{N}$ eine ε-approximative Lösung des Leading Ones Problems zu finden, für grosse N etwa wie $\frac{1}{2}\left(\sqrt[1+\varepsilon]{e} - 1\right)N^2$ zu verhalten, denn der Faktor $\frac{N}{N-1}$ in der oberen Schranke kommt beliebig nahe an 1 heran für wachsende N.

2.4 Binary Value Problem

Mit unserem $(1+1)$-EA können wir auch das Binary Value Problem angehen, d.h. wir gewichten den Wert einzelner Bits nach ihrer Stelle im Bitstring exponentiell ansteigend und bilden $f(\overline{x})$ eben durch gewichtete Summation dieser Bits:

Definition 2.3 *Die Binary Value Funktion ist wie folgt definiert:*

$$f(\overline{x}) = \sum_{k=1}^{N} 2^{k-1} x_k$$

mit $x_k \in \mathbb{B}$ *und* $N \in \mathbb{N}$

Das Bit, welches der Potenz 2^{N-1} zugeordnet wird, nennt man 'Most Significant Bit' (MSB), dasjenige, welches der niedrigsten Potenz $2^0 = 1$ zugeordnet wird, nennt man dementsprechend 'Least Significant Bit' (LSB). In unserem Falle ist also x_1 das das LSB und x_N das MSB.

Unter Anwendung des $(1+1)$-EA sieht man leicht, dass die Bestimmung der zu erwarteten Rechenschritte auf das Resultat des Count Ones Optimierproblems hinausläuft, denn eine Verbesserung in $f(\overline{x})$ wird immer nur dann erzielt, falls ein auf 0 gesetztes Bit zu einer 1 wird. Dies berechtigt uns zu folgender Aussage:

Satz 2.11 *Die erwartete Laufzeit des* $(1+1)$*-EA zur Optimierung des Binary Value Problems der Problemgrösse* N *ist* $\Theta(N \log N)$.

Schwieriger ist hingegen die Frage nach der Laufzeit, wenn man sich mit einer approximativen Lösung zufriedengibt. Im Gegensatz zum Count Ones Problem spielt es hier nämlich eine zentrale Rolle, an welcher Stelle die auf 1 oder 0 gesetzten Bits stehen. $f(\overline{x})$ wird nur dann maximal, wenn $\overline{x} = \overline{x}^* = (1,1,1,\ldots,1,1)$, also wenn alle N Bits auf 1 gesetzt sind und somit ihren mehr oder weniger grossen Beitrag zu $f(\overline{x})$ leisten. Wir wollen mit dem $(1+1)$-EA irgendein $f(\overline{x}_\varepsilon)$ finden, für das gilt $f(\overline{x}_\varepsilon) > \frac{1}{1+\varepsilon} f_{max} = \frac{1}{1+\varepsilon} f(\overline{x}^*)$ mit $\varepsilon > 0$.

f_{max} ist $2^N - 1$. Dies kann man induktiv leicht beweisen. Der Beitrag des MSB zu $f(\overline{x})$ beträgt $2^{N-1} = \frac{2^N}{2}$, was für jedes $N \geq 1$ mehr als der Hälfte von f_{max} entspricht. Das am zweitstärksten gewichtete Bit trägt (sofern auf 1 gesetzt) mehr als einen Viertel zu f_{max} bei, das am drittstärksten gewichtete Bit mehr als einen Achtel, usw. Sind in einem Bitstring also beispielsweise die ersten beiden Bits auf 1 gesetzt, so beträgt der Binärwert dieses $\overline{x}$ mehr als $\left(\frac{1}{2} + \frac{1}{4}\right) f_{max}$, d.h. mehr als $\frac{3}{4} f_{max}$, unabhängig vom Zustand der anderen Bits und v.a. unabhängig von N. Ganz allgemein lässt sich sagen:

Satz 2.12 *Sind in einem Bitstring der Länge* $N \geq 1$ *die* j *signifikantesten Bits auf* 1 *gesetzt, so ist dessen Binärwert grösser als* $\frac{2^j - 1}{2^j}(2^N - 1)$.

Unser $f(\overline{x}_\varepsilon)$ soll nun grösser sein als $\frac{1}{1+\varepsilon} f_{max} = \frac{1}{1+\varepsilon}(2^N - 1)$. Dies ist wegen des obigen Satzes sicher bei einem $\overline{x}_\varepsilon$ erfüllt, dessen j signifikanteste Bits auf 1 gesetzt sind, wenn für dieses $j \in \mathbb{N}$ gilt:

$$\frac{1}{1+\varepsilon} \leq \frac{2^j - 1}{2^j} \tag{49}$$

Wir wollen den Term (49) nach j auflösen:

$$\begin{aligned} &\frac{1}{1+\varepsilon} \leq \frac{2^j - 1}{2^j} \Rightarrow 1 - \frac{1}{1+\varepsilon} \geq 1 - \frac{2^j - 1}{2^j} \\ &\Rightarrow \frac{\varepsilon}{1+\varepsilon} \geq \frac{1}{2^j} \Rightarrow \frac{1+\varepsilon}{\varepsilon} \leq 2^j \Rightarrow j \geq \lceil lb\left(\frac{1+\varepsilon}{\varepsilon}\right) \rceil \end{aligned} \tag{50}$$

lb bezeichnet dabei den Logarithmus zur Basis 2. Wir müssen $lb\left(\frac{1+\varepsilon}{\varepsilon}\right)$ aufrunden, denn $j \in \mathbb{N}$ und $\frac{2^j-1}{2^j}$ wächst monoton für $j \geq 1$. Wir können also eine obere Schranke für den Erwartungswert der Laufzeit des $(1+1)$-EA für das Auffinden einer ε-approximativen Lösung des Binary Value Problems herleiten, indem wir mit dem Worst Case Bitstring $\overline{x}_0 = \overline{0}$ starten und ausrechnen, wieviele Variationen wir im Schnitt abzuwarten haben, bis die signifikantesten $\lceil lb\left(\frac{1+\varepsilon}{\varepsilon}\right)\rceil$ Bits von $\overline{x}$ auf 1 gesetzt sind. Was dabei mit den restlichen Bits geschieht, spielt keine Rolle. Sind k der signifikantesten $\lceil lb\left(\frac{1+\varepsilon}{\varepsilon}\right)\rceil$ Bits auf 1 gesetzt mit $k \in \{0, 1, \ldots, \lceil lb\left(\frac{1+\varepsilon}{\varepsilon}\right)\rceil - 1\}$ und ist der Bitstring N Bits lang, so beträgt die Wahrscheinlichkeit $p_{advance,BV}(N,k)$, dass man beim Variationsschritt des $(1+1)$-EA innerhalb der signifikantesten $\lceil lb\left(\frac{1+\varepsilon}{\varepsilon}\right)\rceil$ Bits eine 0 auf eine 1 ändert:

$$p_{advance,BV}(N,k) = \frac{\lceil lb\left(\frac{1+\varepsilon}{\varepsilon}\right)\rceil - k}{N} \tag{51}$$

Da unsere Bitstringvariationen voneinander unabhängig sind, ist die Erwartung, dass man

$$T_{advance,BV}(N,k) = \frac{1}{p_{advance,BV}(N,k)} = \frac{N}{\lceil lb\left(\frac{1+\varepsilon}{\varepsilon}\right)\rceil - k} \tag{52}$$

Variationen abwarten muss, bis eine 0 in den signifikantesten $\lceil lb\left(\frac{1+\varepsilon}{\varepsilon}\right)\rceil$ Bits zu einer 1 wird. Wir bilden nun unsere obere Schranke für den Erwartungswert der Laufzeit $T_{BV,\varepsilon}(N)$ des approximativ gelösten Binary Value Problems durch Summierung der $T_{advance,BV}(N,k)$ für $k \in \{0, 1, \ldots, \lceil lb\left(\frac{1+\varepsilon}{\varepsilon}\right)\rceil - 1\}$:

$$\begin{aligned} T_{BV,\varepsilon}(N) &\leq \sum_{k=0}^{\lceil lb\left(\frac{1+\varepsilon}{\varepsilon}\right)\rceil - 1} T_{advance,BV}(N,k) \\ &= \sum_{k=0}^{\lceil lb\left(\frac{1+\varepsilon}{\varepsilon}\right)\rceil - 1} \frac{N}{\lceil lb\left(\frac{1+\varepsilon}{\varepsilon}\right)\rceil - k} = N \sum_{k=0}^{\lceil lb\left(\frac{1+\varepsilon}{\varepsilon}\right)\rceil - 1} \frac{1}{\lceil lb\left(\frac{1+\varepsilon}{\varepsilon}\right)\rceil - k} \end{aligned} \tag{53}$$

