Institut für Technische Informatik
und Kommunikationsnetze
Computer Engineering and Networks Laboratory

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo

Semesterarbeit FS 2008

Datenstrukturen für schnelle Dominanzüberprüfung in Evolutionären Algorithmen

Student: Philip Tschiemer
philipt @ vis.ethz.ch

Betreuer: Johannes Bader
bader @ tik.ee.ethz.ch

Professor: Eckart Zitzler
zitzler @ tik.ee.ethz.ch

Danksagung

Ich würde gerne Johannes Bader für seine Hilfestellungen und insbesondere sein wertvolles Feedback sogar an Wochenenden danken, und auch Eckart Zitzler für die Möglichkeit an einer interessanten und weitgehend selbstständigen Lehrveranstaltung teilzunehmen.

Ausserdem sei Sandra Nicolodi für konstruktiven, und unzähligen VIS Büro Besuchern für weniger konstruktiven Input gedankt.

Inhaltsverzeichnis

1 Einführung

Dominanzüberprüfungen in Evolutionären Algorithmen für die Mehrzieloptimierung sind eine der am häufigsten aufgerufenen Vergleichoperationen. In dieser Semesterarbeit betrachte ich deshalb Datenstrukturen für die schnelle Durchführung von Dominanzüberprüfungen.

1.1 Motivation

Nachfolgend werde ich den thematischen Rahmen einführen, um die Motivation für schnelle Dominanzüberprüfungen weiter auszuführen. Dazu stelle ich erst die Basis von Optimierungsproblemen vor, um dann Evolutionäre Algorithmen als Lösungsansatz von Optimierungsproblemen, insb. Mehrzieloptimierungsproblemen, vorzustellen.

1.1.1 Optimierung

Viele Problemstellungen, wie z.B. die alltägliche Frage nach dem schnellsten Anschluss nach Sibu (Malaysien), besitzen eine praktisch nicht aufzählbare Anzahl von Möglichkeiten, von welchen aber nur wenige einer optimalen Lösung entsprechen. Es kann von einem Optimierungsproblem gesprochen werden. Diese lassen sich prinzipiell reduzieren auf die Aufgabe 'für $d \in D$ minimiere (maximiere) f(x)', oder werden formal ausgeschrieben wie folgt:

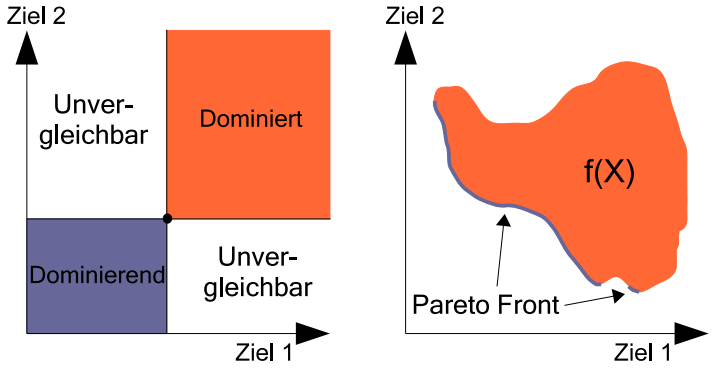
Gegeben:	D	(Entscheidungsraum)
	$f_i : D \to \mathbb{R} \quad (i = 1..M)$	(Zielfunktionen)
	$g_i : D \to \mathbb{R}, g_i(d) = \quad (i = 1..c)$	(Einschränkungen)
	$\text{opt} \in \{min, max\}$	(Optimierungsziel)
Gesucht:	$\text{arg}_{d \in D}\ \text{opt}_{i=1..M}(\ f_i(d)\ )$	

Es können mehr als eine Lösung existieren.

Gilt $M = 1$, so existiert mindestens ein Optimum. Wenn hingegen $M > 1$, was sich in unserem Beispiel als zusätzliches Ziel (die Reise soll möglichst viel kosten) modellieren lässt, so ist die Existenz eines Optimum nach obiger Definition nicht mehr garantiert. In der sogenannten Mehrzieloptimierung wird oft das Konzept der Pareto Dominanz verwendet, welches für ein Minimierungsproblem wie folgt definiert ist:

$$
\begin{array}{rcll}
 & & x, y \in D & \\
x \prec y & \leftrightarrow & \forall i \in \{1..M\}: \quad f_i(x) < f_i(y) & \text{(x dominiert y strikt)} \\
x \preceq y & \leftrightarrow & \forall i \in \{1..M\}: \quad f_i(x) \leq f_i(y) & \text{(x dominiert y schwach)} \\
x \parallel y & \leftrightarrow & \exists i, j \in \{1..M\}: & \text{(x, y unvergleichbar)} \\
 & & f_i(x) < f_i(y) \wedge f_j(x) > f_j(y) &
\end{array}
$$

Optima von Mehrzieloptimierungsproblemen bestehen oft aus einer Menge von unvergleichbaren Lösungen, man spricht auch von einer nicht dominierten Front, bzw. Pareto Front (siehe Abb. 1b). Unvergleichbare Lösungen modellieren einen Trade-off zwischen den verschiedenen Zielen. Oft ist ein Überblick über die verschiedenen Lösungsvarianten von Interesse.



(a) (b)

Abbildung 1: Optimierung nach Minima in zwei Dimensionen. (a) Einteilung des Zielbereichs nach Dominanzkonzept. (b) Pareto Front einer Lösungsmenge.

Deterministische Optimierer eignen sich i.d.R nur für sehr spezifische Probleme, da diese an die Struktur des Problems adaptiert werden. Ist die Struktur nur schwer fassbar, in unserem Beispiel ausdrückbar durch die zusätzliche Forderung pittoreske Landschaften zu durchreisen, werden oft randomisierte Optimierer, wie *Local Search* oder *Simulated Annealing* verwendet.

Diese merken sich mindestens eine Lösung und versuchen durch gezielt-randomisiertes Sampling optimalere Lösungen zu entdecken; oft wird ein Nachbarschaftskonzept verwendet, d.h. Stichproben werden lediglich in einer gewissen semantischen Nähe der aktuell optimalsten Lösungen genommen.

Eine für die Mehrzieloptimierung verwendete Klasse von randomisierten Optimierern sind Evolutionäre Algorithmen.

1.1.2 Evolutionäre Algorithmen

Evolutionäre Algorithmen (EA) operieren i.d.R nicht auf einzelnen Lösungen, sondern führen Mengen von Lösungen. EA versuchen die Gesamtqualität dieser Menge zu verbessern, was insb. für Mehrzieloptimierungsprobleme interessant ist; nicht-dominierte (Trade-off) Lösungen werden dominierten Lösungen vorgezogen, was in einer Suche nach der nicht-dominierten Front resultiert.

Konkret generieren EA wiederholt eine Menge von neuen potenziellen Lösungen, und von allen Lösungen werden dann durch Auswahlverfahren die gesamthaft Besten ermittelt.

EA sind inspiriert von der Evolutionstheorie, so dass auch im Zusammenhang mit EA ein spezifisches Vokabular verwendet wird: Lösungen werden zu Individuen, Mengen von Individuen zu Populationen, die Zielfunktion zur Fitnessfunktion, Variationsoperatoren sind Mutation und Kreuzung (mit zwei oder mehr Eltern). Der prinzipielle Ablauf von EA kann entsprechend als wiederholter Generationswechsel beschrieben werden: eine Population generiert Nachkommen, von welchen die Stärksten überleben[1]. Folgend der Ablauf als Algorithmus:

Algorithmus Schema eines Evolutionären Algorithmus

$\mu \leftarrow$ *Initiale Population*
while *Stop-Kriterium* **do**
  $\lambda \leftarrow$ *Variiere*(μ) {Generiere neue Population ausgehend von Bisheriger (Mutiere/Kreuze Individuen) }
  $\mu \leftarrow$ *WähleFitteste*(μ, λ) {Wähle gesamthaft betrachtet stärkste Individuen}
end while
return μ

Viele EA für die Mehrzieloptimierung verwenden das Konzept der Pareto Dominanz zur Berechnung der überlebenden Individuen (typische Fragen sind z.B. wird das Individuum dominiert, von wievielen?), so dass die Dominanzüberprüfung (u.a. Feststellung der Dominanzrelation zwischen zwei Individuen) eine der meist-aufgerufenen Operationen ist. Es ist also von Interesse über gute Datenstrukturen zur Verringerung der Laufzeit von Dominanzüberprüfungen zu verfügen.

[1] Natürliche Selektion, bzw. *Environmental Selection*.

1.2 Aufgabenstellung

Ziel dieser Arbeit ist die Zusammen- und Gegenüberstellung bestehender Datenstrukturen für die Dominanzüberprüfung, die Einschätzung deren Zweckmässigkeit in Bezug auf die unterschiedlichen Anwendungsfälle mittels theoretischer Betrachtungen und Experimenten, und das Ausarbeiten von neuen Ansätzen.

1.3 Übersicht des Berichts

Der zweite Teil stellt Hintergrundwissen vor. Es wird die verwendete Notation eingeführt, bestehende Datenstrukturen prinzipiell vorgestellt, und die verschiedenen Anwendungsfälle (i.e. verschiedene Typen von Dominanzüberprüfungen) zusammengefasst. Desweiteren werden Aspekte dreier Evolutionärer Algorithmen etwas genauer betrachtet.

Im dritten Teil wird eine eigene Datenstrukturen vorgestellt, welche bisher weniger beachtete geometrische Eigenschaften verwendet.

Im vierten Teil werden die Datenstrukturen verglichen. Es werden asymptotischen Laufzeiten der Datenstrukturen im Kontext der Anwendungsfälle betrachtet, und qualitative Aussagen über einzelne Datenstrukturen werden gemacht. Zudem wurden vier ausgewählte Datenstrukturen implementiert, und deren Effizienz bzgl. Dominanzüberprüfungen getestet; die Ergebnisse werden vorgestellt.

2 Hintergrund

In diesem Teil des Berichts werden zunächst die verwendete Notation und Annahmen eingeführt. Darauf folgt eine Zusammenfassung von Typen von Dominanzüberprüfungen; im Speziellen werden zwei spezifische Evolutionäre Algorithmen genauer erläutert. Abschliessend werden Datenstrukturen vorgestellt, welche für Dominanzüberprüfungen eingesetzt, oder im Zusammenhang mit Dominanzüberprüfungen erwähnt werden.

2.1 Annahmen und Notation

Im Weiteren wird angenommen, dass das betrachtete Optimierungsproblem jeweils minimiert werden soll.

Da die Dominanzüberprüfung eine Operation auf Punktmengen im Hyperraum ist, wird z.T. von Punkten anstatt von Zielfunktionswerten eines Individuums gesprochen. Jedes Individuum wird durch einen Punkt repräsentiert; die Terme sind in der Folge austauschbar. So auch Population und Punktmenge.

$x, y \in \mathbb{R}^M, d \in \{1..M\}:$ $x \prec_d y$**.** x_d dominiert y_d in Dimension d.

$x, y \in \mathbb{R}^M:$ $x \prec y \quad \leftrightarrow \quad \forall d \in \{1..M\} : x \prec_d y$**.** x dominiert y strikt.

$x, y \in \mathbb{R}^M:$ $x \parallel y \quad \leftrightarrow \quad \neg(x \prec y \vee y \prec x)$**.** x und y sind unvergleichbar.

$\in_{u.a.r.}$ *uniformly at random in*, in extenso $x \in_{u.a.r.} X$ bezeichnet das zufällige (gleichverteilte) Ziehen eines Elements aus der Menge X.

$rg(x)$ Rang von x bzgl. einer durch die Relation $\triangleleft$ total geordneten, nicht-leeren, endlichen Menge X. Es gilt:

$$i \in \{1..|X|\} : rg(x) = i \leftrightarrow i = |\{x' \in X : x' \triangleleft x\}| + 1$$

Die Ordnungsrelation und Bezugsmenge sind kontextabhängig implizit gegeben.

$diag_e = \{\frac{1}{\sqrt{M}}\}^M$ Vektor mit identischer Komponente in allen Dimensionen. Es gilt insbesondere $||diag_e||_2 = 1$ (Einheitsvektor).

$x \in \mathbb{R}^M : diag_x = diag_e \cdot < diag_e, x >$**.**

$\mathbb{D} = \{(t \to o + diag_e \cdot t) \in (\mathbb{R} \to \mathbb{R}^M) : o \in \mathbb{R}^M, diag_o = 0\}$**.** Funktionenraum aller parallel zum Vektor $diag_e$ verlaufenden Geraden; sind gegeben durch einen Offset $o \in \mathbb{R}^M$ welcher in der zu $diag_e$ orthogonalen Hyperebene (durch

den Nullpunkt) liegt. Im Besonderen gilt $D(0) = o$. Zudem sei $D_0 = t \rightarrow diag_e \cdot t$ (Gerade ausgehend vom Ursprung). $D \in \mathbb{D}$ werden im Weiteren als Diagonalen bezeichnet.

$x \in \mathbb{R}^M, D \in \mathbb{D} : dist_x^D = dist_x = ||(x - diag_x) - D(0)||_2$**.** Abstand eines Punktes zu einer diagonalen Geraden D. Wenn diese nicht gegeben ist, so ist $D = D_0$ anzunehmen.

2.2 Typen von Dominanzüberprüfungen

Dominanzüberprüfungen in Evolutionären Algorithmen (EA) kommen lediglich in wenigen, z.T. speziellen Varianten vor. Die drei grundlegenden Typen, zwischen denen ich unterscheide, betrachten unterschiedliche Aspekte der Dominanzrelation eines Abfragepunktes bzgl. einer Abfragemenge - wie sie auch in der Fitnessberechnung von EA für die Mehrzieloptimierung vorkommen. Es folgen die drei Typen.

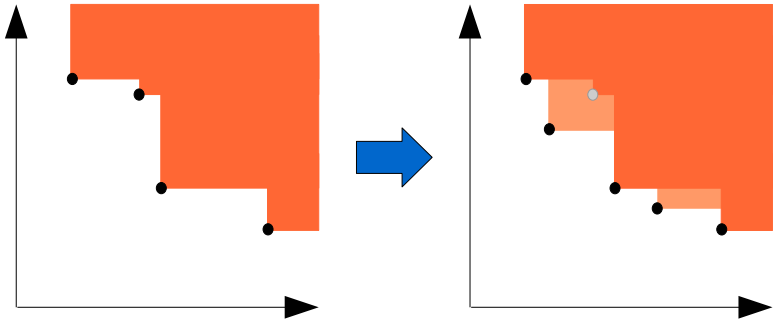
Dominanztest

Für den Abfragepunkt q soll festgestellt werden, ob es in einer Punktmenge P einen q dominierenden Punkt gibt, i.e. ob $\exists p \in P : p \prec q$. Es gibt zudem zwei typische Szenarien in denen Dominanztests vorkommen: Fortschreitende Front, und Frontentdeckung.

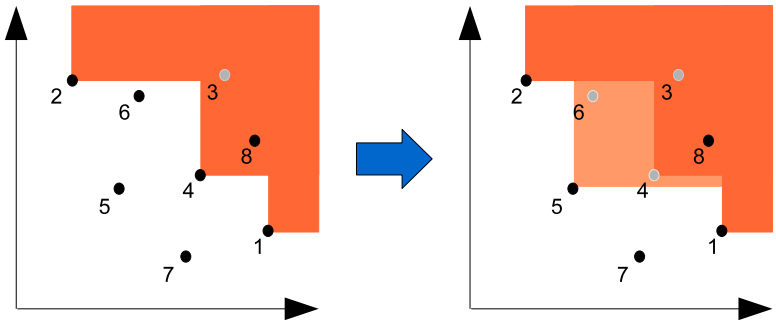
Fortschreitende Front

Die Bezugspopulation existiert in Form eines Archivs, welches alle bisher optimalsten (nicht-dominierten) Lösungen enthält. Gegeben ist eine Punktmenge, deren Elemente auf Dominanz getestet werden sollen; ist ein Punkt q nicht-dominiert, so sollen alle von q dominierten Punkte des Archivs entfernt, und q zum Archiv hinzugefügt werden. Das von den Punkten des Archivs dominierte Hypervolumen verbessert sich in der Regel nur. Siehe auch Abb. 2.
Vorkommen: Pareto Archived Evolution Stategy[a](PAES), Distance-based Pareto Genetic Algorithm[b] (DPGA), Strength Pareto Evolutionary Algorithm 1/2[6] (SPEA).

[a] [2], Seiten 272-279.
[b] [2], Seiten 253-261.



(a)



(b)

Abbildung 2: Die Zahlen beschreiben die Reihenfolge, in welcher die Individuen hinzugefügt werden, graue Punkte entsprechen verworfenen Individuen. (a) Beispiel einer fortschreitenden Front; mit 2 neuen nicht-dominierten Individuen. (b) Beispiel einer Front im Aufbau (Frontentdeckung); nach 2 Schritten.

Frontentdeckung

Eine beliebige Punktmenge P ist gegeben. Gesucht ist die Menge der nicht-dominierten Punkte von P. Frontentdeckung wird oft in einer Weise ähnlich den Dominanztests mit fortschreitender Front durchgeführt (z.B. NSGA). Siehe Abb. 2b. Vorkommen: Non-dominated Sorting Genetic Algorithm I/II[5] (NSGA), Thermodynamic Genetic Algorithm[a] (TDGA).

[a] [2], Seiten 270-272.

Das erste, respektive zweite Szenario wird in der Literatur *Dynamic*, respektive *Static Non-dominance Problem* genannt [20].

Zählen dominierter Punkte

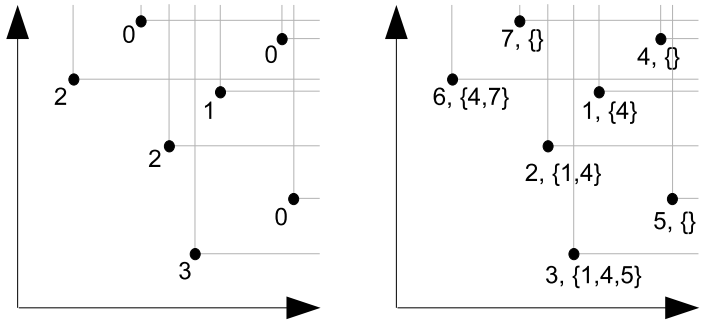
Für den Abfragepunkt q, und Bezugsmenge P ist die Grösse der Menge $\{p \in P : q \prec p\}$ gefragt. Da die Operation symmetrisch ist, mit dem Zählen dominierender Punkte, wird nicht weiter auf letztere eingegangen. Die Zahl dominierter Punkte fliesst typischerweise direkt in die Fitnessberechnung ein. Siehe Abb. 3.

Vorkommen: Multiobjective Genetic Algorithm[2](MOGA), Niched Pareto Genetic Algorithm [3](NPGA), SPEA 1/2.

Aufzählen dominierter Punkte

Für den Abfragepunkt q, und Bezugsmenge P ist die dominierte Menge $\{p \in P : q \prec p\}$ gesucht. Die Operation ist wiederum symmetrisch zum Aufählen der dominierenden Punkte. Typischerweise fliesst eine Eigenschaft der dominierten Punkte in die Fitnessberechnung mit ein. Siehe Abb. 3b.

Vorkommen: SPEA 1/2.



(a) Dominiertes Zählen (b) Dominiertes Aufzählen

Abbildung 3: Was jeweils interessiert ist (a) die Anzahl dominierter Individuen. (b) die Menge aller dominierter Individuen.

[2][2], Seiten 200-209.
[3][2], Seiten 218-223.

Die von mir betrachtete Literatur behandelt prominent die beiden *Non-dominance Problems*.

Es folgen wenige, spezifische Algorithmen, wie sie in Evolutionären Algorithmen vorkommen; insb. ein Monte Carlo Verfahren, welches nicht in das obige Schema passt.

2.3 Spezifische Algorithmen mit Dominanzüberprüfungen

Folgend werden Teile von Evolutionären Algorithmen beschrieben, welche Dominanzüberprüfungen einsetzen. Diese Teile verkörpern exemplarische Anwendungen von Dominanzüberprüfungen, welche sehr eigene Anforderungen an die Datenstrukturen stellen.

2.3.1 Nicht-dominiertes Sortieren

Nicht-dominiertes Sortieren einer Punktmenge P resultiert in einer Sequenz $(P)_{i\in\mathbb{N}}$ von k in sich nicht-dominierten Punktmengen, so dass gilt

$$\forall i \in \{1..k-1\} : (\exists p \in P_i, q \in P_{i+1} : p \prec q \wedge \neg\exists p \in P_i, q \in P_{i+1} : q \prec p)$$

i.e. ist es eine Sortierung in nicht-dominierte Fronten.

Die Umsetzung sieht i.d.R. so aus, dass für eine gegebene Punktmenge rekursiv die nicht-dominierte Front bestimmt wird. Die Frontentdeckung stellt den bzgl. Dominanzüberprüfungen wesentlichen Teil dar.

In 4.2.2 wird der grundlegende Algorithmus wie in [5] beschrieben, für die Unterstützung von Datenstrukturen für die Dominanzüberprüfung geringfügig angepasst.

2.3.2 SPEA2

Zu Beginn jeder Generation von SPEA2 sind zwei Punktmengen vorhanden: ein Elitearchiv nicht-dominierter Punkte E, und eine neu generierte Nachkommenschaft P. Aus beiden Mengen wird zuerst die gesamthaft nicht-dominierte Menge E' sowie $P' = (E \cup P) \setminus E'$ (alle anderen Punkte) ermittelt. Wenn E' eine vorgegebene Grösse G überschreitet, wird E' auf G Punkte reduziert (siehe [6]). Darnach wird für alle Individuen p ihre Grundfitness $\rho(p)$ berechnet, dies geschieht in zwei Schritten:

1. Für alle Punkte wird die Anzahl dominierter Punkte ermittelt. Sei für $p \in E' \cup P' : \delta(p) = |\{x \in P' : p \prec x\}|$ die Anzahl von p dominierter Punkte.

2. Die Grundfitness $\rho(p)$ entspricht dann der Summe der Anzahl dominierter Punkte aller dominierender Punkte, i.e. $p \in E' \cup P' : \rho(p) = \sum_{q \in E' \cup P', q \prec p} \delta(q)$ (es gilt $\forall e \in E' : \delta(e) = 0$).

Die schlussendliche Fitness errechnet sich aus weiteren Faktoren, welche hier vernachlässigt werden.

In 4.2.3 werde ich eine Restrukturierung des oben beschriebenen Vorgangs vorschlagen.

2.3.3 Fast Hypervolume Based Search by Monte Carlo Sampling

Ein jüngerer und vielversprechender Ansatz[8, 7] für die Natürliche Selektion beruht auf dem Berechnen des von jedem Individuum beigetragenen Hypervolumens, i.e. des Hypervolumens, welches nur von dem betroffenen Individuum dominiert wird (siehe Abb. 4a).

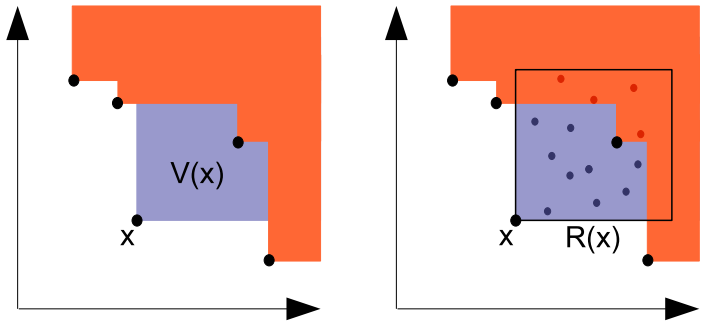
Das Hypervolumen in vielen Dimensionen deterministisch zu berechnen ist eine aufwendige Angelegenheit[4]. J. Bader, K. Deb, und E. Zitzler schlagen eine neue, stochastische Methode[3] vor, welche nach ersten Experimenten gute Resultate auch in höheren Dimensionen liefert. Das Monte Carlo Verfahren führt eine enorme Anzahl von Dominanzüberprüfungen durch. Ein genauer Algorithmus bzgl. Dominanzüberprüfungen wurde in [3] nicht vorgestellt, weshalb Varianten in Abschnitt 4.2.4 beschrieben werden.

Das Berechnen des Hypervolumens funktioniert konzeptuell wie folgt: für ein Individuum x wird eine Samplingbox $R(x)$ mit Volumen S_x ermittelt, welche x als minimalsten Punkt besitzt und das gesamte nur von x dominierte Hypervolumen umfasst; das Hypervolumen eines dominierten Individuums ist gleich Null. Sei $N_s \in \mathbb{N}$ (N_s sehr gross) die Anzahl der uniform aus $R(x)$ gezogenen Stichproben und $N_x \in \{0..N_s\}$ die Anzahl nur von x dominierter Stichproben. Das effektive Hypervolumen kann dann approximiert werden durch $S_x \cdot \frac{N_x}{N_s}$. Siehe Abb. 4b.

2.4 Datenstrukturen

In den nächsten Abschnitten werden Datenstrukturen vorgestellt, welche für Dominanzüberprüfungen eingesetzt, oder im Zusammenhang mit Dominanzüberprüfungen erwähnt werden. Nebst Listen, finden sich darunter hauptsächlich geometrische Suchstrukturen, oder auf Dominanztests ausgelegte Datenstrukturen.

[4] Bisherige Algorithmen haben Laufzeiten exponentiell in der Anzahl Dimensionen[4].



(a) (b)

Abbildung 4: Form des Hypervolumens eines Individuums x (bläulich). (a) Effektives Volumen $V(x)$. (b) Beispiel-Samplebox $R(x)$. $V(x)$ wird approximiert durch $S_x \cdot \frac{N_x}{N_s}$ (S_z := Volumen $R(x)$, N_x := Anzahl bläuliche Punkte, N_s := Anzahl bläuliche + rötliche Punkte).

Auf Datenstrukturen, welche für alle Dominanzüberprüfungen einsetzbar sind, sollten folgende Operationen ausführbar sein:

Einfügen	Einfügen eines neuen Punktes, mit Koordinaten und Index.
Entfernen	Löschen eines Punktes aus der Struktur, gegeben einen Index.
Aufbau mit N	Aufbau der Struktur von Grund auf mit N gegebenen Punkten, mit Koordinaten und Index.
Optimieren	Optimierung der ggf. degenerierten Struktur.
Dominanztest	Durchführung eines Dominanztests gegeben einen Referenzpunkt.
Zählen	Zählen aller dominierter Punkte gegeben einen Referenzpunkt.
Aufzählen	Aufzählung aller dominierter Punkte gegeben einen Referenzpunkt.

Es folgt die Vorstellung der Datenstrukturen.

2.4.1 Listen

Listen, respektive Arrays implementieren grundsätzlich Mengen, bei welcher die Einfügesequenz eine interne Traversierungsordnung festlegt. Bei Dominanzüberprüfungen wird durch die Menge der Elemente iteriert, bis die ent-

sprechende Frage beantwortet werden kann.

Listen sind eine grundlegende Standard-Datenstruktur, welche die Einsatzmöglichkeiten nicht weiter einschränken.

2.4.2 Kd-Trees

Kd-Trees sind auf beliebig viele Dimensionen erweiterte eindimensionale, binäre Suchbäume. Jedem Knoten wird nebst Koordinatenvektor eine diskriminierende Dimension $k \in 1..M$ (M = Anzahl Dimensionen) zugewiesen; bei einer Suche wird an jedem Knoten anhand der k-ten Komponente der Koordinaten von Knoten und Abfragepunkt das weitere Verhalten (analog zu eindimensionalen Suchbäumen) bestimmt. Es ergibt sich eine Partitionierung des Suchraumes in Hyperrechtecke. Siehe Abb. 5.

In traditionellen Kd-Trees wird einem Knoten der Tiefe t die diskriminierende Dimension $k = t \mod M$ zugewiesen, d.h. auf dem Suchpfad zu einem Knoten werden (ggf. wiederholt) aufsteigende Diskriminatoren angetroffen. Die Tiefe des Baumes hängt wie bei der eindimensionalen Variante von der Einfügesequenz ab.

Eine randomisierte Variante[14] macht das erwartete Laufzeitverhalten bzgl. Abfrageoperationen unabhängig von der Einfügesequenz.

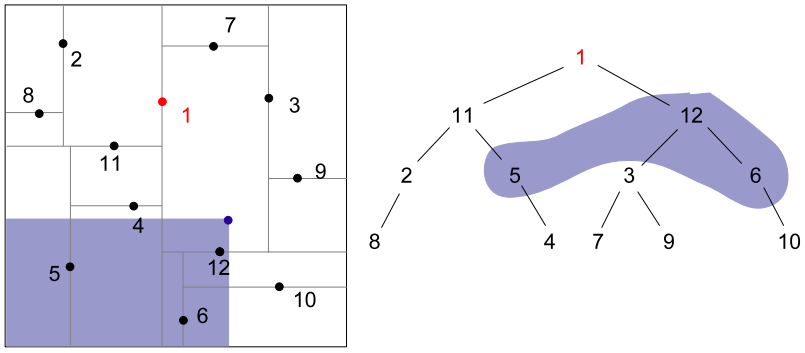


Abbildung 5: Beispiel eines zweidimensionalen Kd-Tree, und Dominanztest. Die Wurzel ist rot, der Testpunkt blau. Es sind alle Punkte gesucht, welche in der bläulichen Region liegen.

Dominanzüberprüfungen werden zu Bereichsabfragen.

Kd-Trees sind grundsätzlich für alle Dominanzüberprüfungstypen geeignet.

2.4.3 K-Range Trees

Ein eindimensionaler Range Tree, respektive ein 1-Range Tree, entspricht einem normalen binären Suchbaum mit geordneter Elementeverkettung. Ein 2-Range Tree besteht aus einem 1-Range Tree bzgl. der ersten Komponente und im jedem Knoten wurzelt ein zusätzlicher 1-Range Tree, der alle Kinderknoten bzgl. der zweiten Komponente enthält. Ein K-Range Tree höherer Ordnung ist analog iterativ aufgebaut. Siehe Abb. 6.

Die Einfügesequenz bestimmt die Tiefe des jeweiligen Range Trees.

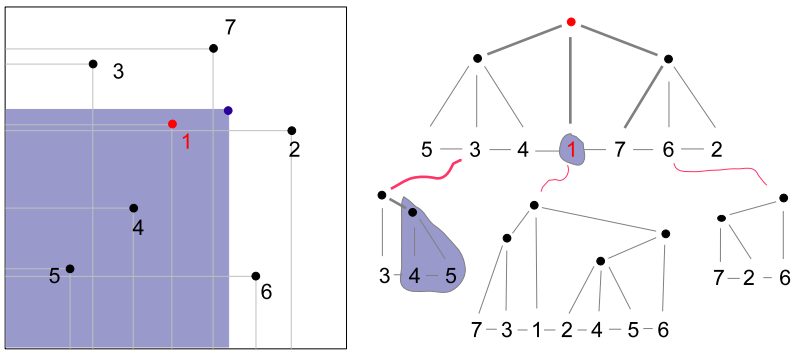


Abbildung 6: Beispiel eines zweidimensionalen k-Range Trees mit Bereichsabfrage. Der Wurzelknoten ist rot. Dicke Links zwischen den Knoten zeigen den Verlauf der Abfrage auf.

K-Range Trees sind ausgelegt für Bereichsabfragen; Dominanzüberprüfungen werden prinzipiell wie bei Kd-Trees durchgeführt.

K-Range Trees sind uneingeschränkt einsetzbar.

2.4.4 Dominated und Non-Dominated Trees

Dominated Trees konstruieren aus bis zu M Koordinatenvektoren sogenannte zusammengesetzte Punkte (*Composite Points*), so dass mindestens einer der konstituierenden Punkte (*Constituent Points*) vom zusammengesetzten Punkt dominiert wird. Die Menge der zusammengesetzten Punkte selber wird so konstruiert, dass diese über der Dominanzrelation eine total geordnete Menge ergibt.