Wir vertauschen die Summationsreihenfolge der zu summierenden Brüche:

$$T_{BV,\varepsilon}(N) \leq \sum_{i=1}^{\lceil lb\left(\frac{1+\varepsilon}{\varepsilon}\right)\rceil} \frac{1}{i} \tag{54}$$

Aus der Behandlung des Count Ones Problems wissen wir das Resultat bereits (Vergleiche (5) bis (7)):

$$T_{BV,\varepsilon}(N) = O\left(N \log\left(\lceil lb\left(\frac{1+\varepsilon}{\varepsilon}\right)\rceil\right)\right) \tag{55}$$

Wir wollen den Fall ausschliessen, wo das approximative und das nichtapproximative Problem identisch sind. Wegen $f(\overline{x}_\varepsilon) > \frac{1}{1+\varepsilon} f_{max}$ bedeutet dies, dass $\frac{1}{1+\varepsilon} f_{max} < f_{max} - 1$, woraus folgende Bedingung ableitbar ist: $\frac{1+\varepsilon}{\varepsilon} < f_{max} = 2^N - 1$. Damit formulieren wir einen Satz für die Obergrenze der erwarteten Laufzeit des $(1+1)$-EA zum Auffinden einer ε-approximativen Lösung des Binary Value Problems:

Satz 2.13 *Die erwartete Laufzeit des $(1+1)$-EA auf das ε-approximative Binary Value Problem angewendet ist $O\left(N \log\left(\lceil lb\left(\frac{1+\varepsilon}{\varepsilon}\right)\rceil\right)\right)$, sofern $\frac{1+\varepsilon}{\varepsilon} < 2^N - 1$ und $O(N \log N)$ sonst.*

Wir wissen jetzt, ab wievielen signifikantesten Bits von $\overline{x}_\varepsilon$, die geschlossen auf 1 gesetzt sind, wir sicher sein können, dass gilt $f(\overline{x}_\varepsilon) > \frac{1}{1+\varepsilon} f_{max}$.

Nun wollen wir herleiten, wieviele signifikanteste Bits in $\overline{x}_?$ mindestens geschlossen auf 1 stehen müssen, damit noch gelten kann $f(\overline{x}_?) > \frac{1}{1+\varepsilon} f_{max}$. Sei $\overline{x}_?$ nun also ein Bitstring, dessen j signifikanteste Bits alle auf 1 gesetzt sind, das nächstsignifikante Bit allerdings 0 ist. x_{N+1-j} bis x_N seien also gleich 1, x_{N-j}, dessen Beitrag zu $f(\overline{x}_?)$, falls auf 1 gesetzt, 2^{N-1-j} betragen würde, sei 0. $f(\overline{x}_?)$ kann somit maximal $f_{max} - 2^{N-1-j} = (2^N - 1) - 2^{N-1-j}$ betragen. Gesucht ist nun das kleinste $j \in \mathbb{N}$, für das gilt:

$$f_{max} - 2^{N-1-j} > \frac{1}{1+\varepsilon} f_{max} \tag{56}$$

Wir lösen diese Ungleichung nach j auf (beachte: $f_{max} = 2^N - 1$):

$$\begin{aligned}
f_{max} - 2^{N-1-j} > \frac{1}{1+\varepsilon} f_{max} &\Rightarrow f_{max} - \frac{1}{1+\varepsilon} f_{max} > 2^{N-1-j} \\
&\Rightarrow \frac{\varepsilon}{1+\varepsilon} f_{max} > 2^{N-1-j} \Rightarrow \frac{\varepsilon}{1+\varepsilon} \frac{f_{max}}{2^{N-1}} > 2^{-j} \\
&\Rightarrow \frac{1+\varepsilon}{\varepsilon} \frac{2^{N-1}}{f_{max}} < 2^j \Rightarrow lb\left(\frac{1+\varepsilon}{\varepsilon}\right) + \underbrace{lb\left(\frac{2^{N-1}}{f_{max}}\right)}_{<0} < j \\
&\Rightarrow j = \lceil lb\left(\frac{1+\varepsilon}{\varepsilon}\right)\rceil
\end{aligned} \tag{57}$$

Meist genügt es aber auch, wenn wir $j = \lceil lb\left(\frac{1+\varepsilon}{\varepsilon}\right)\rceil - 1$ als die Mindestanzahl signifikantester Bits, die auf 1 gesetzt sein sollen, wählen. $j = \lceil lb\left(\frac{1+\varepsilon}{\varepsilon}\right)\rceil$ signifikanteste Bits brauchen wir nur im Falle, wo für $i \in \mathbb{N}, 1 \leq i \leq N-1$ gilt:

$$\frac{f_{max}}{2^i - \frac{1}{2^{N-i}}} \geq \frac{1+\varepsilon}{\varepsilon} \geq \frac{f_{max}}{2^i} \tag{58}$$

in diesen $N-1$ Intervallen gilt $\lceil lb\left(\frac{1+\varepsilon}{\varepsilon}\right)\rceil > lb\left(\frac{1+\varepsilon}{\varepsilon}\frac{2^{N-1}}{f_{max}}\right) \geq \lceil lb\left(\frac{1+\varepsilon}{\varepsilon}\right)\rceil - 1$.

Um eine untere Schranke für die Laufzeit $T_{BV,\varepsilon}(N)$ für das ε-approximativ gelöste Binary Value Problem herzuleiten, müsste man die erwartete Laufzeit für jeden der $2^{\lceil lb\left(\frac{1+\varepsilon}{\varepsilon}\right)\rceil - 1}$ verschiedenen Zustände der signifikantesten $\lceil lb\left(\frac{1+\varepsilon}{\varepsilon}\right)\rceil - 1$ Bits berücksichtigen, denn jeder dieser Zustände kommt im Initialisierungsbitstring $\overline{x}_0$ mit gleicher Wahrscheinlichkeit vor. Es entsteht der komplizierte Term (mit $\alpha = \lceil lb\left(\frac{1+\varepsilon}{\varepsilon}\right)\rceil - 1$):

$$T_{BV,\varepsilon}(N) \geq \sum_{s=0}^{\alpha-1}\left(\frac{1}{2^\alpha}\binom{\alpha}{s}\sum_{k=s}^{\alpha-1}\frac{N}{\alpha-k}\right) \tag{59}$$

Man vergleiche diesen Term auch mit (8). Aus der Behandlung des Count Ones Problems können wir direkt angeben, dass das approximierte Binary Value Problem $\Omega(N\log(\alpha)) = \Omega\left(N\log\left(\lceil lb\left(\frac{1+\varepsilon}{\varepsilon}\right)\rceil - 1\right)\right)$ ist. Siehe dazu (9) bis (12).

Wir interessieren uns nun dafür, ob sich für alle relevanten ε die untere Schranke der Ordnung $N\log\left(\lceil lb\left(\frac{1+\varepsilon}{\varepsilon}\right)\rceil - 1\right)$ (bzw. $N\log\left(\lceil lb\left(\frac{1+\varepsilon}{\varepsilon}\right)\rceil\right)$ für die Intervalle nach (58)) und die obere Schranke der Ordnung $N\log\left(\lceil lb\left(\frac{1+\varepsilon}{\varepsilon}\right)\rceil\right)$ so in Beziehung setzen lassen, dass diese nur durch einen konstanten Faktor getrennt sind. ε kann per Definition maximal 1 sein. Ist $\varepsilon = 1$, so sind wir im Spezialintervall mit $i = 1$ gemäss (58). Es ist dann $\lceil lb\left(\frac{1+\varepsilon}{\varepsilon}\right)\rceil = 1$. Das ε-approximativ gelöste Binary Value Problem ist $\Omega(N\log\gamma)$ mit $\gamma = \lceil lb\left(\frac{1+\varepsilon}{\varepsilon}\right)\rceil - 1$ oder $\gamma = \lceil lb\left(\frac{1+\varepsilon}{\varepsilon}\right)\rceil$ und es sei $O(N\log\beta)$ mit $\beta = \lceil lb\left(\frac{1+\varepsilon}{\varepsilon}\right)\rceil$. γ ist mindestens 1 für alle in Frage kommenden ε (wie gesagt auch für $\varepsilon = 1$), und β ist höchstens um 1 grösser als γ. $\frac{\beta}{\gamma}$ kann also maximal 2 sein. Wir können somit für alle relevanten ε sagen, dass approximative Binary Value Problem sei $\Omega(N\log\gamma)$ und $O(N\log(2\gamma))$ bzw. es sei $\Omega\left(N\log\left(\frac{\beta}{2}\right)\right)$ und $O(N\log(\beta))$. Es ist $N\log\left(\frac{\beta}{2}\right) = N\log\beta - N\log 2 = N\log\beta + O(N)$. Obere und untere Schranke sind also im Prinzip von gleicher Ordnung, wenn man die Einschränkungen bezüglich des Maximalwerts von ε mitberücksichtigt, denn sie unterscheiden sich nur durch einen untergeordneten Term voneinander. Somit können wir sagen mit $\beta = \lceil lb\left(\frac{1+\varepsilon}{\varepsilon}\right)\rceil$:

Satz 2.14 *Die erwartete Laufzeit des* $(1+1)$*-EA beim* ε*-approximierten Binary*

Value Problem ist für $\frac{1+\varepsilon}{\varepsilon} < 2^N - 1$ *von* $\Theta\left(N \log\left(\lceil lb\left(\frac{1+\varepsilon}{\varepsilon}\right)\rceil\right)\right)$ *und* $\Theta(N \log N)$ *sonst.*

2.5 Zusammenfassung Kapitel 2

Wir rekapitulieren kurz die wichtigsten Erkenntnisse dieses Kapitels:

- Das Count Ones Problem mit dem $(1+1)$-EA ist $\Theta(N \log N)$ optimiert und $\Theta\left(N \log\left(\frac{1+\varepsilon}{\varepsilon}\right)\right)$ im approximativen Fall.
- Das Leading Ones Problem ist mit dem $(1+1)$-EA $\Theta(N^2)$ und $\Theta\left(\frac{N^2}{1+\varepsilon}\right)$ approximativ gelöst.
- Die erwartete Laufzeit des $(1+1)$-EA$_p$ beim Leading Ones Problem ist $\Theta(N^2)$ für $p = \frac{1}{N}$ und exponentiell für $p = \frac{1}{2}$. Für $p + \frac{1}{N}$ hat man immer mit einem Aufwand zu rechnen, der grösser als $\frac{1}{2}\left(\sqrt[1+\varepsilon]{e} - 1\right) N^2$ ist, um mit dem $(1+1)$-EA$_p$ eine approximative Lösung des Leading Ones Problems zu finden. Mit wachsendem N kommt der tatsächlich zu erwartende Aufwand jedoch sehr nahe an $\frac{1}{2}\left(\sqrt[1+\varepsilon]{e} - 1\right) N^2$ heran.
- Das Binary Value Problem ist $\Theta(N \log N)$ optimiert und $\Theta\left(N \log\left(\lceil lb\left(\frac{1+\varepsilon}{\varepsilon}\right)\rceil\right)\right)$ im approximativ gelösten Falle.

3 (1 + 1)-MOEA angewendet auf Optimierprobleme mit mehreren Zielfunktionen

3.1 Das $LOTZ$ Problem

Bisher haben wir uns mit Optimieraufgaben beschäftigt, in welchen unsere Bitstrings bzw. Genotypen bezüglich nur einer einzigen Zielfunktion zu optimieren waren. Laumanns, Zitzler und Thiele haben in [3] unter der Bezeichnung $LOTZ$ (Leading Ones, Trailing Zeros) ein einfaches Modellproblem mit zwei Zielfunktionen beschrieben. Es müssen dabei sowohl die Leading Ones als auch die Trailing Zeros des Suchvektors maximiert werden. Man kann die $LOTZ$ Funktion als Erweiterung der Leading Ones Funktion auf zwei Dimensionen betrachten. Formal lautet die Definition von $LOTZ$:

Definition 3.1 ($LOTZ$) *Die Funktion* $LOTZ\colon \mathbb{B}^N \to \mathbb{N}^2$ *ist wie folgt definiert:*

$$LOTZ(\overline{x}) = (LOTZ_1(\overline{x}), LOTZ_2(\overline{x})) = \left(\sum_{i=1}^{N} \prod_{j=1}^{i} x_j, \sum_{i=1}^{N} \prod_{j=i}^{N} (1 - x_j) \right)$$

Es ist leicht einzusehen, dass $LOTZ_1(\overline{x})$ und $LOTZ_2(\overline{x})$ nicht durch einen einzigen Bitstring gleichzeitig maximiert werden können, daher ist zu erwarten, dass mehrere paretooptimale Suchvektoren existieren. Der Zielraum des $LOTZ$ Problems lässt sich in Untermengen F_i mit $i \in \{0, 1, 2, ..., N\}$ unterteilen, wobei $i = LOTZ_1(\overline{x}) + LOTZ_2(\overline{x})$. Siehe dazu Abbildung 5.

Wie leicht zu sehen ist, entspricht F_N der Paretofront, F_{N-1} ist leer. Durch eine Mutation eines Bits von $\overline{x}$ kann nur entweder bezüglich $LOTZ_1(\overline{x})$ oder $LOTZ_2(\overline{x})$ ein Fortschritt erreicht werden, dann nämlich, wenn entweder die erste 0 oder die letzte 1 im Bitstring mutiert wird. Die Wahrscheinlichkeit, dass man bei der Invertierung eines zufälligen Bits entweder bezüglich $LOTZ_1(\overline{x})$ oder $LOTZ_2(\overline{x})$ einen Fortschritt macht (und somit ausgehend von einem Bitstring $\overline{x}$ einen dominanten Bitstring $\overline{x}'$ generiert), ist $\frac{2}{N}$ (doppelt so gross also wie beim Leading Ones Problem), folglich muss man darauf im Durchschnitt $\frac{N}{2}$ Schleifendurchläufe warten. Spätestens nach $N-1$ Fortschritten (F_{N-1} ist leer) jeweils bezüglich $LOTZ_1(\overline{x})$ oder $LOTZ_2(\overline{x})$ ist man mit dem in der Einleitung beschriebenen $(1+1)$-MOEA von jedem beliebigen Startvektor $\overline{x}_0$ aus mit dem Vektor $LOTZ(\overline{x})$ auf der Paretofront angelangt. (Es geht natürlich auch früher, denn Fortschritte grösser als 1 bezüglich $LOTZ_1(\overline{x})$ oder $LOTZ_2(\overline{x})$ sind möglich und auch nur jeder vierte Startvektor ist zum Worst Case Szenario kompatibel, indem er mit einer 0 beginnt und mit einer 1 endet.) Der $(1+1)$-MOEA benötigt im Durchschnitt also nicht mehr als $\frac{N}{2}(N-1) = \frac{N^2}{2} - \frac{N}{2}$ Bitstringva-

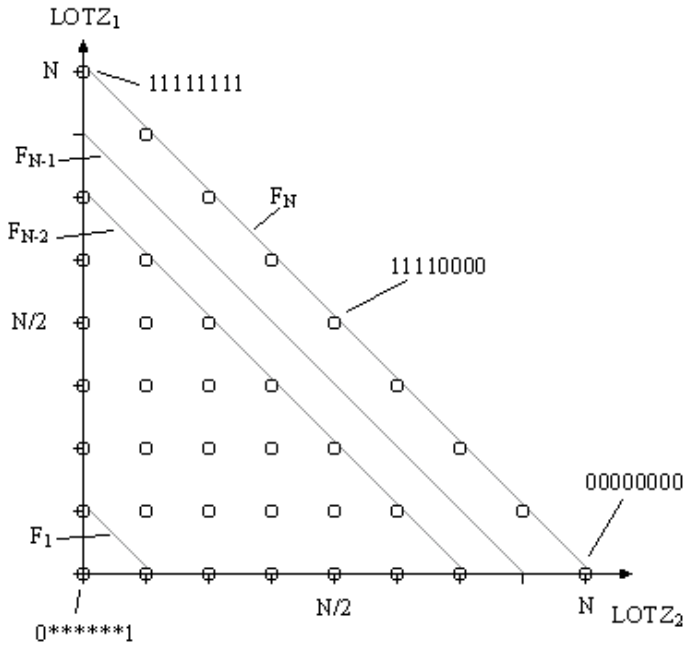

Abbildung 5: Der Zielraum der Funktion $LOTZ$

riationen bzw. Funktionsevaluationen, um ein $\overline{x}^*$ aus der Paretomenge X^* zu finden. Das Problem ist mit dem $(1+1)$-MOEA angegangen also $O(N^2)$.