Bei Dominanztests wird sodann erst der kleinste nicht-dominierende (1), sowie der grösste dominierte (2) zusammengesetzte Punkt ermittelt. Ist (1) nicht der kleinste Punkt, so ist der Abfragepunkt dominiert; alle konstituierenden Punkte zu (2) oder grösseren Punkten gehörend, können nicht dominieren; es kann nichts ausgesagt werden über alle konstituierenden Punkte,

welche die zusammengesetzten Punkte von (1) bis exklusiv (2) erzeugen, d.h. sie müssen einzeln überprüft werden. Siehe Abb. 7, links.

Einfügen und Entfernen von einzelnen Punkten führt zu Degeneration der Datenstruktur: weniger als M Individuen konstituieren einen zusammengesetzten Punkt.

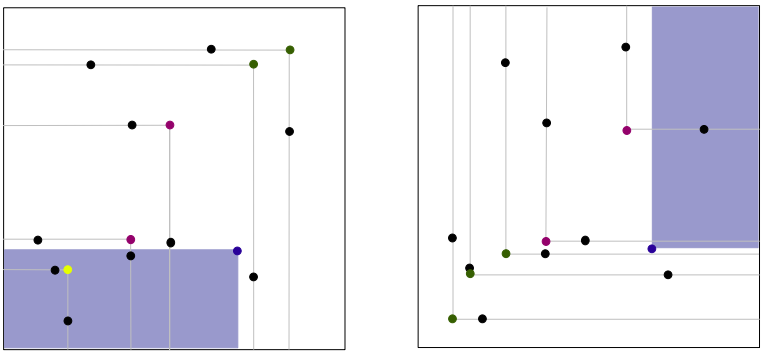


Abbildung 7: Beispiel eines Dominated Tree (links), und eines Non-Dominated Tree (rechts) mit Abfragen. Schwarze Punkte entsprechen Individuen, der Abfragepunkt ist blau, die restlichen Punkte sind zusammengesetzte Punkte (grün : ausgeschlossen, gelb: Dominanz mindestens einmal erfüllt, magenta: keine Aussage möglich.

Dominated Trees werden lediglich zur Testen auf Dominanz verwendet; zur (Auf-)Zählung von dominierten Punkten werden die symmetrischen Non-Dominated Trees verwendet. Siehe Abb. 7, rechts.

2.4.5 Quad-Trees

Knoten eines Quad-Trees unterteilen den verbleibenden Hyperraum in 2^M Teilräume. Koordinatenvektor von Knoten und Abfragepunkt werden komponentenweise verglichen, und das Verhältnis in einem Bitvektor der Länge M gespeichert (ein gesetztes Bit bedeutet, dass der Abfragepunkt in der entsprechenden Komponente den Knoten dominiert). Sind weder alle Bits gesetzt (Abfragepunkt dominiert Knoten), noch keine Bits (Knoten dominiert Abfragepunkt) so wird die Suche auf dem Kind mit identischem Vergleichsbitvektor weitergeführt. Quadtrees sind also 2^M-äre Suchbbäume. Siehe Abb. 8.

Einfüge- und Entfernsequenzen beeinflussen die optimalst logarithmische Tiefe des Suchbaumes.

Quad-Trees werden nur zum Testen bzgl. nicht-dominierte Fronten eingesetzt.

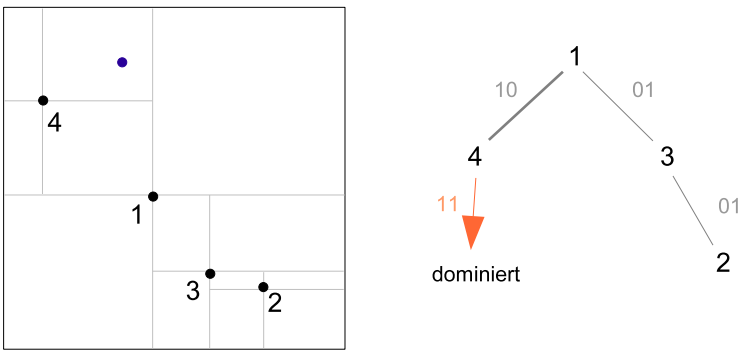


Abbildung 8: Beispiel einer nicht-dominierten Front, und Dominanztest. Der Abfragepunkt ist blau, der Abfragepfad im Quadtree ist verstärkt.

2.4.6 Dominance Decision Trees

Die von O. Schütze[20] für Dominanztests adaptierten Dominance Decision Trees (DDT) sind M-äre Suchbäume. Für das i-te Kind eines Knoten gilt, dass der Knoten das Kind bzgl. der i-ten Komponente dominiert, und bzgl. keiner vorhergehenden Komponente. Siehe Abb. 9.

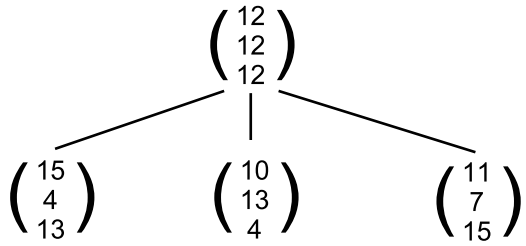


Abbildung 9: Beispiel eines Dominance Decision Trees.

DDTs sind für die Durchführung von Dominanztests gedacht.

2.4.7 Binary Decision Diagrams

Für den spezifischen Fall des Dominanztests einer fortschreitenden Front, respektive des *Dynamic Non-Dominance Problem*, stellten M. Lukasiewycz et al.[21] eine Adaption von *Binary Decision Diagrams* (BDD) vor.

BDD implementieren lediglich Bool'sche Funktionen. M. Lukasiewycz et al. schlugen vor eine dimensionsweise Abbildung des Zielraumes auf ganze Zahlen in Zweierbasis mit definierter Länge zu verwenden, so dass die Abbildung die Dominanzrelation respektiert. Die neuen Koordinaten eines Indivi-

duums werden in eine Bool'sche Funktion (BDD) übersetzt, so dass aus einer Belegung der freien Variablen mit den Koordinaten eines Abfragepunktes eine binäre Antwort auf die Dominanzfrage resultiert. Durch Verknüpfung mittels Disjunktionen kann der gesamte von einer Population dominierte Hyperraum durch Auswertung einer Bool'schen Funktion auf Nicht-Dominanz getestet werden.

Bei der Vereinigung und kostspieligen Vereinfachung (siehe [23]) von BDDs gehen die impliziten Informationen bzgl. dargestelltem Individuum verloren; die Entfernung der einzelnen Teile ist nicht mehr möglich. Hieraus ergeben sich die Einsatzeinschränkungen auf Dominanztests mit fortschreitender Front.

2.4.8 Komplexitäten Speicher und Laufzeit

Es folgt eine Zusammenstellung der Speicher- und Laufzeitkomplexitäten der vorgestellten Datenstrukturen.

Diese Angaben werden später in 4.3.1 wiederverwendet.

Datenstruktur	Speicher	Einfügen	Entfernen
Listen	$O(MN)$	$O(M)$	$O(1)$
Kd-Tree$_{optimal}$	$O(MN)$	$O(M + logN)$	$O(logN)$
Randomized Kd-Tree	$E[MN]$	$E[M + logN]$	$E[logN]$
K-Range Tree	$O(MN + (logN)^{M-1})$	nv	nv
Dominated und Non-Dominated Trees	$O(MN + Mlog(N/M))$	$O(M + Mlog(N/M))$	$O(1)$
Quad-Tree$_{optimal}$	$O(MN)$	$O(M + Mlog_M N)$	$O(M^2 log_M N)$
Dominance Decision Trees	$O(MN)$	nv	nv
Binary Decision Diagrams[1]	$O_{exp}(logN)$	$O_{exp}(N)$	-

Datenstruktur	Aufbau mit N	Optimieren
Listen	$O(MN)$	-
Kd-Tree$_{optimal}$	$O(MN + NlogN)$	$O(NlogN)$
Randomized Kd-Tree	$E[MN + NlogN]$	-
K-Range Tree	$O(MN + N(logN)^{M-1})$	$O(N(logN)^{M-1})$
Dominated und Non-Dominated Trees	$O(MN + MNlogN)$	$O(MN + MNlogN)$
Quad-Tree$_{optimal}$	nv	nv
Dominance Decision Trees	nv	nv
Binary Decision Diagrams	-	$O(MN^2)$

Datenstruktur	Dominanztest	Zählen	Aufzählen[2]
Listen	$O(MN)$	$O(MN)$	$O(MN + k)$
Kd-Tree$_{optimal}$	$O(MN^{1-1/M})$	$O(MN^{1-1/M})$	$O(MN^{1-1/M} + k)$
Randomized Kd-Tree[3]	$E_<$	$E_<$	$E_< + k$
K-Range Tree	$O((logN)^M)$	$O((logN)^M)$	$O((logN)^M + k)$
Dominated und Non-Dominated Trees[4]	$O(Mlog(N/M) + tM)$	$O(Mlog(N/M) + tM)$	$O(Mlog(N/M)+tM+k)$
Quad-Tree$_{optimal}$	$O(Mlog_M N)$	-	-
Dominance Decision Trees	nv	nv	nv
Binary Decision Diagrams	$O(M)$	-	-

Tabelle 1: Komplexität Datenstrukturen bzgl. Speicher, Wartungsoperationen und Dominanzüberprüfungen. N = Grösse Punktmenge, M = Dimension, nv = nicht verfügbar, - = Operation nicht vorgesehen.

[1] Nach den experimentellen Ergebnissen von [21].
[2] k gleich der Anzahl dominierter Punkte.
[3] Obere Limite für einen zufälligen Kd-Tree mit einem zufälligen Abfragebereich innerhalb eines uniform gesampelten Hyperrechtecks (siehe auch [15]), es gilt:
$E_< = logN + N + \sum_{i=1}^{M-1} \binom{M}{i} \cdot N^{1.07-i/M}$
[4] $0 \leq t \leq N$, t die Anzahl ggf. separat zu betrachtender Punkte.

3 Eigene Datenstruktur

Mein eigener Ansatz für eine neue Datenstruktur beruht auf der Idee mit zusätzlichen geometrischen Eigenschaften der Punkte schnell Entscheidungen (z.B. bzgl. des Ausschlusses) treffen zu können. Werden hierzu eine geringe und konstante Anzahl k von Kriterien herangezogen, so kann ein Dominanztest zweier Punkte bestenfalls in $O(k)$, schlimmstenfalls in $O(M+k)$ (Kriterien erlauben keine Entscheidung, normaler Dominanztest ist nötig) durchgeführt werden, wobei M die Dimension ist.

Würden die Punkte zudem in einer Suchstruktur gehalten, welche lediglich auf diesen Eigenschaften operiert, so könnten ggf. Punkte, welche aufgrund der Eigenschaften ausschliessbar sind, bereits durch den Aufbau der Datenstruktur auf optimale Weise ausgeschlossen werden; diese Struktur hätte eine Laufzeit, welche unabhängig von der Dimension wäre.

Im Anschluss werde ich zunächst auf die Eigenschaften eingehen, auf welchen meine Datenstruktur gründet, darauf folgt die Definition einer Datenstruktur, und deren Laufzeitanalyse. Abschliessend werde ich noch auf Verbesserungsmöglichkeiten eingehen.

Die Datenstruktur wird in Teil 4 mit den Datenstrukturen aus Abschnitt 2.4 verglichen.

3.1 Geometrische Kriterien

Eine zentrale Rolle für die in diesem Abschnitt vorgestellten Kriterien für schnelle Entscheidungen ist das Konzept der Diagonalen (siehe 2.1); die Diagonale D_0 (Diagonale durch Nullpunkt) wird im Weiteren als Bezugselement verwendet. Die verwendeten Eigenschaften sind für $x \in \mathbb{R}^M$:

$min(x), max(x) \in \mathbb{R}$**.** Minimale und maximale Komponente von x.

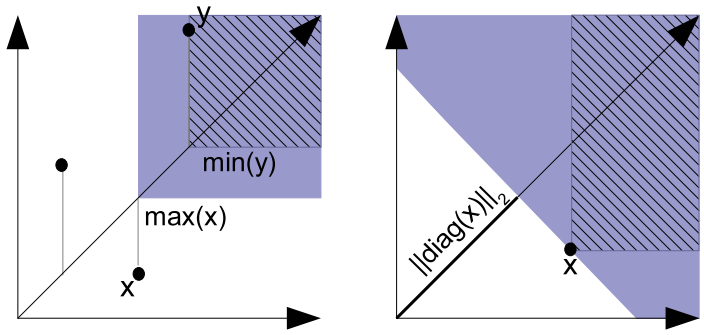
$t_x \in \mathbb{R}$**.** $D_0(t_x)$ beschreibt den Punkt auf der Diagonalen D_0, auf welchen der Punkt x durch orthogonale Projektion abgebildet wird.

$dist_x \in \mathbb{R}$**.** Abstand des Punktes x von der Diagonalen D_0 (siehe auch 2.1).

Es folgen die verwendeten Entscheidungskriterien.

Minimale und maximale Komponente

Ist die maximalste Komponente eines Punktes x kleiner als die minimalste Komponente eines Punktes y, so dominiert Punkt x den Punkt y strikt (siehe Abb. 10a), i.e. $max(x) < min(y) \rightarrow x \prec y$; sind die Komponenten hingegen identisch, dominiert Punkt x den Punkt y möglicherweise schwach.



(a) (b)

Abbildung 10: (a) Durch $max(x)$ ist der blaue, von x sicher dominierte Bereich festgelegt, durch $min(y)$ der schattierte Bereich, in welchem y liegt. (b) Von x dominierte Punkte liegen im schattierten Bereich, welcher ganz 'hinter' x, im blauen Bereich, liegt.

Ggf. kann mittels dieser Eigenschaft ein Dominanztest mit zwei Punkten mit lediglich einem Vergleich entschieden werden.

Es kann hingegen keine Aussage über die Dominanzrelation gemacht werden, wenn $max(x) > min(y)$; Punkt x kann Punkt y dominieren, muss aber nicht - z.B. $x = (1, 10), y_1 = (2, 11)$.

Leider kann in beliebigen Dimensionen aus dem Vergleich der minimalsten, und maximalsten Komponenten kein weiterer Schluss gezogen werden; wohingegen in zwei Dimensionen eine weitere Eigenschaft für Unvergleichbarkeit von $x, y \in \mathbb{R}^M$ besteht:

$$\begin{aligned}(([min(x), max(x)] \subset [min(y), max(y)]) \quad \vee \\ ([min(y), max(y)] \subset [min(x), max(x)])) \leftrightarrow x \parallel y\end{aligned}$$

D.h. ist das Interval $[min(x), max(x)]$ eine echte Teilmenge von $[min(y), max(y)]$, oder umgekehrt, so sind die Punkte x und y unvergleichlich. Diese Eigenschaft benötigt mindestens zwei Vergleiche, womit es sich nicht lohnt, damit einzelne Dominanzvergleiche durchzuführen. Allerdings ist in zwei Dimensionen der Einsatz eines Segment-/Interval-Baumes[5] überdenkenswert.

Projektion auf Diagonale

Betrachtet man für die Punkte x, y die Projektionen $D_0(t_x), D_0(t_y)$ (t_x, t_y die Koordinaten von x, y bezogen auf den Basisvektor $diag_e$) auf die Diagonale

[5][10], Seite 448-457.

D_0, so ist klar, dass $x \prec y \rightarrow t_x < t_y$ (siehe Abb. 10b); d.h. y liegt 'hinter' x.

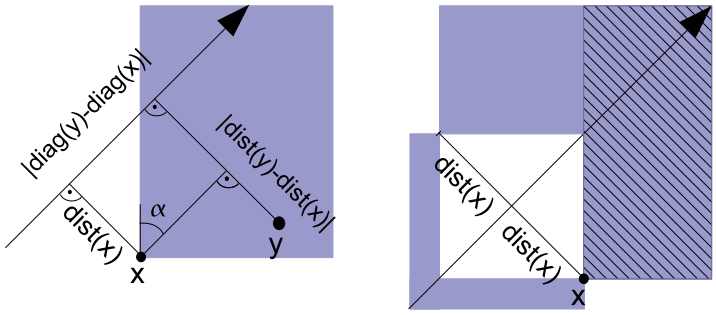
Insbesondere ist dies auch für die schwache Dominanzrelation gültig.