Um eine untere Schranke für die Problemlaufzeit des $(1+1)$-MOEA angewendet auf $LOTZ$ zu finden, unterteilen wir unseren Bitstring in Gruppen zu je 2 aufeinanderfolgenden Bits x_j und x_{j+1}. Es existieren in einem Bitstring der Länge N mindestens $\frac{N-1}{2}$ solcher Zweiergruppen. Gegeben sei ein Beispiel mit $N = 11$:

$$\underbrace{x_1\,x_2}\;\underbrace{x_3\,x_4}\;\underbrace{x_5\,x_6}\;x_7\;\underbrace{x_8\,x_9}\;\underbrace{x_{10}\,x_{11}}$$

Wir können Bits auch immer erst dann initialisieren, wenn sie zur Berechnung der Funktionswerte relevant werden, bzw. wir können sie uninitialisiert lassen, solange man sie zur Funktionsberechnung nicht benötigt. Folgt eine noch nicht initialisierte Zweiergruppe auf eine geschlossene Gruppe von Leading Ones Bits, oder geht sie einer Gruppe von Trailing Zeros voran, so ist ihr Inhalt sicher relevant für die Berechnung des $LOTZ$-Funktionsvektors, und man initialisiert folglich diese Zweiergruppe. Mit Wahrscheinlichkeit $\frac{1}{4}$ und unabhängig vom

Zustand aller anderen Bits gilt für die initialisierte Zweiergruppe $x_j = 0$ und $x_{j+1} = 1$. Eine solche Gruppe stellt sowohl für die Leading Ones wie auch für die Trailing Zeros ein 'Hindernis' dar, der Bitstring ist nicht paretooptimal und es ist somit mindestens ein weiterer Fortschritt bezüglich $LOTZ_1(\overline{x})$ oder $LOTZ_2(\overline{x})$ notwendig. Die Erreichung eines solchen Fortschritts nimmt bei nicht paretooptimalen Suchvektoren im Durchschnitt immer $\frac{N}{2}$ Schleifendurchläufe in Anspruch, da er nur durch Invertierung entweder der ersten 0 oder der letzten 1 im Bitstring erfolgen kann, also durch die Mutation von 2 aus N Bits und somit mit Wahrscheinlichkeit $\frac{2}{N}$. Auf dem Weg zu einem paretooptimalen Suchvektor mit dem (1+1)-MOEA müssen wir mindestens $\frac{N-1}{2}$ Bitpaare initialisieren. Deren Initialisierung provoziert mindestens in einem Viertel der Fälle einen Aufwand von $\frac{N}{2}$ Schleifendurchläufen bzw. Suchvektorevaluationen, unabhängig von den Zuständen der übrigen Bits. Der erwartete Aufwand des $(1+1)$-MOEA zum Auffinden eines paretooptimalen Suchvektors (untere Schranke) ist daher mindestens:

$$T_{LOTZ,(1+1)}(N) \geq \frac{N-1}{2}\frac{1}{4}\frac{N}{2} = \frac{N^2}{8} - \frac{N}{8} \tag{60}$$

Das $LOTZ$ Problem mit dem $(1+1)$-MOEA angegangen ist daher $\Omega(N^2)$. Obere und untere Schranke sind von gleicher Ordnung, somit folgt:

Satz 3.1 *Der erwartete Aufwand des* $(1+1)$*-MOEA für das Auffinden eines paretooptimalen Suchvektors des* $LOTZ$ *Problems ist* $\Theta(N^2)$.

Nun wollen wir wissen, wie lange es im Durchschnitt dauert, bis der $(1+1)$-MOEA für das $LOTZ$ Problem einen Suchvektor $\overline{x}_\varepsilon$ findet, dessen $LOTZ(\overline{x}_\varepsilon)$ mindestens einen Vektor aus der Paretofront $F^* = F_N$ ε-dominiert. Für alle Vektoren der Paretomenge $\overline{x}^* \in X^*$ gilt: $LOTZ_1(\overline{x}^*) + LOTZ_2(\overline{x}^*) = N$. Damit ein Vektor $\overline{x}_\varepsilon$ einen Vektor $\overline{x}^* \in X^*$ ε-dominieren kann, muss sicher gelten:

$$(1+\varepsilon)(LOTZ_1(\overline{x}_\varepsilon) + LOTZ_2(\overline{x}_\varepsilon)) \geq N \tag{61}$$

Beispielsweise ε-dominiert der Vektor $\overline{f}_\varepsilon = \left(\lceil \frac{N}{1+\varepsilon} \rceil, 0\right)$ den paretooptimalen Vektor $\overline{f}^* = (N, 0)$. Bedingung (61) reicht aber noch nicht in jedem Fall aus. Damit ein Vektor $\overline{x}_\varepsilon$ in jedem Fall ein $\overline{f}^* \in F_N$ ε-dominiert, muss folgende Bedingung erfüllt sein:

$$LOTZ_1(\overline{x}_\varepsilon) + LOTZ_2(\overline{x}_\varepsilon) \geq \frac{N}{1+\varepsilon} + \frac{1}{1+\varepsilon} \tag{62}$$

(62) ist in jedem Falle eine hinreichende Bedingung an $\overline{x}_\varepsilon$. Gilt (62) für ein $\overline{x}_\varepsilon$, so reicht der Scheitelpunkt A in Abbildung 8 des von $LOTZ(\overline{x}_\varepsilon)$ ε-dominierten Bereiches über die Gerade, welche die Koordinatenachsen $LOTZ_1$ und $LOTZ_2$ jeweils an der Stelle $N+1$ schneidet, hinaus. Dadurch schneidet der Bereich, der von $LOTZ(\overline{x}_\varepsilon)$ ε-dominiert wird, aus der Paretofront F_N auf der Geraden $LOTZ_1(\overline{x}^*)+LOTZ_2(\overline{x}^*) = N$ einen Abschnitt heraus, der länger oder gleich lang wie $\sqrt{2}$ ist. Auf einem solchen Abschnitt liegt sicher ein Paretopunkt, denn diese liegen auf ihrer verbindenden Geraden mit konstantem Abstand $\sqrt{2}$ auseinander. Mit (62) wird sichergestellt, dass gilt:

$$\lfloor (1+\varepsilon) LOTZ_1(\overline{x}_\varepsilon) \rfloor + \lfloor (1+\varepsilon) LOTZ_2(\overline{x}_\varepsilon) \rfloor \geq N \tag{63}$$

wodurch $\overline{x}_\varepsilon$ alle $\overline{x}^*$ ε-dominiert, für die gilt $LOTZ_1(\overline{x}^*) \leq \lfloor (1+\varepsilon) LOTZ_1(\overline{x}_\varepsilon) \rfloor$ *und* $LOTZ_2(\overline{x}^*) \leq \lfloor (1+\varepsilon) LOTZ_2(\overline{x}_\varepsilon) \rfloor$. Zum besseren Verständnis siehe die Abbildungen 6, 7 und 8.

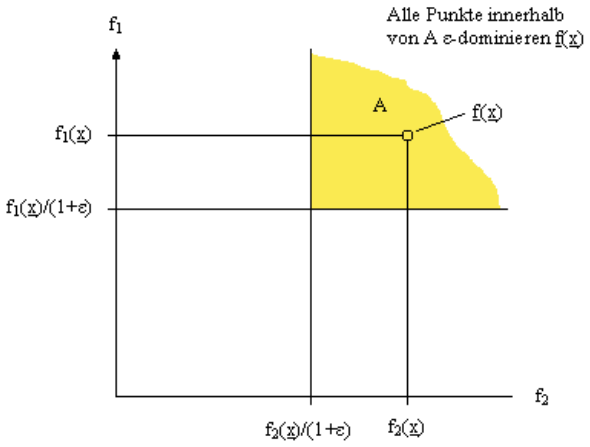

Abbildung 6: Bereich A, in dem ein ε-dominanter Vektor zu $\overline{f}(\overline{x})$ liegen kann

Wir betrachten den Fall, wo $\frac{1}{1+\varepsilon} \leq \frac{N-2}{N}$, da wir sonst das nichtapproximative $LOTZ$ Problem vorliegen hätten (Beachte: F_{N-1} ist leer). Für die Herleitung einer oberen Schranke wollen wir wissen, wieviele Schleifendurchläufe man beim $(1+1)$-MOEA maximal zu erwarten hat, wenn gelten soll $LOTZ_1(\overline{x}_\varepsilon) + LOTZ_2(\overline{x}_\varepsilon) = \lceil \frac{N}{1+\varepsilon} \rceil + 1 > \frac{N}{1+\varepsilon} + \frac{1}{1+\varepsilon}$. Wir benötigen dazu maximal $\left(\lceil \frac{N}{1+\varepsilon} \rceil + 1 \right) \frac{N}{2}$ Schleifendurchläufe (mit $\overline{x}_0 = (0,0,0,\ldots,0,0)$ und der Worst Case Annahme, dass wir bezüglich $LOTZ_1(\overline{x})$ und $LOTZ_2(\overline{x})$ nur Fortschritte der Grösse 1 machen). Es gilt:

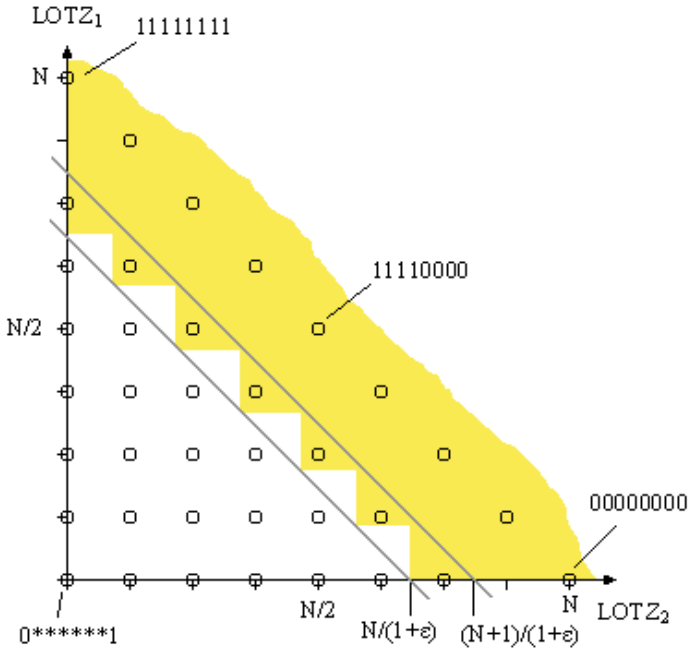

Abbildung 7: Minimale und Hinreichende Anforderungen an Zielraumvektoren, um einen Vektor der Paretomenge F_N zu ε-dominieren

$$T_{LOTZ,(1+1),\varepsilon}(N,\varepsilon) \leq \left(\lceil\frac{N}{1+\varepsilon}\rceil + 1\right)\frac{N}{2}$$
$$< \left(\frac{N}{1+\varepsilon} + 2\right)\frac{N}{2} = \frac{N^2}{2(1+\varepsilon)} + N \qquad (64)$$

Das Problem ist somit $O\left(\frac{N^2}{1+\varepsilon}\right)$.

Zur Herleitung einer unteren Schranke wollen wir wissen, wieviele Schleifendurchläufe man im Minimum zu erwarten hat, bis gilt $LOTZ_1(\overline{x}_\varepsilon)+LOTZ_2(\overline{x}_\varepsilon) = \lceil\frac{N}{1+\varepsilon}\rceil$, denn es muss ja gelten $LOTZ_1(\overline{x}_\varepsilon)+LOTZ_2(\overline{x}_\varepsilon) \geq \frac{N}{1+\varepsilon}$, und da wir uns hier mit diskreten Werten befassen, muss aufgerundet werden. Wir brauchen hier wieder das Gedankenmodell, welches wir bereits bei der Herleitung einer unteren Schranke für das nichtapproximative $LOTZ$ Problem zuvor in diesem Abschnitt verwendeten: Wir müssen mindestens $\frac{\lceil\frac{N}{1+\varepsilon}\rceil-1}{2}$ Gruppen zu 2 Bits initialisieren, wovon uns mindestens jede vierte solche Gruppe dazu zwingt, unter Zeitaufwand bezüglich $LOTZ_1(\overline{x})$ oder $LOTZ_2(\overline{x})$ einen Fortschritt zu machen.

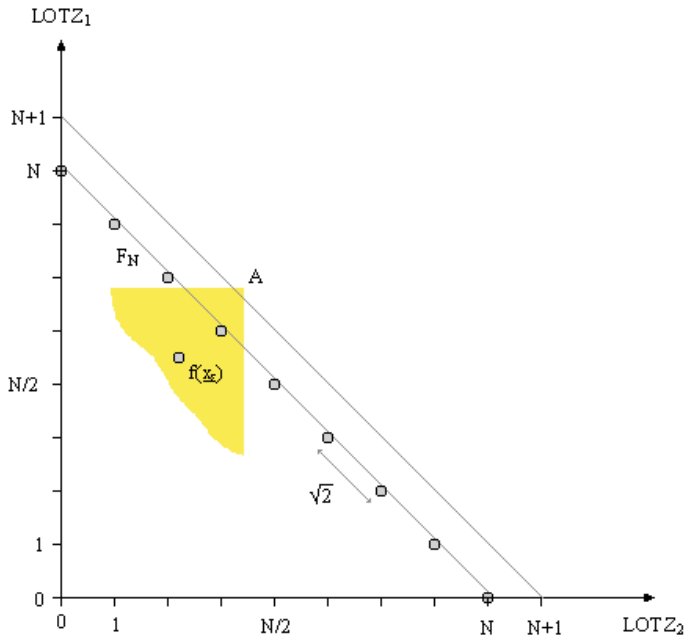

Abbildung 8: Hinreichende Anforderung an Zielraumvektoren, um einen Vektor der Paretomenge F_N zu ε-dominieren. Reicht A über die Gerade $LOTZ_1 + LOTZ_2 = N+1$ hinaus, so ε-dominiert $LOTZ(\overline{x}_\varepsilon)$ mindestens einen Vektor der Paretofront F_N.

Die betreffende Anzahl Schleifendurchläufe ist im Durchschnitt zum Erreichen eines solchen Fortschritts $\frac{N}{2}$. Es gilt somit

$$\begin{aligned} T_{LOTZ,(1+1),\varepsilon}(N,\varepsilon) &\geq \frac{\lceil \frac{N}{1+\varepsilon} \rceil - 1}{2} \frac{1}{4} \frac{N}{2} \\ &\geq \frac{\frac{N}{1+\varepsilon} - 1}{2} \frac{1}{4} \frac{N}{2} = \frac{1}{16} \left(\frac{N^2}{1+\varepsilon} - N \right) \end{aligned} \tag{65}$$

Das Problem ist also $\Omega\left(\frac{N^2}{1+\varepsilon}\right)$, obere und untere Schranke sind also beide von quadratischer Ordnung in N, reduziert mit dem Faktor $\frac{1}{1+\varepsilon}$. Damit kann man sagen:

Satz 3.2 *Die erwartete Laufzeit des auf das LOTZ Problem angewendeten*

(1+1)-MOEA bis zum Auffinden eines ε-approximativen Suchvektors, der mindestens einen Vektor der Paretomenge F_N ε-dominiert, ist $\Theta\left(\frac{N^2}{1+\varepsilon}\right)$

3.2 Das $COCZ$ Problem

Neben dem $LOTZ$ Problem wurde in [3] auch das $COCZ$ Problem definiert. Es kann als das auf zwei Zielfunktionen erweiterte Count Ones Problem betrachtet werden. Ziel ist es, sowohl die Anzahl auf 1 gesetzter Bits, als auch die Summe der auf 1 gesetzten Bits in der ersten Hälfte des Bitstrings und der auf 0 gesetzter Bits in der zweiten Bitstringhälfte zu maximieren. Der Bitstring soll in zwei gleich grosse Hälften zerlegt werden, deshalb soll N ein ganzes Vielfaches von 2 sein. Formal lautet die Definition:

Definition 3.2 ($COCZ$) *Die Funktion* $COCZ : \mathbb{B}^N \to \mathbb{N}^2$ *ist wie folgt definiert:*

$$COCZ(\overline{x}) = (COCZ_1(\overline{x}), COCZ_2(\overline{x})) = \left(\sum_{i=1}^{N} x_i, \sum_{i=1}^{N/2} x_i + \sum_{i=N/2+1}^{N} (1 - x_i)\right)$$

mit $N = 2k, k \in \mathbb{N}$.

Wir können die Bildmenge F, bestehend aus $\left(\frac{N}{2}+1\right)^2$ Vektoren in $\frac{N}{2}+1$ Gruppen F_i mit $i \in \{0, \ldots, \frac{N}{2}\}$ unterteilen. F_i ist die Gruppe von Bildvektoren, die durch die Abbildung aller Entscheidungsvektoren mit i auf 1 gesetzten Bits in der ersten Bitstringhälfte entsteht. $F_{\frac{N}{2}}$ entspricht der Paretofront, $\frac{N}{2} - i$ ist die Hammingdistanz aller auf F_i abgebildeter Suchvektoren $\overline{x} \in X$ zu den Suchvektoren $\overline{x}^*$, die auf die Paretofront abgebildet werden. Siehe dazu auch Abbildung 9.