Diese Eigenschaft eignet sich in der Umkehrung als Ausschlusskriterium, i.e.

$$\neg(t_x < t_y) \rightarrow \neg(x \prec y)$$

d.h. liegt x 'hinter' y, so dominiert Punkt x den Punkt y nicht.

Abstand von Diagonale



(a) (b)

Abbildung 11: (a) Grundlage für drittes Kriterium in zwei Dimensionen, in extenso $\alpha = \frac{\pi}{4}$. Wenn Punkt x den Punkt y dominiert, so muss das Verhältnis $\frac{|dist_x - dist_y|}{||diag_x - diag_y||_2}$ kleiner gleich 1 sein. (b) Der blaue Bereich umfasst alle nach dem dritten Kriterium nicht ausschliessbaren Punkte; der schattierte Bereich umfasst alle von x dominierten Punkte.

Die dritte Eigenschaft verwendet zusätzlich die Distanz $dist_x, dist_y$ der Punkte $x, y \in \mathbb{R}^M$ von der Diagonalen, und stellt sie in Bezug zur Projektion auf die Diagonalen. Zentral ist der folgende Umstand: für alle Dimensionen M existiert eine minimale Konstante c_M, so dass gilt:

$$(x \prec y) \vee (y \prec x) \rightarrow (|dist(y) - dist(x)| < c_M \cdot |t_y - t_x|)$$

.

Aus Abbildung 11a ist ersichtlich, dass $c_2 = 1$.

Generell ist $c_M = tan(cos^{-1}(\frac{1}{\sqrt{M}}))$, was wie folgt einzusehen ist. Sei $R \subset \mathbb{R}^M$ das von x dominierte Hypervolumen. Lege man nun um R den kleinsten umfassenden, M-dimensionalen Kegel mit Spitze in x. Kegel und R haben all jene Punkte gemein, welche von x aus durch Vergrösserung nur

einer Komponente von x erreichbar sind. Der Innenwinkel α des Kegels ist somit gleich dem Winkel zwischen Diagonale und einem Einheitsvektor $e = (0, .., 0, 1, 0, .., 0) \in \mathbb{R}^M$, i.e. $cos(\alpha) = \frac{e * diag_e}{||e||_2 \cdot ||diag_e||_2} = \frac{1}{\sqrt{M}}$. Der Tangens leitet sich ab aus der Berechnung des Radius der Kegelhülle auf einer bestimmten Tiefe (siehe auch Abb. 11a).

Auch diese Eigenschaft ist in der Umkehrung als Ausschlusskriterium anwendbar, i.e.

$$\neg(|dist(y) - dist(x)| < c_M \cdot |t_y - t_x|) \rightarrow \neg((x \prec y) \vee (y \prec x))$$

Es sind jedoch zwei Aspekte zu beachten: (1) das Kriterium berücksichtigt in dieser Form nicht ob, welcher Punkt potenziell welchen dominiert, d.h. es ist sinnvollerweise in Kombination mit dem ersten Ausschlusskriterium zu verwenden (2) der M-dimensionale Raum wird auf zwei Dimensionen reduziert, wobei für das Distanzmass Informationen verloren gehen (alle Punkte mit Abstand d von der Diagonalen bilden in der Ebene zwei Geraden, im Raum aber eine Röhre!). Siehe auch Abb. 11b.

Es folgt die Definition einer Datenstruktur, welche die vorgestellten Kriterien verwendet.

3.2 Definition Datenstruktur

Struktur

Die Population wird in einer für die Punkte $p \in P$ nach t_p ($D_0(t_p) = diag_p$) sortierten Liste gehalten. Der Listencharakter ist notwendig, lässt sich aber z.B. mittels einer Skiplist, oder einem Range Tree[9] effizient umsetzen.

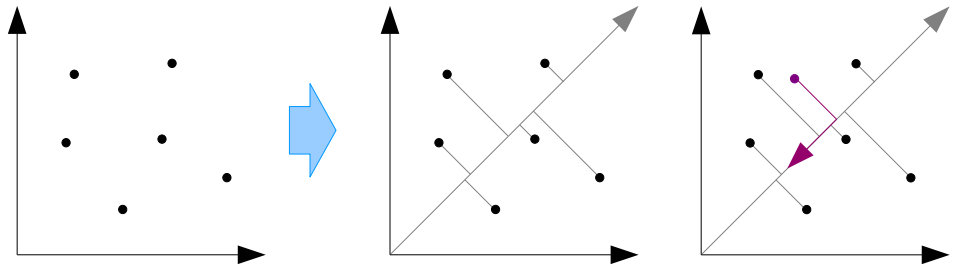
Soll ein Punkt x eingefügt werden, so werden erst alle verwendbaren geometrischen Eigenschaften berechnet; dies sind $min(x), max(x), t_p, dist_x$. Sodann wird der Punkt in die Liste eingefügt. Die Zeitkomplexität der Berechnung ist $O(M)$, diejenige der Einfügeoperation $O(logN)$.

Bei der Entfernung eines Punktes muss lediglich ein Listenelement entfernt werden; die Operation ist möglich in Zeitkomplexität $O(logN)$.

Ausführung Dominanzüberprüfungen

Die Ausführung der Dominanzüberprüfung wird für den Dominanztest durchgegangen, ist aber analog für das (Auf-)Zählen dominierter Punkte.

Soll ein Punkt $x \in \mathbb{R}$ bzgl. der Punktmenge getestet werden, werden erst die geometrischen Eigenschaften von x berechnet. Darnach wird in der Liste derjenige Punkt ausfindig gemacht, welcher die grösste, kleinere oder gleiche



(a) (b)

Abbildung 12: (a) Aufbau der Datenstruktur. Die Punktmenge wird bzgl. ihrer diagonalen Komponente geordnet. (b) Durchführung Dominanztest. Die diagonale Komponente des Abfragepunktes (magenta) wird auf der Liste lokalisiert, und die Liste anschliessend in Richtung der Kandidaten traversiert.

diagonale Komponente hat, und von diesem aus werden alle Punkte mit kleinerer diagonaler Komponente traversiert, und einzeln mit Zuhilfenahme der geometrischen Eigenschaften mit x verglichen. Der Algorithmus ist folgend dargestellt:

Algorithmus Dominanztest

Gegeben: Punktmenge der Grösse N in Datenstruktur P (wovon alle mit unterschiedlicher diagonaler Komponente), Abfragepunkt q

---

Berechne $min(q), max(q), t_q, dist_q$.
$dominiert \leftarrow falsch$
$p \leftarrow arg_{n \in P} \neg \exists p' \in P : t_n < t_{p'} \leq t_q$
if p existiert **then**
  repeat
    if $max(p) < min(q)$ **then**
      $dominiert \leftarrow wahr$
    else if $|dist(p) - dist(q)| < c_M \cdot |t_p - t_q|$ **then**
      $dominiert \leftarrow$ dominiert p q ?
    end if
    $p \leftarrow arg_{n \in P} \neg \exists p' \in P : t_n < t_{p'} \leq t_p$
  until $dominiert \vee p$ existiert nicht
end if
return $dominiert$

Die Zeitkomplexität des Dominanztests ergibt sich zu $O(M)$ (Berechnung Eigenschaften) und $O(kM)$ (k die Anzahl zu traversierender Punkte, $k \leq N$).

3.3 Laufzeit- und Speicherkomplexität

Folgend eine Zusammenstellung der Komplexitäten der Datenstruktur bzgl. in 2.4 vorgestellten Operationen (und Speicherbedarf). Es sei M die Dimension, und N die Grösse der Punktmenge. Von zentraler Bedeutung ist die Berechnung der geometrischen Eigenschaften, was in $O(M)$ Multiplikationen getan werden kann; zu beachten ist, dass die Laufzeit von Multiplikationen und Vergleichen von der Architektur abhängt (Multiplikationen aber tendenziell länger dauern). Für die Komplexitäten der Operationen auf der Liste wird angenommen, dass eine Skiplist[9] verwendet wird.

Speicher. Der von Listen benötigte Speicher ist linear in M, i.e. $O(M)$.

Einfügen. Die Einfügeoperation berechnet erst die geometrischen Eigenschaften ($O(M)$ Multiplikationen), und fügt dann den Punkt in die geführte Liste ($O(logN)$ Vergleiche).

Entfernen. Das entsprechende Element wird aus der Liste entfernt $\rightarrow O(logN)$ Vergleiche.

Aufbau mit N. Es gibt keine optimale Aufbauprozedur, in extenso müssen alle Punkte einzeln eingefügt werden. Die Komplexität ergibt sich somit zu $O(MN)$ Multiplikationen, $O(NlogN)$ Vergleichen.

Optimieren. Operation ist nicht definiert.

Dominanztest. Berechnen der geometrischen Eigenschaften der Abfragepunkte ($O(M)$ Multiplikationen), Lokalisierung des entsprechenden Wertes in der Liste ($O(logN)$ Vergleiche), Traversierung der Liste ($O(kM)$ Vergleiche, wobei k die Anzahl besuchter Punkte). Schlimmstenfalls werden alle Punkte traversiert, womit sich die Laufzeit zu $O(MN)$ ergibt.

Zählen. Analog zu Dominanztest.

Aufzählen. Analog zu Dominanztest, bis auf das noch die dominierten Punkte ausgegeben werden müssen, i.e. $O(MN + d)$ (d die Anzahl dominierter Punkte).

Die Worst-Case Laufzeit eines Dominanztests ist somit identisch zu jener von Listen, hat allerdings mehr Overhead; dies trifft jedoch auf die meisten Datenstrukturen für die Dominanzüberprüfung zu. Die Effizienz der Datenstrukturen beruht grundlegend auf der Gutmütigkeit für konkrete Populationen und Abfragepunkte.

In Teil 4 wird weiter darauf eingegangen.

3.4 Verbesserungsmöglichkeiten

Folgend einige Gedanken über mögliche Ausbesserungen der Datenstruktur.

Merken der kleinsten maximalen Komponente

Wird die kleinste maximale Komponente aller Punkte einer Menge gespeichert, so kann für einen Dominanztest zwischen der Berechnung der geometrischen Eigenschaften und den Vergleichen der einzelnen Kandidaten zuerst der Abfragepunkt bzgl. dieser Komponente getestet werden; ggf. wird der Abfragepunkt dominiert.

Der Vergleich benötigt konstante Zeit, und erspart potenziell viel Zeit.

Unter der Annahme, dass die kleinste maximale Komponente lediglich abnehmen kann, wird für das Merken nur ein Speicherelement benötigt, und es kann in konstanter Zeit beim Einfügen neuer Punkte angepasst werden, oder nicht. Kann der Wert variieren, so ist es überlegenswert einen Minimum-Heap[9] einzusetzen.

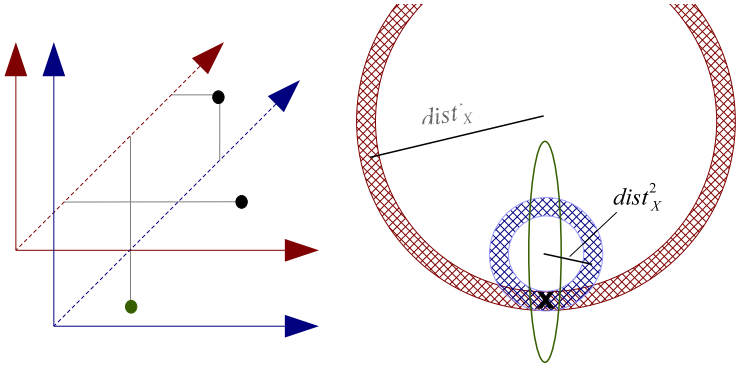
Wahl der Diagonalen

Bislang wurde D_0 als Bezugsdiagonale für alle Eigenschaften verwendet, doch dies ist nicht zwingenderweise so. Abbildung 13a zeigt ein einfaches Beispiel, wieso es für die Dominanzüberprüfungen von Vorteil ist, die Diagonale an die Punktmenge anzupassen. Dabei ist zu unterscheiden für welche Eigenschaft welche Diagonale gewählt werden sollte:

Minimale und maximale Komponente Extremalkomponenten haben Gültigkeit innerhalb eines Bezugssystems (hier: Nullpunkt). Wird der Nullpunkt virtuell verschoben, so ändern sich die Komponenten. Je näher ein Punkt an einer Diagonale ist, desto näher liegen Minimal- und Maximalkomponente beieinander (sie sind identisch, wenn der Punkt auf der Diagonalen liegt). Wie Abbildung 13a andeutet, hat die Diagonale idealerweise einen kleinen Abstand von allen Punkten, also wäre z.B. die Diagonale, welche durch den Schwerpunkt der Menge geht, i.e. $\frac{1}{|P|} \cdot \sum_{x \in P} x$, eine gute Wahl.

Projektion auf Diagonale Die Wahl der Diagonalen hat keinen Einfluss auf $diag_x$ und somit auch t_x ($D_0(t_x) = diag_x$), da die Punkte orthogonal zum Verlauf der Diagonalen projeziert werden.

Abstand von der Diagonalen Ist der Abstand eines Punktes von der Diagonalen gleich Null, so ist der Ausschlussbereich am grössten (der Effekt der Drehung um die Diagonale ist am kleinsten). Da es unmöglich ist



(a) (b)

Abbildung 13: (a) Auswirkung unterschiedlicher Bezugssyteme (rot, blau) auf minimale (für schwarze Punkte eingezeichnet) und maximale Komponente (für grünen Punkt eingetragen). Im blauen System dominiert der grüne Punkte beide Schwarzen nach dem ersten Kriterium, im roten System lediglich den einen Punkt. (b) Kandidatenmengen nach zweitem Ausschlusskriterium für zwei unterschiedliche Diagonalen, Sicht orthogonal zu Diagonalen. Asymmetrische Population (grün umrandet), x Abfragepunkt, Bereiche mit gleichem Abstand von x zu den beiden Diagonalen (blau, resp. rot schattiert). Die Schnittmenge von Population und schattierten Bereichen bildet die Kandidatenmenge.

den Abstand für alle Punkte gleich Null zu haben, ist es vielleicht naheliegend eine allen Punkten möglichst nahe Diagonale (wie oben) zu verwenden. Allerdings ist im Hinterkopf zu behalten, dass für gewisse Punktmengen (z.B. stark asymmetrische), Diagonalen vorteilhafter sind, welche die Eigenheiten der Punktmenge (Asymmetrie) ausnützen (siehe Abb. 13b).

Da die Wahl der Diagonalen die Eigenschaften der Punkte beeinflusst, kann man ggf. einen Schritt weiter gehen, und mehrere Bezugsdiagonalen verwenden. Für die Extremalkomponenten ist dies z.B. interessant wenn die Punktmenge aus mehreren kleinen Population besteht (mehrere angestrebte Idealpunkte).

Grundsätzlich besteht die Möglichkeit die Diagonale(n) laufend an die Punktmenge anzupassen, da dies jedoch eine teure Operation ist (alle Eigenschaften der Punkte müssen neu berechnet werden), sollte dies nicht zu oft getan werden.

Eigenschaften als Koordinaten

Werden t_x und $dist_x$ als Koordinaten eines Punktes in der Ebene aufgefasst, erschliesst sich eine weitere Möglichkeit, um die Kandidaten zu ermitteln (siehe Abb. 14). Die Kandidatenmenge des zweiten Ausschlusskriteriums wäre sodann gleich dem von zwei Geraden vertikal eingeschlossenen Bereich; die Geraden kreuzen sich in den neuen Koordinaten des Abfragepunktes, und ihre Steigungen betragen $\pm c_M$. Die Frage nach den eingeschlossenen Punkten könnte dann mit einem Bereichsabfragen-Ansatz angegangen werden.

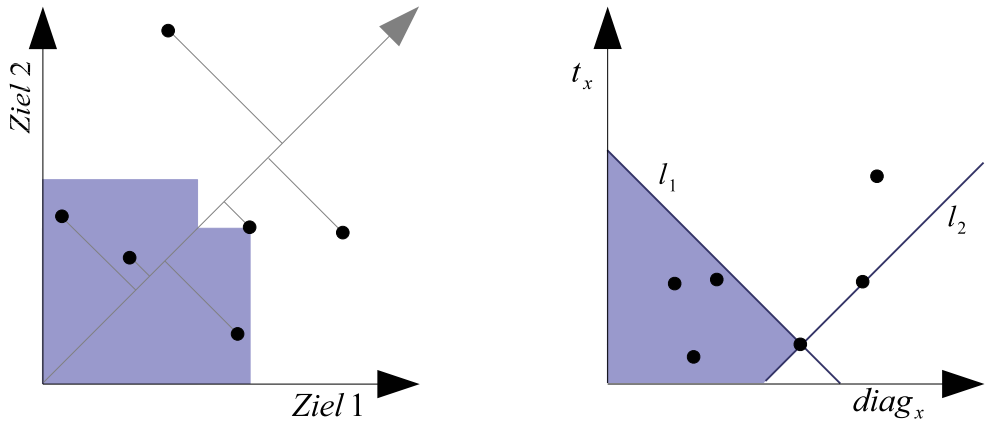


Abbildung 14: Betrachtung des potenziell dominierenden Bereichs (blau) im gegebenen Koordinatensystem (links), und mit neuen Koordinaten (rechts) nach Anwendung des ersten und zweiten Ausschlusskriteriums. Die Menge befindet sich oberhalb von l_2, und unterhalb von l_1.

4 Vergleich Datenstrukturen

Die Aufgabe Datenstrukturen für die Dominanzüberprüfung in Evolutionären Algorithmen zu vergleichen ist geteilt in verschiedene Aspekte. Auf der einen Seite stehen die Datenstrukturen, welche zu vergleichen sind; von diesen interessiert welches Verhalten (i.e. Laufzeiten) sie für welche Punktmenge an den Tag legen (z.B. welches Verhalten zeigt sich für konvexe, nichtdominierte Fronten?). Auf der anderen Seite stehen die Evolutionären Algorithmen, in welchen die Datenstrukturen eingesetzt werden sollen; diese stellen in ihren Ausprägungen und für unterschiedliche Punktmengen unterschiedliche Anforderungen an die Datenstrukturen (z.B. werden, wenn mehr Dominanzüberprüfungen durchzuführen sind, Operationen für Dominanzüberprüfungen stärker gewichtet als Wartungsoperationen).

Forschungsberichte, welche diese beiden Aspekte von theoretischer Seite angehen, scheinen selten; vielmehr werden die jeweils vorgestellten Datenstrukturen zusammen mit wenigen Vergleichsdatenstrukturen für bestimmte Standardprobleme in EA empirisch getestet.

Im Anschluss werde ich zunächst interessante Eigenschaften von Punktmengen erläutern, wovon ich lediglich diejenigen weiterverwenden werde, welche sich für einen praktischen Vergleich der Datenstrukturen eignen. Darnach werde ich spezifische Teile von EA mit Dominanzüberprüfungen aufzeigen und für den Einsatz mit Datenstrukturen restrukturieren. Darauf folgt eine Betrachtung der Worst-Case Laufzeiten der Algorithmen mit Verwendung unterschiedlicher Datenstrukturen. Es wurden vier Datenstrukturen implementiert, und empirisch getest; was im abschliessenden Abschnitt folgt.

4.1 Modelle Punktmengen

Dieser Abschnitt führt die qualitativen Modelle (Frontform, Stichprobenraum) ein, welche bei den weiteren Betrachtungen der Datenstrukturen wieder auftauchen. Ausserdem werden verschiedene Aspekte von Punktmengen erwähnt, welche für weitere Analysen von Interesse sein könnten.

Abschliessend wird ein Frontenmodell präsentiert, welches ich zur Generierung von Punktmengen verwendet habe, und sich ggf. zum Modellieren und insb. Analysieren der zuvor beschriebenen Aspekte eignet.

4.1.1 Generelle Eigenschaften

Frontenform

In Evolutionären Algorithmen (EA) werden oft (z.b. NSGA-II[5]), SPEA2[6]) nicht-dominierte Fronten, i.e. Punktmengen mit paarweise unvergleichlichen

Punkten, z.B. in Form eines Elitearchivs geführt. Abfragen dieser EA entsprechen dann i.d.R. Abfragen bzgl. dieser nicht-dominierten Front. Daraus resultiert mein Interesse das Verhalten von Datenstrukturen für unterschiedliche Fronten weiter zu untersuchen.

Eine leicht aufzeigbare Eigenschaft von nicht-dominierten Fronten ist, dass, wenn die Punkte auf den zu $diag_e$ orthogonalen, $(M-1)$-dimensionalen Hyperraum projeziert werden, keine zwei Punkte an die gleiche Stelle zu liegen kommen. Diese Eigenschaft kann ggf. verwendet werden, um nicht-dominierte Fronten dadurch zu modellieren, indem zunächst diese im $(M-1)$-dimensionalen Hyperraum generiert und dann hochprojeziert werden.

Ich werde mich auf drei qualitativ beschriebene Frontenformen beschränken, welche im Besonderen eine unterschiedliche Beziehung zum Hyperraum haben; die Formen sind stark karikaturiert, und symmetrisch um die Raumdiagonale (D_0) (siehe auch Abb. 15):

Stark dominierend. Die Front wird gebildet durch einen M-dimensionalen Kegel mit Spitze in $x \in \mathbb{R}^M$, und Symmetriezentrum parallel zu $diag_e$. Wird der Kegel in die kleinste umfassende Box gepackt, so dominiere der Kegel den Grossteil der Box (Abb. 15a).

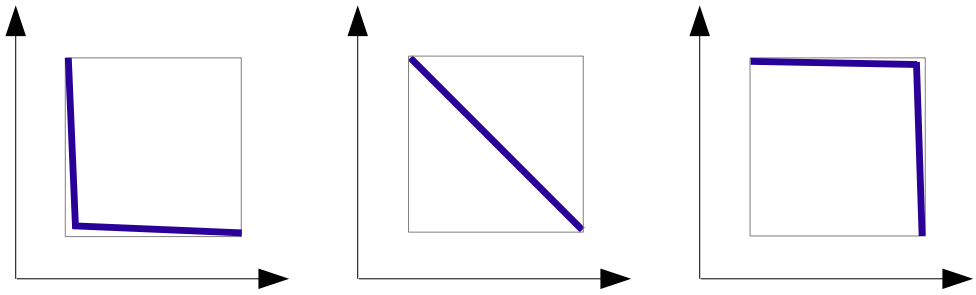
Mässig dominierend. Die Punkte liegen in der $(M-1)$-dimensionalen Hyperebene orthogonal zur Raumdiagonale D_0; im Besonderen ist dies ein Kegel nach selbiger Regel wie oben, aber mit Innenwinkel $\frac{\pi}{2}$ (Abb. 15b).

Schwach dominierend. Symmetrisch zur stark dominierten Front (Abb. 15c).

Stichprobenraum

Die Definition eines Stichprobenraums erleichtert die Einschätzung des Laufzeitverhaltens einiger Datenstrukturen indem ein Bild der Anzahl dominierender, respektive dominierter Punkte eines Abfragepunktes vermittelt werden kann.

Ich unterscheide zwischen drei Fällen, welche sich gleichermassen auf alle Frontenformen anwenden lassen. Sei $P \subset \mathbb{R}^M$ die Population. Es folgen die Stichprobenräume:



(a) (b) (c)

Abbildung 15: Formen von nicht-dominierten Fronten (blau), und kleinste umfassende Box (grau). (a) Stark dominierend. (b) Mässig dominierend. (c) Schwach dominierend.

Box Der Stichprobenraum entspricht einem Hyperrechteck in M Dimensionen, welches durch einen idealen Punkt $b^l \in \mathbb{R}^M$, und maximalen Punkt $b^h \in \mathbb{R}^M$ beschränkt wird, es gilt:

$$i \in \{1..M\} : b_i^l = min_{p \in P}(p_i), b_i^h = max_{p \in P} p_i$$

Siehe auch Abb. 15.

Die Box als Stichprobenraum ist motiviert durch das Monte Carlo Verfahren von J. Bader et al. (siehe 2.3.3), bei welchem Stichproben aus einer Box genommen werden.

Adaptiv Dieser Stichprobenraum sei qualitativ beschrieben.

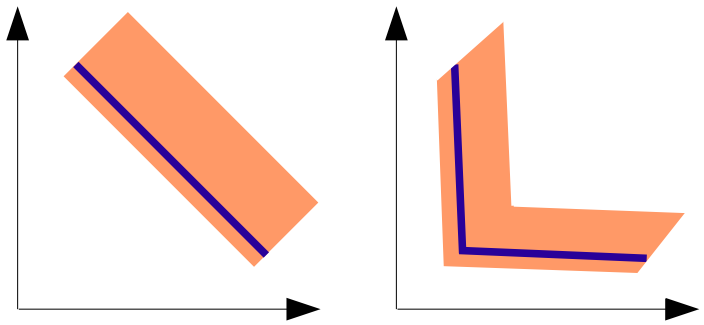
Der Stichprobenraum wird aufgespannt durch mehrere diagonal verschobene Bezugspopulationen. Dabei seien 4 von 5 Kopien von der ursprünglichen Population dominiert.

Der adaptive Stichprobenraum gründet auf der Annahme, dass Evolutionäre Algorithmen neue Individuen generieren deren Zielwerte sich in der Nähe der bisherigen Punktmenge befinden (Form), und dass diese auch tendenziell schlechter sind (Verhältnis Anzahl dominierte Punkte zu totale Anzahl Punkte).

Zweitpopulation Die Abfragepunkte werden aus einer Population, sei dies eine Zweitpopulation oder die Population auf welcher Dominanzüberprüfungen ausgeführt werden, gezogen.

Diverse Evolutionäre Algorithmen für die Mehrzieloptimierung (z.B. MOGA[a]) führen Dominanzüberprüfungen auf einer Punktmengen aus, wobei die Abfragepunkte aus derselben Menge gezogen werden. Hieraus folgt die Zweitpopulation als Stichprobenraum.

[a] [2], Seiten 200-209.



(a) (b)

Abbildung 16: Beispiele des adaptiven Stichprobenraums (orange) für 2 nichtdominierte Fronten (blau).

Weitere Eigenschaften

Punktmengen als Populationen und Abfragepunkte verfügen über weitere Eigenschaften, welche präzisere Einschätzungen des Verhaltens von Datenstrukturen und Evolutionären Algorithmen erlauben (z.B. wie stark sind Wartungsoperationen in diesem Verfahren für diese Population zu gewichten?). Mögliche Eigenschaften werden folgend kurz erläutert.

Anzahl dominierter/dominierender Punkte. Ist die Anzahl der von einem Abfragepunkt dominierten und ihn dominierenden Punkte bekannt, so kann versucht werden den Einfluss auf die Datenstrukturen im Best- und Worst-Case zu ermitteln; z.B. ergibt sich für Listen ein sehr direkter Zusammenhang zu der Anzahl benötigter Dominanzvergleiche bei Dominanztests (siehe auch 4.3.2).

Verbesserungshäufigkeit einer Front. In elitären Evolutionären Algorithmen wird meist entweder ein Archiv einer nicht-dominierten Front geführt (z.B. PAES[a]), oder aus einer beliebigen Punktmenge eine Front aufgebaut (siehe auch 2.3.1). Was helfen würde die Gewichtung von Operationen für die Dominanzüberprüfung, respektive Wartung besser einzuschätzen, ist die effektive Verbesserungshäufigkeit der jeweiligen Front; i.e. wie oft werden neue, bislang nicht-dominierte Punkte zur Front hinzugefügt, und dominierte Punkte entfernt? Ggf. kann die erwartete Anzahl dominierter/dominierender Punkte zu einer Abschätzung herangezogen werden.

[a] [2], Seiten 272-279.

Maximale Frontgrösse. Wird für eine beliebige Population der Grösse N eine Front nach 2.3.1 aufgebaut, so ist es eine grobe Abschätzung für die Laufzeiten der Datenstruktur N direkt zu verwenden; ist z.B. die Population stark geordnet, so ist die maximale Frontgrösse wesentlich kleiner (z.B. 1 für eine unter der Dominanzrelation total geordneten Menge).

Es folgt die Vorstellung eines Modells von Populationen, welches versucht die Berechenbarkeit der obigen Eigenschaften zu berücksichtigen.

4.1.2 Frontenmodell

Um theoretische Aussagen, respektive Abschätzungen über das Verhalten von Datenstrukturen und Evolutionären Algorithmen zu überprüfen, sollten diese empirisch getestet werden. Da Analysen i.d.R. für spezifische Punktmengen mit bestimmten Eigenschaften (siehe oben) durchgeführt werden, sollte es möglich sein konkrete Populationen zu generieren, welche (nahezu) über die

gewünschten Eigenschaften verfügen.

Hierzu stelle ich nun ein Modell vor, welches aus der Vereinigung von $k \in \mathbb{N}$ formgleichen, nicht-dominierten Fronten P^i $(i = 1..k)$ entsteht. Jeder Front P^i ist dabei eine Basis β^i zugeordnet, welche alle auf einer gegebenen Raumdiagonalen $D \in \mathbb{D}$ liegen, so dass $\beta^i = \beta^1 + (i-1) \cdot d \cdot diag_e$ $(i = 1..k,\ d \in \mathbb{R}_{>0})$. Die Form der Fronten (siehe 4.1.1) wird beschrieben durch einen M-dimensionalen Kegel mit Spitze in der jeweiligen Basis, und Drehachse parallel zur Raumdiagonalen. Die genaue Form ist gegeben durch den Winkel $\alpha \in (-\frac{\pi}{4}, \frac{\pi}{4})$ zwischen Kegelwand und der Hyperebene orthogonal zur Raumdiagonalen. Punkte werden so auf der Kegelwand generiert, dass der Abstand der Punkte von der Raumdiagonalen, respektive von der entsprechenden Basis gleich der, respektive proportional zu den aus einer nicht-negativen Verteilungsfunktion Δ gezogenen Werten δ_j ist. Für die k Fronten sei eine Gewichtung $w^i \in [0, 1]$ gegeben (insb. $\sum_{i=1}^{k} w^i = 1$), mittels welcher die Verteilung der Punkte auf die Fronten gesteuert wird, i.e. w^i der Punkte sind in Front i anzutreffen. Abbildung 17 stellt das Modell graphisch dar.

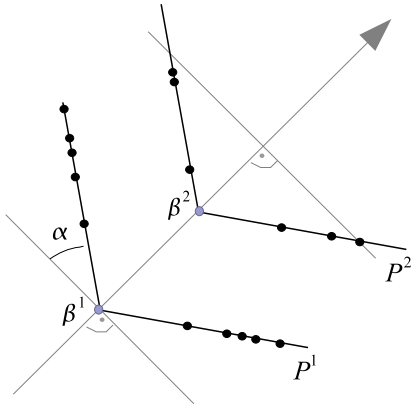


Abbildung 17: Prinzip Frontenmodell mit $N = 16$ Punkten in zwei Dimensionen. Fronten P^1, P^2 mit Basispunkten β^1, β^2, Frontenverschiebung d, Frontengewichten $w^1 = \frac{10}{16}, w^2 = \frac{6}{16}$. Die Frontenform (schwarze Segmente) ist gegeben für positives α. Der Abstand der effektiven Punkte (schwarz) von der Diagonalen wird aus Δ gezogen.

Das Modell ist unabhängig von der Populationsgrösse und der Dimension, was es leicht adjustierbar macht. Die maximale Frontengrösse kann generell leider nicht als $max_{i=1..k}(|P^i|)$ angenommen werden. Offen sind die Probleme der erwarteten maximalen Frontgrösse, Anzahl dominierter/dominierender

Punkte, und der Verbesserungshäufigkeit.