Jeder Suchvektor $\overline{x}$ hat nur direkte Hammingnachbarn $\overline{x}'$ mit $COCZ(\overline{x}) \neq COCZ(\overline{x}')$. Ein dominanter Vektor $\overline{x}'$ kann durch die 1-Bit Mutation des (1+1)-MOEA nur dann entstehen, wenn eine 0 in der ersten Hälfte des Bitstrings $\overline{x}$ zu einer 1 mutiert. Das $COCZ$ Problem, angegangen mit dem (1+1)-MOEA, entspricht somit einem Count Ones Problem für die erste Hälfte des Bitstrings. Die $\overline{x}_i$, also die Suchvektoren mit i zu 1 gesetzten Bits in der ersten Bitstringhälfte, sind in ihrer Anzahl binomialverteilt bezüglich i, d.h. Es gibt für $i \in \{0, \ldots, \frac{N}{2}\}$ jeweils $\binom{\frac{N}{2}}{i} 2^{\frac{N}{2}}$ Suchvektoren $\overline{x}_i \in X$. Nur durchschnittlich in jedem zweiten Mutationsschritt wird ein Bit aus der ersten Hälfte des Bitstrings mutiert, daher ist die erwartete Laufzeit des (1 + 1)-MOEA zum Auffinden eines paretooptimalen Suchvektors doppelt so gross wie die erwartete Lauftzeit des (1 + 1) zur Optimierung eines Bitstrings mit Problemgrösse $\frac{N}{2}$. Mit den Resultaten aus Abschnitt 2.1 kann man daraus direkt folgern:

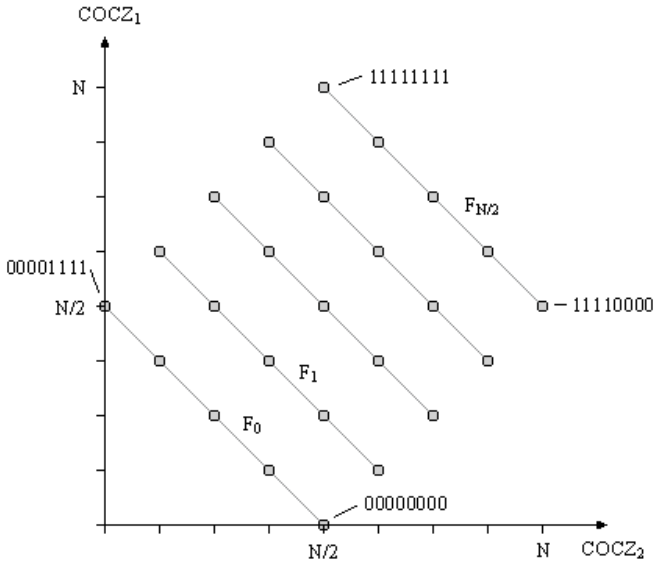

Abbildung 9: Der Zielraum des $COCZ$ Problems. $F_{\frac{N}{2}}$ entspricht der Paretofront

Satz 3.3 *Die erwartete Laufzeit des* $(1+1)$*-MOEA zum Auffinden eines paretooptimalen Suchvektors des* $COCZ$ *Problems ist* $\Theta(N \log N)$.

Wieder wollen wir wissen, wie lange der $(1+1)$-MOEA erwartungshalber braucht, bis er ein $\overline{x}_\varepsilon$ findet, für welches der Vektor $COCZ(\overline{x}_\varepsilon)$ ein Element der Paretofront $F_{\frac{N}{2}}$ ε-dominiert.

Für die Vektoren $\overline{x}^*$ der Paretofront gilt $COCZ_1(\overline{x}^*) + COCZ_2(\overline{x}^*) = \frac{3N}{2}$. Der beim Variationsschritt des $(1+1)$-MOEA hervorgehende Suchvektor $\overline{x}'$ vergrössert $COCZ_1(\overline{x}') + COCZ_2(\overline{x}')$ gegenüber $COCZ_1(\overline{x}) + COCZ_2(\overline{x})$ nur dann, wenn er $\overline{x}$ dominiert, und dies ist nur dann der Fall, wenn bei der Variation eine 0 in der ersten Bitstringhälfte von $\overline{x}$ zu einer 1 wird. Ist $\overline{x}'$ dominant zu $\overline{x}$, so macht die $COCZ$ Funktion sowohl bezüglich $COCZ_1(\overline{x})$ als auch bezüglich $COCZ_2(\overline{x})$ einen Schritt der Grösse 1, und da die Paretofront zur Richtung beider $COCZ$-Teilfunktionen in einem Winkel von 45 Grad steht, macht $COCZ(\overline{x})$ einen Schritt der Grösse $\sqrt{2}$ in Richtung der Paretofront (Bzw. in Richtung der Geraden, die die Paretopunkte verbindet). Man kann die Optimierung/Approximierung des $COCZ$ Problems somit als Optimierung/Approximierung des Count Ones Problems im Zielraum betrachten mit einer um $\sqrt{2}$ gestreckten Schrittweite. Das Ganze ist in Abbildung 10 de-

tailiert dargestellt und nochmals beschrieben. Damit $COCZ(\overline{x}_\varepsilon)$ einen Vektor aus der Paretomenge $F_{\frac{N}{2}}$ ε-dominieren kann, muss auf jeden Fall gelten:

$$(1+\varepsilon)(COCZ_1(\overline{x}_\varepsilon) + COCZ_2(\overline{x}_\varepsilon)) \geq \frac{3N}{2} \tag{66}$$

(Falls $N = 4s$, $s \in \mathbb{N}$, so ε-dominiert der Vektor $\left(\lceil\frac{3N}{4(1+\varepsilon)}\rceil, \lceil\frac{3N}{4(1+\varepsilon)}\rceil\right)$ den paretooptimalen Vektor $\left(\left(\frac{3N}{4}\right), \left(\frac{3N}{4}\right)\right)$.) Weil man mit der Generierung eines dominanten Vektors bezüglich $(COCZ_1(\overline{x}) + COCZ_2(\overline{x}))$ einen Schritt der Grösse 2 macht, der Vektor $COCZ(\overline{x})$ seinen Abstand von der Paretofront dabei aber nur um $\sqrt{2}$ verringert, darf ein Vektor $COCZ(\overline{x}_\varepsilon)$, der ein Element der Paretofront ε-dominiert, maximal $\frac{3N}{2}\frac{\varepsilon}{1+\varepsilon}\frac{1}{\sqrt{2}} = \frac{3N}{4}\frac{\varepsilon}{1+\varepsilon}\sqrt{2}$ von der Paretofront entfernt sein, damit (66) erfüllt ist. Dies genügt allerdings noch nicht, um sagen zu können, dass $\overline{x}_\varepsilon$ mit Sicherheit einen Vektor der Paretofront ε-dominiert. Wie aus Abbildung 10 ersichtlich ist, muss man, wenn man das Count Ones Problem in der ersten Bitstringhälfte den Rändern nach optimiert (d.h. man optimiert die Count Ones Funktion für die ersten $\frac{N}{2}$ Bits eines Bitstrings, dessen zweite Hälfte entweder geschlossen auf 0 oder 1 steht), bis auf $\frac{\varepsilon}{1+\varepsilon}\frac{N}{2}\sqrt{2}$ an die Paretofront herankommen. Wie gesagt ist auch die Schrittweite um $\sqrt{2}$ gestreckt, so kann man als obere Schranke das Count Ones Problem mit dem $(1+1)$-EA mit Approximationsgüte ε lösen (bezüglich der Ordnung des Resultates spielt es dabei keine Rolle, dass der $(1+1)$-EA in der Tat doppelt so schnell ist, da er nicht in jeder zweiten Variation gar nicht auf die relevanten Bits zugreift wie der $(1+1)$-MOEA) und als untere Schranke kann man die erwartete Laufzeit des Count Ones Problems mit Approximationsgüte $\frac{3\varepsilon}{2}$ verwenden. Die Resultate kennen wir aus Kapitel 2. Das Problem ist $O\left(N\log\left(\frac{1+\varepsilon}{\varepsilon}\right)\right)$ und $\Omega\left(N\log\left(\frac{1+\frac{3\varepsilon}{2}}{\frac{3\varepsilon}{2}}\right)\right) = \Omega\left(N\log\left(\frac{\frac{2}{3}+\varepsilon}{\varepsilon}\right)\right)$ Somit formulieren wir den:

Satz 3.4 *Die erwartete Laufzeit des* $(1+1)$*-MOEA zum Auffinden eines Vektors* $\overline{x}_\varepsilon$*, der einen Vektor der Paretomenge* ε*-approximiert ist* $O\left(N\log\left(\frac{1+\varepsilon}{\varepsilon}\right)\right)$ *und* $\Omega\left(N\log\left(\frac{\frac{2}{3}+\varepsilon}{\varepsilon}\right)\right)$

3.3 Zusammenfassung Kapitel 3

Wir rekapitulieren kurz die wichtigsten Erkenntnisse dieses Kapitels:

- Der erwartete Aufwand des $(1+1)$-MOEA für das Auffinden eines paretooptimalen Suchvektors des $LOTZ$ Problems ist $\Theta(N^2)$ im optimierten Fall und $\Theta\left(\frac{N^2}{1+\varepsilon}\right)$ im approximativen Fall.

- Der erwartete Aufwand des $(1+1)$-MOEA für das Auffinden eines paretooptimalen Suchvektors des $COCZ$ Problems ist $\Theta(N \log N)$ im optimierten Fall. Im approximativen Fall ist das Problem $O\left(N \log\left(\frac{1+\varepsilon}{\varepsilon}\right)\right)$ und $\Omega\left(N \log\left(\frac{\frac{2}{3}+\varepsilon}{\varepsilon}\right)\right)$.