Im Appendix ist ein MATLAB Script angegeben, welches zur Generierung von Punkten nach obigem Modell verwendet werden kann.

4.2 Algorithmen mit Dominanzüberprüfungen

In diesem Abschnitt werden existierende Evolutionären Algorithmen mit Dominanzüberprüfungen restrukturiert, um Datenstrukturen für Dominanz überprüfungen zu unterstützen.

Die Laufzeitkomplexitäten, welche unterschiedliche Anforderungen an die Datenstrukturen stellen, werden im Anschluss in 4.3.1 bei der Eignungsabschätzung der Datenstrukturen wieder aufgegriffen.

Verwendet werden nebst einem Algorithmus für den Dominanztest mit fortschreitender Front (siehe Abschnitt 2.2) die Dominanzüberprüfungen betreffenden Elemente der in Abschnitt 2.3 vorgestellten Verfahren. Für SPEA2, und das Monte Carlo Verfahren von J. Bader et al. werden zudem Varianten von Algorithmen vorgestellt.

Für jeden Algorithmus werden Laufzeiten unter Verwendung von Funktionskürzeln für die Operationen zusammengestellt. Die Operationen sind abhängig von der jeweiligen Populationsgrösse N. Die Kürzel seien folgend tabelliert.

Operation	Kürzel
Einfügen	$T_{Einf}(N)$
Entfernen	$T_{Entf}(N)$
Aufbau mit N	$T_{AufbauN}(N)$
Optimieren	$T_{Opt}(N)$
Dominanztest	$T_{Test}(N)$
Zählen	$T_{Z}(N)$
Aufzählen	$T_{Az}(N)$

Die angebenen Laufzeiten beschränken sich dabei auf die Operationen, welche auf den Datenstrukturen aufgerufen werden. Konstante Faktoren, welche für alle Datenstrukturen identisch sind werden vernachlässigt.

Es folgen die Algorithmen und Laufzeiten.

4.2.1 Dominanztest mit fortschreitender Front

Für eine Erklärung sei auf Abschnitt 2.2 verwiesen.

Algorithmus Dominanztest mit fortschreitender Front
Gegeben: Populationsarchiv A der Grösse N, Stichprobenraum S

while *Stop-Kriterium* **do**
  $q \in_{u.a.r.} S$
  if q wird von keinem Punkt aus A dominiert **then**
    Finde und entferne von q dominierte Punkte.
    Füge q zu Archiv A hinzu
  end if
  if A degeneriert **then**
    Optimiere A
  end if
end while
return A

Laufzeit: $T_{Test}(N) + T_{Einf}(N) + T_{Az}(N) + k \cdot T_{Entf}(N) + o \cdot T_{Opt}(N)$
k = Anzahl dominierter Punkte
o = Optimierungshäufigkeit

Die zentralste Komponente dieses Algorithmus ist das Testen auf Nicht-Dominanz. Die weiteren Operationen (Entfernen, Einfügen, Optimieren) werden generell weniger frequent aufgerufen.

4.2.2 Nicht-dominiertes Sortieren

Siehe auch Abschnitt 2.3.1 und [5].

Algorithmus Frontentdeckung
Gegeben: Unstrukturierte Population P der Grösse N

Initialisiere Datenstruktur ND {Enthält bisher nicht-dominierte Punkte}
$D \leftarrow \emptyset$ {Enhält dominierte Punkte}
$p \leftarrow$ erstes Element aus P
$P \leftarrow P \setminus \{p\}$
Füge p zu ND
for all $p \in P$ **do**
  if p nicht-dominiert von ND **then**
    $liste \leftarrow$ zähle von p dominierte Punkte in ND auf
    for all $l \in liste$ **do**
      Enferne l aus ND
    end for
    $D \leftarrow D \cup liste$
    Füge p zu ND

end if
end for
return (ND, D)

Laufzeit: $T_{Einf}(0) + \sum_{i=1}^{N-1}(T_{Test}(i) + T_{Az}(i) + \sum_{j=1}^{d}(T_{Entf}(i-j)) + T_{Einf}(i))$
d = Anzahl dominierter Punkte, insb. $d \leq i$

Das erste Summenzeichen ergibt sich aus der Tatsache, dass die Grösse der nicht-dominierten Front für jeden Punkt aus P höchstens um eins zunimmt; das Zweite ergibt sich analog aus der steten Verkleinerung.

Eine wichtige Komponente der Frontentdeckung ist das Testen auf Dominanz. Da allerdings generell während dem Aufbau der Front ständig Punkte verworfen werden müssen, sind die Wartungsoperationen stärker zu gewichten als beim Dominanztest mit fortschreitender Front.

4.2.3 SPEA2

Anstatt die in Unterabschnitt 2.3.2 beschriebenen zwei Schritte zur Berechnung der Grundfitness in der angegebenen Reihenfolge durchzuführen, schlage ich vor, die Schritte wie folgt zu kombinieren:

1. Initialisiere für alle $p \in E' \cup P' : \rho(p) = 0$.
2. Ermittle für alle $p \in E' \cup P'$ die Menge Δ_p der dominierten Punkte; die Grösse von Δ_p entspricht $\delta(p)$. Besuche dann alle $d \in \Delta_p$ und update die Grundfitness wie folgt: $\rho(d) = \rho(d) + \delta(p)$.

Es ergeben sich für diese Variante zwei Vorteile (es sei angenommen, dass die zwei Punktmengen in 2 Datenstrukturen gehalten werden): (1) dass, da für alle Punkte nur eine Operation mit Dominanzüberprüfung durchgeführt werden muss, die Gesamtkosten von Dominanzüberprüfungen verringert wird (2) dass, da gesamthaft die Dominanzüberprüfungen (für Front- und Fitnessbestimmung) auf separaten Datenstrukturen ausgeführt werden können, die Datenstrukturen an die jeweilige Dominanzüberprüfung angepasst werden können. Das sequenzielle Besuchen der dominierten Punkte ist quasi kostenfrei, da dies in der ursprünglichen Variante dem Summieren der $\delta(p)$ aller dominierender Punkte p gleichkommt.

Für den Algorithmus wird angenommen, dass E in Form einer Datenstruktur und P als Menge gegeben ist.

Algorithmus Fitnessberechnung SPEA2

Gegeben: Nicht-dominierte Elitemenge E der Grösse K, unstrukturierte Zweitpopulation P der Grösse L, Grössenlimit Elitearchiv G

```
{Separiere neue dominierte/nicht-dominierte Punktmengen}
P' ← ∅
for all p ∈ P do
  if p von E nicht-dominiert then
    P ← P \ {p}
    liste ← Zähle alle e ∈ E auf, welche von p dominiert werden
    for all l ∈ liste do
      Entferne l aus E
      P' ← P' ∪ {l}
    end for
    Füge p zu E hinzu
  end if
end for

{Baue/Optimiere Datenstrukturen}
if |E| > G then
  Reduziere E auf Grösse G
end if
if E degeneriert then
  Optimiere E
end if
P ← Konstruiere Datenstruktur aus P ∪ P'

{Berechne Fitness}
for all p ∈ E' ∪ P do
  rho(p) ← 0
end for
for all e ∈ E ∪ P do
  liste ← Zähle alle p ∈ P auf, welche von e dominiert werden
  δ(e) ← Länge von liste
  for all l ∈ liste do
    ρ(l) ← ρ(l) + δ(e)
  end for
end for
```

Es sei angenommen, dass das Elitearchive E nach Reduzierung die Grösse G hat, und dass die Anzahl der dominierten Punkte durch $K+L-G$ gegeben ist.

Laufzeit Separation: $\sum_{i=1}^{L}(T_{Test}(K+i) + T_{Az}(K+i) + \sum_{j=1}^{d}(T_{Entf}(K+i)))$
d = Anzahl dominierter Punkte in E, insb. $d \leq K+i$

Laufzeit Aufbau/Optimierung: $o \cdot T_{Opt}(G) + T_{AufbauN}(K+L-G)$
o = Optimierungshäufigkeit

Laufzeit Fitnessberechnung: $(K+L) \cdot T_{Az}(K+L-G)$

Bei SPEA2 entstehen im zweiten Schritt (Zählen, respektive Aufzählen von Punkten) die grössten Kosten; der erste Schritt (Bestimmung der gesamthaft nicht-dominierten Menge (dem Dominanztest mit fortschreitender Front ähnlich)) ist generell günstiger.

4.2.4 Fast Hypervolume Based Search by Monte Carlo Sampling

Das Verfahren wurde prinzipiell in Abschnitt 2.3 erläutert. Die Modelle orientieren sich an der in [3] beschriebenen homogenen, natürlichen Selektion.

Die Methode wird in zwei Varianten ausgeführt, wobei nicht beachtet wird, wie die Stichprobenräume berrechnet werden. Die Varianten seien nun erläutert (sei p der Punkt, dessen Fitness berechnet werden soll):

Variante 1. Aus allen Punkten, die in den jeweiligen Stichprobenräume liegen (exklusive p), oder diesen definieren (liegen nicht im Stichprobenraum, maximal M), wird eine neue Datenstruktur aufgebaut. Stichproben werden dann bzgl. dieser Struktur auf Dominanz getestet. Nichtdominierte Abfragepunkte werden demnach nur von p dominiert.

Vorteile: Datenstrukturen zur Berechnung von Stichprobenraum und Fitness sind unabhängig, i.e. können spezialisiert werden. Verwendung der generell kostengünstigten Dominanzüberprüfung

Nachteile: Es wird immer aus potenziell vielen Punkten eine neue Datenstruktur aufgebaut. Die Punkte, gegen welche getestet wird, definieren generell keine nicht-dominierte Front, so dass die Datenstruktur unnötige (dominierte) Punkte enthält.

Variante 2. Alle Punkte werden in einer Datenstruktur gehalten, und die Dominanzüberprüfungen (Zählen dominierter Punkte) werden auf dieser ausgeführt. Wird ein Abfragepunkt von nur einem Punkt dominiert, so ist dies p.

Vorteile: Kosten entstehen nur durch die Dominanzüberprüfung, und die Instandhaltung der Datenstruktur.

Nachteile: Die Kosten der Dominanzüberprüfung sind abhängig von der Grösse der Gesamtpopulation, und nicht von einer potenziell kleinen Teilmenge. Verwendet nicht die kostengünstigste Dominanzüberprüfung; allerdings kann die Operation bei Entdeckung von mindestens 2 dominierenden Punkten abgebrochen werden.

Es folgen die Grundstruktur des Algorithmus, und die Varianten des Monte Carlo Samplings.

Algorithmus Homogene natürliche Selektion

Gegeben: Population P der Grösse N, Anzahl Stichproben J, Populationslimite G

---

while $|P| > G$ **do**
  for all $p \in P$ **do**
    $R \leftarrow$ Stichprobenraum für p

    {Monte Carlo Sampling Variante 1}
    $T \leftarrow$ Alle R definierenden, oder in R liegenden Punkte ausser p
    $f(p) \leftarrow$ Monte Carlo Sampling (Variante 1) mit T, R, J

    {Monte Carlo Sampling Variante 2}
    if P degeneriert **then**
      Optimiere P
    end if
    $f(p) \leftarrow$ Monte Carlo Sampling (Variante 2) mit P, R, J
  end for
  Entferne Punkt mit schlechtester Fitness aus P
end while
return P

Algorithmus Monte Carlo Sampling Variante 1

Gegeben: Unstrukturierte Population P, Stichprobenbox R, Anzahl Stichproben J

Initialisiere Datenstruktur T
Aufbau T mit P
$nd \leftarrow 0$
while $J > 0$ **do**
  $q \in_{u.a.r.} R$
  if q nicht von T dominiert **then**
    $nd \leftarrow nd + 1$
  end if

$J \leftarrow J - 1$
end while
return $Volumen_R * \frac{J-nd}{J}$

Algorithmus Monte Carlo Sampling Variante 2
Gegeben: Strukturierte Datenstruktur P, Stichprobenbox R, Anzahl Stichproben J

$nd \leftarrow 0$
while $J > 0$ **do**
$q \in_{u.a.r.} R$
if q von mindestens 2 Punkten aus T dominiert **then**
$nd \leftarrow nd + 1$
end if
$J \leftarrow J - 1$
end while
return $Volumen_R * \frac{J-nd}{J}$

Laufzeit Variante 1: $T_{AufbauN}(|T|) + J \cdot T_{Test}(|T|)$
$|T| \leq N$

Laufzeit Variante 2: $J \cdot T_Z(N) + oT_{Opt}(N)$
$o =$ Optimierunshäufigkeit

Wenn in Variante 2 das Zählen nach zwei gefundenen Punkten abgebrochen wird entsprechen sich tendenziell die Test- und Zähl-Zeit. Es entsteht die Frage nach den Aufbau- und Optimierungskosten; diese entsprechen sich i.d.R., so dass die zweite Variante kostengünstiger sein müsste, da die Datenstruktur lediglich einmal in der inneren Schlaufe von *Homogene natürliche Selektion* aufgerufen werden muss.

4.3 Theoretische Betrachtungen der Datenstrukturen

Dieser Abschnitt beinhaltet die theoretische Gegenüberstellung der Datenstrukturen für die Dominanzüberprüfung. Zunächst werden die Worst-Case Laufzeiten der Datenstrukturen mit den in 4.2 beschriebenen Algorithmen kombiniert und verglichen. Darnach werden für einzelne Datenstrukturen qualitative Aussagen (z.B Verhalten für unterschiedliche Punktmengen) gemacht.

4.3.1 Laufzeitvergleich

Da die verwendeten Operationskosten der Datenstrukturen das Verhalten im Worst-Case darstellen, und diese lediglich in der Asymptotik strikte Gültigkeit haben, sind die Resultate mit Vorsicht zu geniessen. Sie sollen vielmehr einen Überblick der Nützlichkeit der einzelnen Datenstrukturen erlauben.

Dominance Decision Trees, Dominated- und Non-Dominated-Trees, sowie mein eigener Ansatz werden nicht betrachtet, da deren Worst-Case Laufzeiten mindestens so hoch wie diejenigen von Listen sind, also von Listen sicher geschlagen werden.

Im Anschluss werden erst Annahmen bzgl. nicht gegebener Laufzeiten von Operationen einzelner Datenstrukturen getroffen. Dann werden die Laufzeiten präsentiert und kommentiert.

Annahmen

K-Range Tree

Einfügen Sei die Aufbauzeit für N Punkte für die einzelnen Punkte gemittelt, i.e. $\frac{O(MN+N(logN)^{M-1})}{N} = O(M + (logN)^{M-1})$

Entfernen Für Bäume belaufen sich oft die Entfernkosten auf ähnliche Werte wie die Einfügekosten. Der in M lineare Term sei weggelassen, da dieser das Kopieren des Koordinatenvektors in den internen Speicher der Datenstruktur repräsentiert und beim Entfernen wegfällt $\rightarrow O((logN)^{M-1})$.

Quad-tree

Aufbau mit N Ein Quad-Tree optimaler Tiefe wird durch das rekursive Einfügen des zentralsten Elements des jeweils verbleibenden Hyperraums konstruiert. Zufällige binäre Suchbäume verfügen über eine asymptotisch optimale Tiefe (siehe [14]). Infolgedessen wird angenommen, dass sich zufällige Quad-Trees analog verhalten, i.e. dass ein nahezu optimaler Quad-Tree durch eine beliebige Permutation der Einfügesequenz erstellt werden kann: $O(MN + Nlog(\frac{N}{M}))$.

Optimieren Die Kosten der Optimierung seien den Kosten des Aufbaus mit N Punkten gleichgestellt.

Binary Decision Diagrams

Aufbau mit N Die experimentellen Einfügekosten (siehe [21]) sind linear in N. Es liegt nahe, inkrementellen Aufbau anzunehmen, so dass sich die Kosten des Aufbaus ergeben zu $O(MN) + \sum_{i=1}^{N} O(i) = O(MN + N^2)$.

Laufzeitbetrachtungen

Die Laufzeiten wurden jeweils für variable Populationsgrösse N bei konstanter Dimension $M = 2$ und für $N = 1000$, variables M untersucht. Das Verhalten der einzelnen Datenstrukturen wird in der Folge zusammengefasst. Beispiel-Laufzeiten sind den Abbildungen 18, 19, 20 zu entnehmen.

Listen

Listen haben vernachlässigbare Wartungskosten; sie entsprechen generell dem möglichen Minimum. Die wesentlichen Kosten von Listen entstehen durch die Dominanzüberprüfungen, welche linear in N und M sind[a].
Während Listen generell nicht zu den optimalsten Datenstrukturen gehören, stellen sie über alle Kontexte einen Kompromiss dar, insb. für grössere Populationen in vielen Dimensionen. Sie eignen sich demzufolge als Ausgangspunkt für Verbesserungen, und nicht weiter spezifierte Anwendungen.

[a] Dies sei jedoch für gewisse Anwendungsfälle zu relativieren, siehe auch 4.3.2.

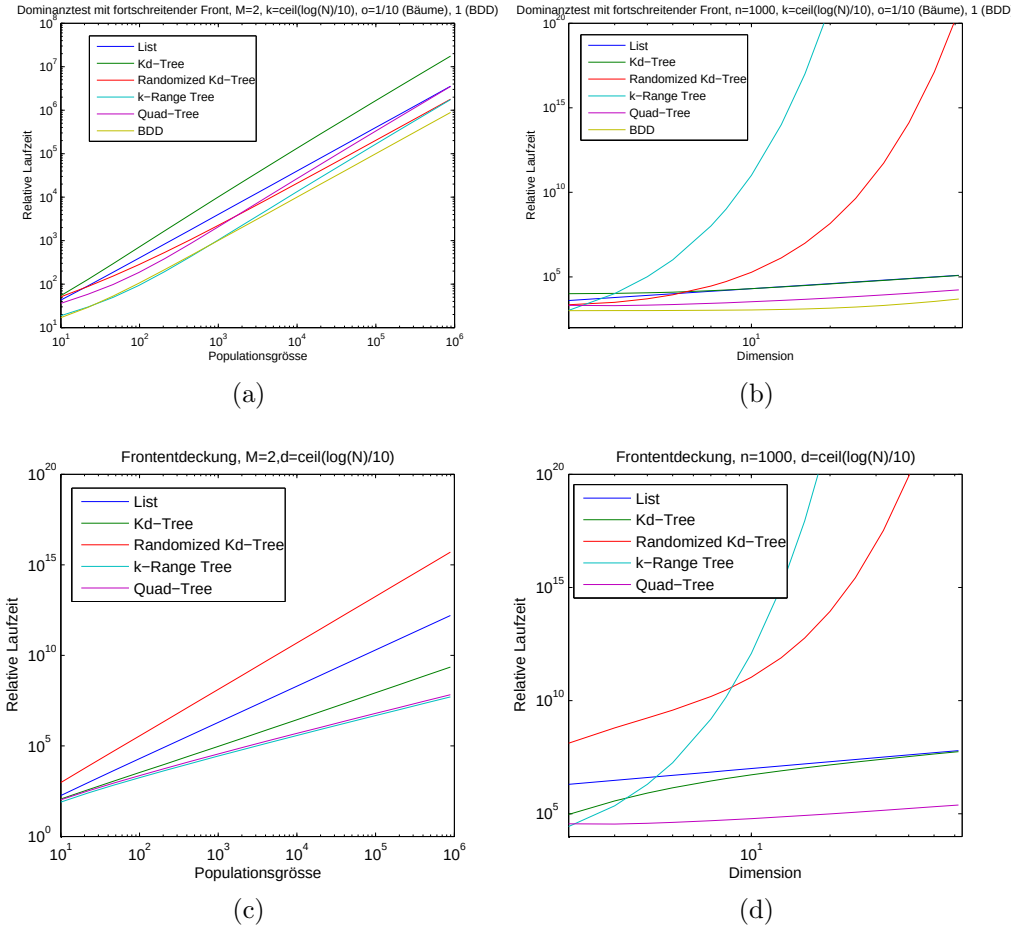
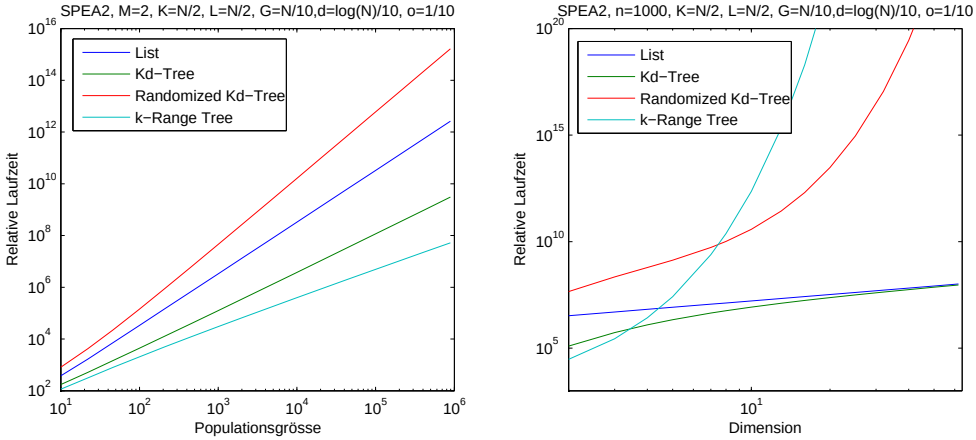


Abbildung 18: Oben: Dominanztest mit fortschreitender Front. Unten: Frontentdeckung.

Kd-Trees

Die Wartungskosten von Kd-Trees bewegen sich im Rahmen von $O(NlogN)$, unabhängig von der Dimension. Die Kosten für Dominanzüberprüfungen sind für M im Limes identisch mit denen von Listen; die obere Schranke der erwarteten Komplexität von randomisierten - respektive zufälligen - Kd-Trees wächst exponentiell in der Anzahl Dimensionen. Allerdings gilt die angegebene, obere Schranke für zufällige, *geschlossene* Abfrageräume.



(a) (b)

Abbildung 19: Variante SPEA2.

Zufällige Kd-Trees scheinen sich nicht für höhere Dimensionen zu eignen. Optimierte Kd-Trees eignen sich gegenüber Listen besser für Anwendungsfälle (z.B. SPEA2, Monte Carlo Variante 2), welche den Schwerpunkt auf Dominanzüberprüfungen setzen und nicht auf Wartungsoperationen, nicht so für z.B. Monte Carlo Variante 1.
Da Evolutionäre Algorithmen randomisierte Verfahren sind, i.e. im Prinzip zufällige Punkte generieren, kann angenommen werden, dass Kd-Trees, welche nicht optimiert werden (Optimierung entfällt), sich analog zu zufälligen Kd-Trees verhalten - respektive kann ein echt-randomisierter Kd-Tree ohne zusätzliche erwartete Kosten (siehe [14]) verwendet werden. Das Laufzeitverhalten bzgl. Dominanzüberprüfungen sollte für diese zufällige Kd-Trees weiter untersucht werden.

K-Range Trees

Die Kosten von Dominanzüberprüfungen sowie Wartungsoperationen von K-Range Trees sind exponentiell in M, was neben dem benötigten Speicher ($O(MN + (logN)^{M-1})$), ein wesentlicher Nachteil von K-Range Trees ist. Denn für niedrige Dimensionen ($M < 5$) verfügt der K-Range Tree über die generell optimalste Laufzeit (insb. Monte Carlo Variante 2). Für Kontexte mit einer grossen Anzahl Wartungsoperationen (Monte Carlo Variante 1), sind sie allerdings weniger geeignet.

Quad-Trees

Die Wartungskosten von Quad-Trees belaufen sich auf asymptotisch ähnliche Werte ($O(NlogN)$) wie bei gewöhnlichen Suchbäumen. Quad-Trees zeichnen sich aus durch eine optimal M-logarithmische Tiefe, was vorallem in höheren Dimensionen von Vorteil ist. Da jedoch die Anzahl Kinder eines Knoten exponentiell in M wächst, ist von praktischer Seite her die Speicherung der Kinder in einer Liste verschwenderisch, so dass besser eine dynamische Struktur (z.B. Suchbaum) verwendet wird; die asymptotische Komplexität des Dominanztests wandelt sich von $O(Mlog_M N)$ zu $O(MlogMlog_M N)$. Quad-Trees eignen sich für Kontexte, welche Dominanztests (fortschreitende Front, Frontentdeckung) ausführen, allerdings verringert sich der Nutzen für Anwendungsfälle mit grossen Populationen und grosser Anzahl von Wartungsoperationen.

Binary Decision Diagrams

Die Performanz von Binary Decision Diagrams (BDD), hängt stark von der Update- und einhergehenden Optimierungsfrequenz ab. Die lediglich in M linearen Kosten des Dominanztests jedoch entsprechen dem bisherigen Optimum. BDDs eignen sich allerdings nur für das Testen auf Dominanz, da keine Informationen bzgl. der konstituierenden Punktmenge vorhanden bleibt (i.e. das Identifizieren von dominierenden, respektive dominierten Punkten unmöglich ist). Wie M. Lukasiewycz et al. [21] erläutern eignen sich BDDs als Hilfsstruktur, z.B. in Kombination mit Listen. Der Einsatz in Evolutionären Algorithmen für die Mehrzieloptimierung mit Elitearchiven (z.B. PAES, SPEA2) ist somit denkbar.

4.3.2 Qualitative Aussagen

In der Folge werden für die Listen, Dominated- und Non-Dominated-Trees sowie meine eigene Datenstruktur qualitative Aussagen gemacht, welche das Laufzeitverhalten besser zu fassen suchen.

Hierzu werden die in **??** vorgestellten nicht-dominierten Punktmodelle und Stichprobenräume herangezogen.

Im Anschluss sei noch ein Wort gesagt bzgl. der Parallelisierung von Datenstrukturen für die Dominanzüberprüfung.

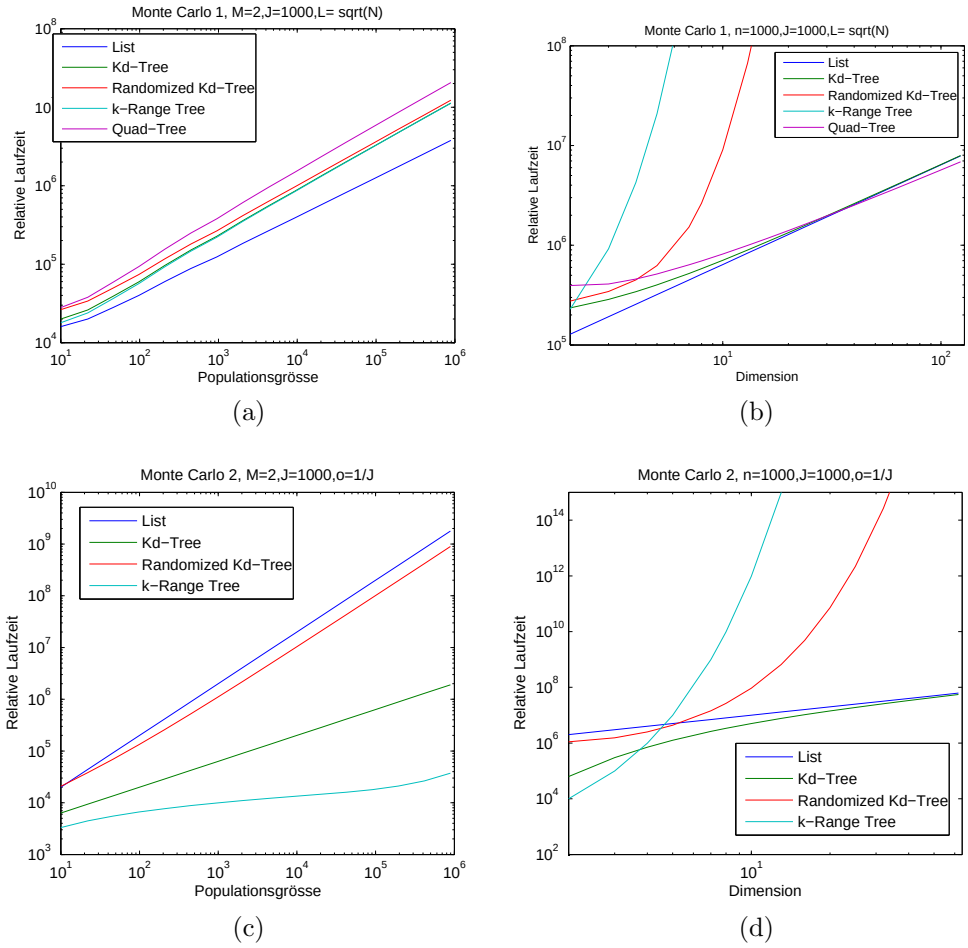


Abbildung 20: Monte Carlo Verfahren, Variante 1 (oben), Variante 2 (unten).

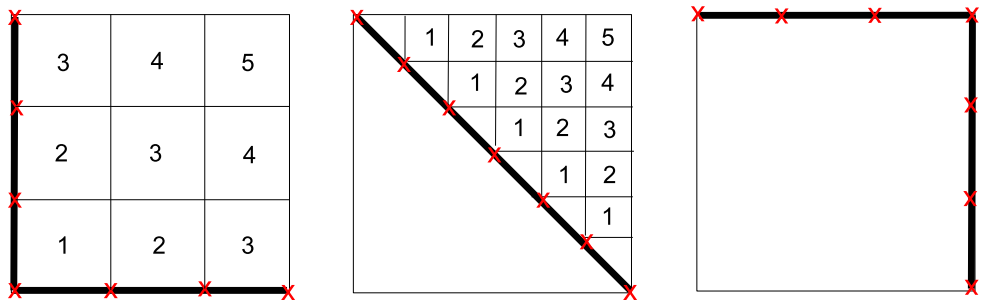
Listen

Die Worst-Case Laufzeiten von Dominanzüberprüfungen auf Listen werden folgend genauer betrachtet. Es sei N die Grösse der Punktmenge P in M Dimensionen.

Soll für einen Abfragepunkt die Anzahl dominierter Punkte festgestellt, oder die dominierten Punkte identifiziert werden, so muss die gesamte Punktmenge auf Dominanz getestet werden, i.e. die Laufzeit beträgt trivialerweise $O(MN)$. Als Folge sind Listen für grosse N nicht optimal für diese Operationen.

Soll ein Abfragepunkt p auf Dominanz getestet werden, so lässt sich generell die benötigte Anzahl von Vergleichen $c(\delta)$ (δ die Anzahl dominierender Punkte) gegen oben beschränken. Der intuitive Ansatz um Dominanztests auf

Listen durchzuführen, überprüft iterativ für jeden Punkt $l \in P$, ob $l \prec p$, und bricht sobald als möglich ab. Es gilt $max(c(\delta)) = N - \delta$, insb. $max(c(\delta)) = 0$ wenn p nicht dominiert ist. Es liegt also nahe, δ etwas genauer zu untersuchen.



(a) (b) (c)

Abbildung 21: Gleichmässige Beispielfronten der Grösse 7. Dargestellt sind generelle Frontform (schwarzer Balken), Punkte (rot), Stichprobenbox, Unterteilung des Stichprobenraumes in Bereiche mit gleicher Anzahl dominierender Punkte.