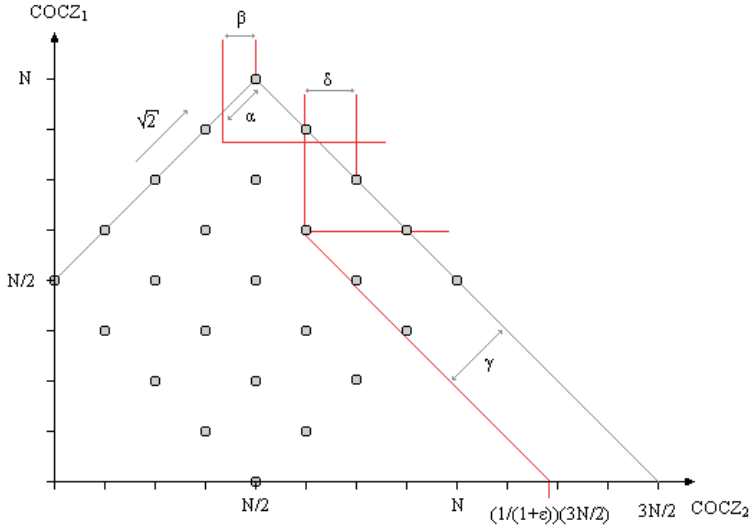

Abbildung 10: Aus der Geometrie des $COCZ$-Zielraumes ist leicht herzuleiten, dass $\alpha = \frac{\varepsilon}{1+\varepsilon}\frac{N}{2}\sqrt{2}$, $\beta = \frac{\varepsilon}{1+\varepsilon}\frac{N}{2}$, $\gamma = \frac{\varepsilon}{1+\varepsilon}\frac{3N}{4}\sqrt{2}$ und $\delta = \frac{\varepsilon}{1+\varepsilon}\frac{3N}{4}$. Die Approximation des $COCZ$ Problems mit dem $(1+1)$-MOEA kann als Approximierung der Count Ones Funktion mit um $\sqrt{2}$ 'gestreckter' Schrittlänge für die Bits in der ersten Bitstringhälfte $(x_1, \ldots, x_{\frac{N}{2}})$ aufgefasst werden. Für die obere Schranke lässt sich dabei das 'gestreckte' $\varepsilon' = \sqrt{2}\varepsilon$ verwenden, was den Effekt der um $\sqrt{2}$ 'gestreckten' Schrittlänge wieder wettmacht, für die untere Schranke lässt sich das noch mehr 'gestreckte' $\varepsilon'' = \frac{3}{2}\sqrt{2}\varepsilon$ verwenden, was den Effekt der um $\sqrt{2}$ 'gestreckten' Schrittlänge mehr als wettmacht.

4 Populationsbasierte EA

4.1 SEMO angewendet auf das $LOTZ$ Problem

Wendet man den SEMO Algorithmus, welchen wir in der Einleitung beschrieben haben, auf das $LOTZ$ Problem an und will man wissen, wieviele Schleifendurchläufe man zu erwarten hat, bis die Paretofront bzw. zu jedem Paretopunkt ein zugehöriger Suchvektor im Archiv ist, so bietet sich die Unterteilung des Optimierablaufes in 2 Phasen an. Die erste Phase dauert solange, bis erstmals ein $\overline{x}^*$ aus der Paretomenge ins Archiv wandert, die zweite Phase dauert solange, bis alle Paretopunkte identifiziert wurden bzw. zu jedem Element der Paretofront ein zugehöriger Suchvektor im Archiv ist.

Eine spezielle Eigenschaft der $LOTZ$ Funktion ist, dass Paare nicht paretooptimaler Suchvektoren, deren Hammingdistanz 1 beträgt, entweder auf denselben Zielvektor abgebildet werden oder aber dass solche Paare aus einem dominierenden und einem dominierten Vektor bestehen. Aus diesem Grund bleibt die Populationsgrösse während Phase 1 immer 1, denn werden $\overline{x}$ und $\overline{x}'$ auf denselben Zielvektor abgebildet, so bleibt $\overline{x}$ im Archiv, ebenso findet $\overline{x}'$ keinen Einlass ins Archiv, falls es von $\overline{x}$ dominiert wird. Einzig wenn $\overline{x}'$ den Vektor $\overline{x}$ dominiert, gelangt $\overline{x}'$ ins Archiv, dafür wird dann aber $\overline{x}$ gelöscht. In [3] wird gezeigt, dass die erwartete Laufzeit von Phase 1 $O(N^2)$ ist. Während Phase 1 verhält sich der SEMO ähnlich wie der $(1+1)$-MOEA. Einziger Unterschied ist, dass der $(1+1)$-MOEA Suchvektoren $\overline{x}$ und $\overline{x}'$, die auf denselben Zielvektor abgebildet werden, im Archiv jeweils vertauscht, während sich der SEMO hier passiv verhält. Dadurch ändert sich aber nichts an der Wahrscheinlichkeit, dass aus dem archivierten Vektor durch die Variation, die in ihrer Art mit derjenigen des SEMO identisch ist, ein dominanter Vektor $\overline{x}'$ hervorgeht. Somit wissen wir, dass Phase 1 $\Theta(N^2)$ ist.

In der zweiten Phase wächst die Population kontinuierlich auf $N+1$ Suchvektoren heran, die durch die Funktion $LOTZ$ jeweils auf die verschiedenen Vektoren der Paretofront abgebildet werden. Das Archiv enthält, während Phase 2 im Gange ist, immer eine Gruppe von Suchvektoren, die auf eine 'zusammenhängende' Gruppe von Zielvektoren abgebildet wird, d.h. wenn im Archiv der Suchvektor $\overline{x}_j$ mit j Leading Ones enthalten ist, $\overline{x}_{j+1}$ mit entsprechend $j+1$ Leading Ones aber nicht, so ist auch $\overline{x}_{j+2}$ sicher nicht im Archiv enthalten. Ein neues Element für das Archiv kann in Phase 2 nur dann gefunden werden, wenn der SEMO bei der zufälligen Auswahl des zu variierenden archivierten Suchvektors entweder denjenigen Vektor mit den meisten Leading Ones oder denjenigen mit den meisten Trailing Zeros erwischt. Die Chance, dass man bei einer Archivgrösse $i < |F^*|$ einen Randvektor auswählt, der noch einen 'freien Nachbarn' hat, ist $\frac{1}{i}$, wenn das Archiv nur einen Vektor enthält oder wenn im Archiv bereits der Vektor $(0,0,0,\ldots,0,0)$ oder $(1,1,1,\ldots,1,1)$ enthalten ist und $\frac{2}{i}$ sonst. Nachdem der SEMO sich einen Vektor aus dem Archiv ausgewählt

hat, dessen Bild im Zielraum am Rand der Paretofront liegt und der noch einen freien Nachbarn hat (einen paretooptimalen Suchvektor mit Hammingdistanz 1, der noch nicht im Archiv drin ist), ist die Chance, dass man genau das richtige Bit mutiert, mit welchem der ausgewählte Vektor in seinen freien Nachbarn übergeht, genau $\frac{1}{N}$, falls mehr als ein Vektor im Archiv enthalten ist oder falls im Archiv entweder der Vektor $(0,0,0,\ldots,0,0)$ oder $(1,1,1,\ldots,1,1)$ enthalten ist und $\frac{2}{N}$ sonst (D.h. wenn nur ein Vektor im Archiv ist ($|P|=1$), der nicht ausschliesslich aus auf 0 oder 1 gesetzten Bits besteht). Somit ist die Chance, dass man bei einem Schleifendurchlauf einen neuen paretooptimalen Vektor erhält, mindestens $\frac{1}{i}\frac{1}{N}$ und höchstens $\frac{1}{i}\frac{2}{N}$ oder eben $\frac{2}{i}\frac{1}{N}$ also $\frac{2}{iN}$. Somit gilt für die Laufzeit der Phase 2:

$$\sum_{k=1}^{N}\frac{Ni}{2} \leq T_2(N) \leq \sum_{k=1}^{N} Ni$$
$$\Rightarrow \frac{N^3 i}{4}+\frac{N^2 i}{4} \leq T_2(N) \leq \frac{N^3 i}{2}+\frac{N^2 i}{2} \tag{67}$$

Die gesammte Optimieraufgabe ist also $\Theta(N^3)$, die erwartete Laufzeit von Phase 1 spielt gegenüber der erwarteten Laufzeit von Phase 2 eine untergeordnete Rolle.

Wir wollen nun wissen, wie lange der SEMO Algorithmus braucht, bis sich in seinem Archiv eine ε-approximative Paretofront befindet, d.h. bis jeder Vektor der Paretofront von mindestens einem Vektor aus dem Archiv P ε-dominiert wird. Wir lassen die Einteilung in 2 Phasen bestehen. Phase 1 bleibt laufzeitmässig bei $\Theta(N^2)$. Wenn nicht N eine gerade Zahl und $\varepsilon = 1$ ist und zudem der SEMO als ersten paretooptimalen Suchvektor denjenigen Vektor mit $\frac{N}{2}$ Leading Ones und ebensovielen Trailing Zeros findet, so sind wir noch nicht fertig nach Phase 1. Wir müssen den SEMO solange weiterlaufen lassen, bis auch die Zielvektoren $(0,N)$ und $(N,0)$ ε-dominiert werden, d.h. die Vektoren $\left(\lfloor\frac{\varepsilon N}{1+\varepsilon}\rfloor, \lceil\frac{N}{1+\varepsilon}\rceil\right)$ bzw. $\left(\frac{\varepsilon N}{1+\varepsilon}, \frac{N}{1+\varepsilon}\right)$ falls $\frac{N}{1+\varepsilon} \in \mathbb{N}$ und $\left(\lceil\frac{N}{1+\varepsilon}\rceil, \lfloor\frac{\varepsilon N}{1+\varepsilon}\rfloor\right)$ bzw. $\left(\frac{N}{1+\varepsilon}, \frac{\varepsilon N}{1+\varepsilon}\right)$ müssen beide im Archiv sein, damit eine ε-approximative Paretofront vorliegt. Weil die Zielraumvektoren im Archiv eine zusammenhängende Gruppe bilden müssen, muss das Archiv mindestens $N+1-2\lfloor\frac{\varepsilon N}{1+\varepsilon}\rfloor$ Suchvektoren enthalten, damit der SEMO eine ε-approximative Paretofront gefunden haben kann. Nach spätestens $N+1-\lfloor\frac{\varepsilon N}{1+\varepsilon}\rfloor$ gefundenen paretooptimalen Suchvektoren liegt im Archiv immer eine ε-approximative Paretofront vor. Siehe dazu auch Abbildung 11.

Wir können für die erwartete Laufzeit der Phase 2 beim ε-approximierten $LOTZ$ Problem behandelt mit dem SEMO schreiben:

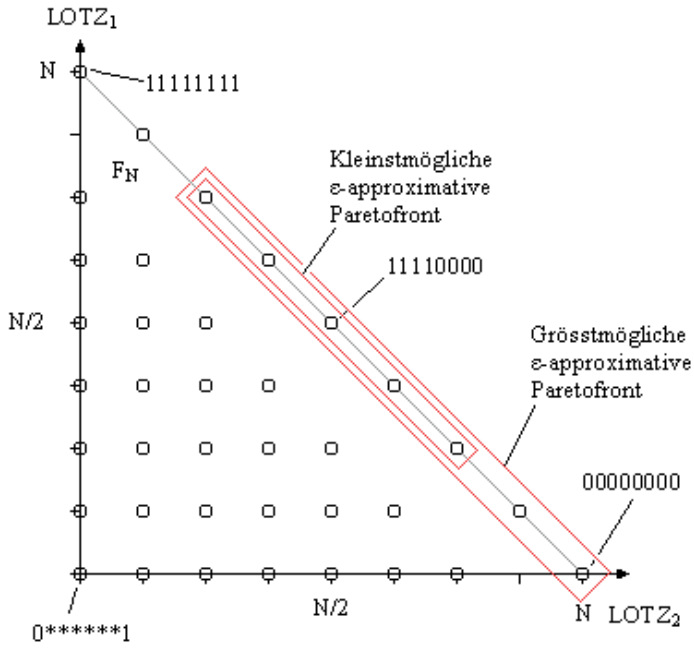

Abbildung 11: Minimale und eine (von 2) maximal möglich grossen ersten ε-approximativen Paretofronten, die der SEMO Algorithmus beim $LOTZ$ Problem findet

$$\begin{aligned} \sum_{k=1}^{N-2\lfloor\frac{\varepsilon N}{1+\varepsilon}\rfloor} \frac{Ni}{2} \leq T_{2,\varepsilon}(N) \leq \sum_{k=1}^{N-\lfloor\frac{\varepsilon N}{1+\varepsilon}\rfloor} Ni \\ \Rightarrow \frac{N\left(N-2\lfloor\frac{\varepsilon N}{1+\varepsilon}\rfloor\right)^2}{4} + \frac{N\left(N-2\lfloor\frac{\varepsilon N}{1+\varepsilon}\rfloor\right)}{4} \leq T_{2,\varepsilon}(N) \\ \leq \frac{N\left(N-\lfloor\frac{\varepsilon N}{1+\varepsilon}\rfloor\right)^2}{2} + \frac{N\left(N-\lfloor\frac{\varepsilon N}{1+\varepsilon}\rfloor\right)}{2} \end{aligned} \tag{68}$$

Verwenden wir in (68) anstatt des Terms $N-2\lfloor\frac{\varepsilon N}{1+\varepsilon}\rfloor$ den Term $N-2\frac{\varepsilon}{1+\varepsilon}N = \frac{1-\varepsilon}{1+\varepsilon}N$, so machen wir einen Fehler, der höchstens von quadratischer Ordnung in N ist, also vernachlässigbar gegenüber den potenteren Termen aus (68) ($\lfloor\frac{\varepsilon N}{1+\varepsilon}\rfloor + 1 > \frac{\varepsilon N}{1+\varepsilon}$ und $(\alpha+1)^3 = \alpha^3 + \underbrace{3\alpha^2 + \cdots}_{Fehler\,quadr.\,Ord.}$). Dasselbe gilt, wenn wir $\lfloor\frac{\varepsilon N}{1+\varepsilon}\rfloor$ durch $\frac{\varepsilon N}{1+\varepsilon}$ ersetzen. Wir formulieren somit den:

Satz 4.1 *Die erwartete Laufzeit des SEMO Algorithmus zum Auffinden einer ε-approximativen Paretomenge des LOTZ Problems ist* $O\left(\frac{N^3}{(1+\varepsilon^2)}\right)$ *und* $\Omega\left(N^3\left(\frac{1-\varepsilon}{1+\varepsilon}\right)^2\right)$.

Würden wir den (1+1)-MOEA in der Art zu einem populationsbasierten EA verändern, dass er immer die nichtdominierten bisher evaluierten Suchvektoren im Speicher behält, so erhielten wir genau dieselben Resultate. Der so adaptierte $(1+1)$-MOEA würde auf der Paretofront einen Random Walk ausführen, bei dem er nur alle $\frac{N}{2}$ Schleifendurchläufe einen Schritt machen würden. Die Random Walks im Freifeld und von 'Wänden' aus sind $\Theta(N^2)$, und beim ε-approximierten Problem sind entsprechend verkürzte Random Walks zu machen. Zum Thema 'Random Walks' siehe [6].

5 Schlusswort, Danksagung

Ich habe durch diese Diplomarbeit am Institut für technische Informatik an der ETH Zürich einen interessanten Einblick in ein aktuelles Forschungsfeld der Theoretischen Informatik erhalten, das wohl noch einige Zeit im Interesse der Informatikgrundlagenforschung bleiben wird. Probleme, die auf den ersten Blick relativ simpel erscheinen, sind bei genauerem Betrachten kaum zu lösen, wie beispielsweise die Bestimmung genauer Erwartungswerte für die Problemlaufzeit populationsbasierter Algorithmen für das $COCZ$ Problem. Es wird spannend sein zu sehen, wie weit sich die Anwendung der Prinzipien der natürlichen Evolution in der Optimierung/Problemlösungsfindung im technischen Bereich durchsetzen können wird.

Mein Dank für die Unterstützung während der ganzen Zeit gilt meinen beiden Betreuern Marco Laumanns und Amela Prelić. Sie hatten immer Zeit für meine Anliegen, auch dann, wenn sie theoretisch keine gehabt hätten. Vielen herzlichen Dank.

Literatur

[1] T. Bäck, D. B. Vogel u. Z. Michalewicz. *Handbook of Evolutionary Computation.* IOP Publishing and Oxford University Press, Bristol, U.K., 1997

[2] F. Kursawe. *Evolution Strategies - Simple 'Models' of Natural Processes?* in: Revue Internationale de Systemique, Volume 7.5, 1993, pages 627-642.

[3] M. Laumanns, L. Thiele, E. Zitzler. *TIK-Report No. 165.* Institut für Technische Informatik u. Informationsnetzwerke, ETH Zürich, April 2003

[4] M. Laumans, L. Thiele, E. Zitzler, K. Deb. *Archiving with Guaranteed Convergence and Diversity in Multi-Ojective Optimization.*, published in *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 439-447, New York, July 2002. Morgan Kaufmann Publishers.

[5] O. Giel und I. Wegener. *Evolutionary Algorithms and the Maximum Matching Problem.* Technical Report ISSN 1433-3325, University of Dortmund, December 2002

[6] R. Motwani u. P. Raghavan. *Randomized Algorithms*, Cambridge University Press, Cambridge UK, 1995