Es sei $M = 2$ und P eine nicht-dominierte Punktmenge der Grösse $N = 2 \cdot k + 1$ ($k \in \mathbb{N}$) die Punkte gleichmässig auf der Front verteilt. Es lässt sich somit die erwartete Anzahl durchzuführender Dominanzüberprüfungen bestimmen. Seien zusätzlich die drei Frontformen approximiert auf jeweils zwei Kanten der Box, respektive eine Diagonale der Box. Es ergibt sich somit für die Frontenformen:

Stark dominierend. $E_2[c(\delta)] \approx \frac{N}{2}$

Mässig dominierend. $E_1[c(\delta)] = \frac{1}{(N-2)^2} \cdot (N \cdot (N-1)) + \sum_{i=1}^{N-2}(i \cdot N + i^2))$

Schwach dominierend. $E_3[c(\delta)] = N$

Da die betrachtete Punktmenge nicht-dominiert ist, kann allerdings Nicht-Dominanz ausgeschlossen werden, wenn p dominiert *oder* dominiert wird. Es bietet sich eine zweite Variante für den Dominanztest auf Listen an: jeder Punkt in P wird iterativ wie besagt mit p verglichen. Die relativen Kosten $k, 1 \leq k \leq 2$ für einen solchen Vergleich entsprechen nicht notwendigerweise dem Doppelten eines einfachen Vergleichs. Die neue erwartete Anzahl von Vergleichen berechnet sich zu:

Stark dominierend. $E_2[c(\delta)] \approx \frac{N}{2}$

Mässig dominierend. $E_1[c(\delta)] = \frac{1}{(N-2)^2} \cdot (N \cdot (N-1)) + 2 \cdot \sum_{i=1}^{N-2} i^2)$

Schwach dominierend. $E_3[c(\delta)] \approx \frac{N}{2}$

Die zweite Variante benötigt eine geringere Anzahl an Vergleichen. Werden die Anzahl Vergleiche gegenübergestellt und mit den zur ersten Vergleichsvariante relativen Kosten gewichtet, so lässt sich berechnen, dass für $k \lesssim 1.4$ die zweite Variante generell günstiger ist.

Ob und wie diese Kosten in Theorie oder Praxis erreichbar sind, wäre Subjekt weiterer Betrachtungen.

Die Kostengrenze ist auch in beliebigen Dimensionen interessant; es sei diesbezüglich ein Ansatz erwähnt:

Ausgegangen wird von einer nicht-dominierten Front in Form einer $(M-1)$-dimensionalen Hyperkugel, welche den Boden von zwei voneinander weggerichteten M-dimensionalen Kegeln mit Öffnungswinkel α und Höhe h, bildet. Diese seien im Speziellen so ausgerichtet, dass die Spitze des einen Kegels alle Punkte dominiert, die Spitze des anderen von allen Punkten dominiert wird. Die Kegel definieren zusammen den Stichprobenraum.

Der von einem Punkt p dominierte Hyperraum sei approximiert durch den kleinsten Kegel, welcher den dominierten Hyperraum umfasst; die Menge $A(p)$ der von p dominierten Punkte entspricht dann dem von dem Kegel umfassten Teil der Hyperkugel mit Gesamtoberfläche A_{tot}. Sind die Punkte der nicht-dominierten Front gleichmässig verteilt, ergeben sich für die erwarteten, maximalen Anzahlen auszuführender Vergleiche für die zwei Varianten der Überprüfung approximativ (N die Populationsgrösse):

1. $N \cdot (\int_0^h (A(\delta) \cdot \frac{A_{tot}-A(\delta)}{A_{tot}}) d\delta + \int_0^h A(\delta) d\delta)$
2. $2 \cdot N \cdot \int_0^h (A(\delta) \cdot \frac{A_{tot}-A(\delta)}{A_{tot}} d\delta)$

Eine zusätzliche Verbesserung von Listen wäre eine günstigere Traversierungsstruktur; an welchem Punkt die Datenstruktur sich den Quad-Trees nähert.

Für den adaptiven Stichprobenraum eignet sich die zweite Variante weniger, da lediglich einer von fünf Punkten die Front dominiert, d.h. die Wahrscheinlichkeit gering ist, dass Nutzen entstehen würde.

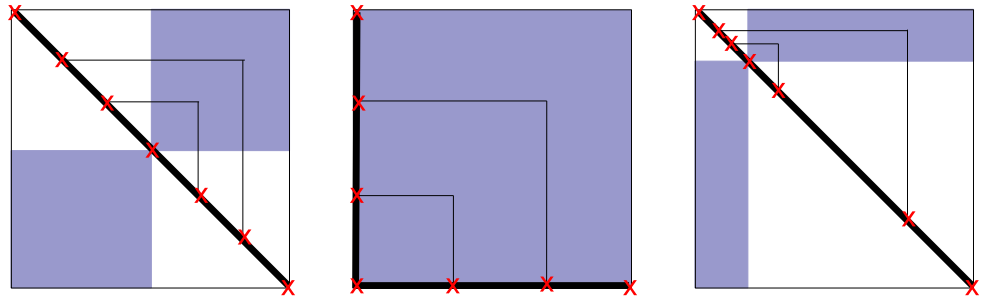
Kommen zur Punktmenge dominierte Punkte hinzu ist der gemittelte dominierte Hyperraum geringer, wodurch die erwartete Anzahl von Vergleichen ansteigt. Listen sind in der Folge für die Dominanzüberprüfung nach

vorgeschlagener Monte Carlo Variante 1 nicht optimal, allerdings mit den Null-Wartungskosten noch eine Option.

Dominated und Non-Dominated Trees

Dominated und Non-Dominated Trees (DND Trees) wie in [16] beschrieben zu implementieren, führt im Verlgeich zu den restlichen Datenstrukturen zu relativ hoher Komplexität, was sie weniger zugänglich macht.

Seien wieder M-dimensionale, nicht-dominierte Fronten P der Grösse N betrachtet mit Box R als Stichprobenraum. Die Dominanzfrage lässt sich trivial entscheiden, wenn der Abfragepunkt a vergleichbar ist mit dem kleinsten, zusammengesetzten Punkt p_{min} (siehe Abb. 22). D.h. das vergleichbare Hypervolumen $V(p_{min})$ in R ist optimalerweise gross.



(a) (b) (c)

Abbildung 22: Gleichmässige Beispielfronten der Grösse 7. Dargestellt sind generelle Frontform (schwarzer Balken), Punkte (rot), Stichprobenbox, Strukturierung des DND Tree.

Für $M = 2$ und eine gleichmässig verteilte Front ergibt sich, dass ungefähr mindestens $\frac{1}{2}$ des Stichprobenraumes mit p_{min} vergleichbar ist (siehe Abb. 22), bzw. für generelles M, dass ungefähr mindestens $\frac{2}{2^M}$ des Hypervolumens zu $V(p_{min})$ gehört, und mit stärkerer Konvexität, respektive Konkavität wächst.

Ist jedoch die Population stark asymmetrisch auf der Front verteilt, respektive gibt es starke Ausreisser (siehe Abb. 22c), so verhalten sich DND Trees ähnlich zu Listen, haben jedoch mehr Nebenkosten. Da moderne Evolutionäre Algorithmen für die Mehrzieloptimierung eine gleichmässige Verteilung auf Elitefronten fördern, dürften sich DND Trees aber relativ gutmütig verhalten.

Eigene Datenstruktur

Der Aufbau der Struktur ist vergleichsweise (z.B. gegenüber Dominated und Non-Dominated Trees) einfach gehalten; es wird lediglich eine Listensuchstruktur für eindimensionale Schlüssel benötigt. Ein möglicher Nachteil ist die eventuell nicht billige Berechnung der geometrischen Kriterien: auch wenn die Komplexität der Berechnung linear in der Anzahl Dimensionen ist, so ist bei geradliniger Konstruktion das Berechnen der Quadratwurzel nötig, was in der Praxis i.d.R. eine kostspielige Operation darstellt. Insofern dürfte es sich anbieten, die Kriterien umzuformen oder abzuschwächen (z.B. zu approximieren). Decken sich Population und Menge der Abfragepunkte, entstehen für die Berechnung der Eigenschaften der Abfragepunkte allerdings keine weiteren Kosten.

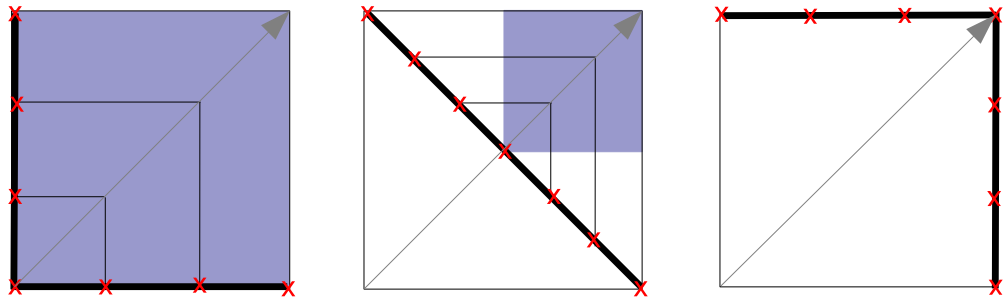
Da die Effizienz der Datenstruktur von den geometrischen Eigenschaften (siehe 3.1) der Punkte abhängt, werden diese für die verschiedenen Frontenformen näher betrachtet.

Für ausgedehnte, gleichmässige Fronten wird das erste Kriterium (Vergleich minimale, maximale Komponente) tendenziell selten erfüllt sein, so dass durch die Anwendung unnötiger Mehraufwand entsteht. Wenn, wie in 3.4 besprochen, für Dominanztests lediglich die kleinste maximale Komponente der Menge verglichen wird, kann der Kriteriumstest bei der Listentraversierung weggelassen werden; die stark Frontenform-abhängige Menge der somit trivial entschiedenen Dominanztests ist in Abbildung 23a blau eingezeichnet.

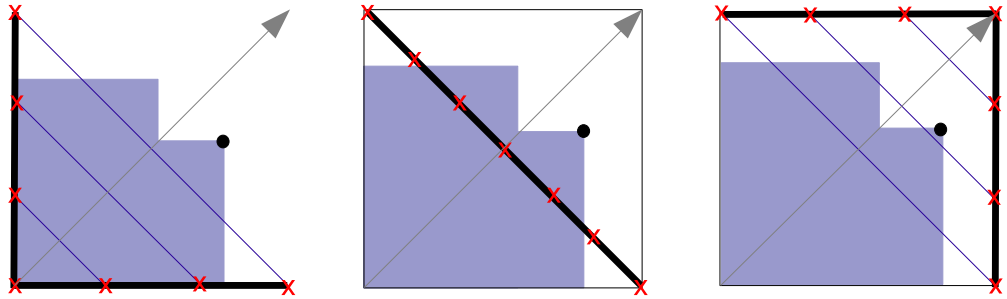
Die alleinige Verwendung des ersten Ausschlusskriteriums in Kombination mit einer Listenstruktur ist unglücklich, da dieses nur die Laufzeit verringert, wenn der Abfragepunkt bzgl. der Raumdiagonale $diag_e$ nicht 'hinter' der Punktmenge lokalisiert ist. Dies wird insbesondere für den adaptiven Stichprobenraum (siehe 4.1) der Fall sein; allerdings wird dann das erste Kriterium (siehe oben) tendenziell öfter erfüllt sein. Das Kriterium ist generell nützlicher, wenn die Punktmenge bzgl. Raumdiagonalen ausgedehnter ist, und der Abfragepunkt darin lokalisiert ist.

Die Effektivität des zweiten Ausschlusskriteriums ist stark abhängig vom Stichprobenraum; der Nutzen scheint für die Frontenformen umgekehrt: während für eine schwach dominierende Front mit Box als Stichprobenraum (siehe Abb. 23b, links) das Kriterium tendenziell einsparend wirken wird, ist das Kriterium für den adaptiven Stichprobenraum voraussichtlich teurer. Für die sonstigen Frontenformen sollte das Kriterium besser abschneiden, wenn die Abfragepunkte näher an der Front liegen, was bei dem adaptiven Stichprobenraum vermehrt der Fall sein wird. Zudem wird das Kriterium in vielen Dimensionen zu wenig präzis sein, vorallem da mit zunehmender Dimension

die Reduzierung auf lediglich zwei Bezugssysteme zu wenig restriktiv ist.



(a)



(b)

Abbildung 23: Geometrische Eigenschaften für verschiedene Frontenformen (schwarzer Balken), sieben gleichmässig verteilte Beispielpunkte (rot), umfassende Box. (a) Maximale Komponenten sind eingezeichnet, und der von der kleinsten maximalen Komponente durch das erste Kriterium dominierte Bereich (blau). (b) Projektion auf die Diagonale, und nicht ausschliessbarer Bereich (blau) nach Anwendung der Ausschlusskriterien.

Die Kriterien haben Potenzial sich gegenseitig zu ergänzen, scheinen aber in der vorgeschlagenen Umsetzung zu wenig aufeinander abgestimmt zu sein (insb. die beiden Ausschlusskriterien). Der durch die nicht-optimale Nutzung der Kriterien entstehende Overhead dürfte vorallem für das Zählen und Aufzählen von dominierten Punkten einen wesentlichen Nachteil darstellen.

Parallelisierung

Da mit der fortschreitenden Entwicklung superskalarer Prozessorarchitekturen die praktische Bedeutung von parallelisierter Berechnung zunimmt, sei kurz darauf eingegangen was dies für Datenstrukturen für die Domi-

nanzüberprüfung bedeuten kann. Im Speziellen seien programmierbare *Graphics Processing Units* (GPUs) erwähnt, welche in ihrer simpelsten Form ihren Weg als Grafikkarte in den typischen Haushalt gefunden haben.

GPUs führen auf (potenziell) grossen Datenmengen in Matrixform auf hoch-parallelisierte Weise identische Operationen aus. Die Programmierbarkeit der GPUs erlaubt, anstatt z.B. der typischen Anwendung als Renderpipeline, ihren Einsatz für andere Berechnungen, z.B. von Dominanzüberprüfungen.

Das Aufbrechen einer Punktmenge in mehrere Mengen ermöglicht die simultane Bearbeitung dieser zu erhöhten Wartungskosten. Von den hier betrachteten Datenstrukturen eignen sich im Besonderen Listen aufgrund ihrer einfachen Struktur, i.e. der Matrixform und der bereits minimalen Wartungskosten, für den Einsatz auf GPUs.

Die Entscheidung nicht-optimale Datenstrukturen einzusetzen, um den Verlust an Effizienz an anderer Stelle möglicherweise mehr als aufzuwägen, ist jedoch rein praktischer Natur.

4.4 Experimente

Für die Experimente wurden vier Datenstrukturen implementiert, auf welchen alle Typen von Dominanzüberprüfungen durchführbar sind: Listen als Standardstruktur, Kd-Trees als Strukturen für die geometrische Suche, sowie Dominated- und Non-Dominated Trees (DND Trees) und die eigene Datenstruktur, welche für Dominanzüberprüfungen entwickelt wurden. Zudem operieren die Datenstrukturen in einem nicht an den Anwendungsfall adaptierten Zustand; Listen verwenden das einfache Muster zum Dominanztesten, die DND Trees, sowie die eigene Datenstruktur führen parallel je zwei Listen (zum Testen sowie (Auf-)Zählen), was zu erhöhter, schlimmstenfalls doppelter Wartungsarbeit führt.

Die Listen sind in Form eines Arrays (nicht Linked Lists) implementiert. Die Kd-Trees wurden mit einem nicht-optimalen Optimierungsalgorithmus ausgestattet (Laufzeit $O(N^2 logN)$ anstatt $O(NlogN)$). Für die geordneten Listen der DND Trees, und der eigenen Datenstruktur wurde eine externe Skiplist-Implementation herangezogen.

Untersucht wurden zwei Aspekte: (1) Performanz für Aufbau und Dominanztesten in Abhängigkeit der Anzahl Abfragepunkte aus einer Box (2) für verschiedene Stichprobenräume (Adaptiv, und insb. Box), Punktmengen mit unterschiedlicher Form die generelle Performanz von Dominanztests und Zählen bei grosser Anzahl $J = 50.000$ von Abfragepunkten. Für alle Tests wurde zudem die Populationsgrösse bei fixer Dimension (zwei), sowie die Dimension bei fixer Populationsgrösse (1000) variiert.

Zur Generierung der Fronten wurde das in 4.1.2 vorgestellte Model herangezogen; für die Frontformen wurde $\alpha \in \{\pi * (\frac{1}{4} + \frac{-1}{64}), 0, \pi * (\frac{-1}{4} + \frac{1}{64})\}$ (stark, mässig, schwach dominierend), und $\Delta = \mathcal{U}(0, 1)$ gewählt.

Es folgt die Betrachtung der Verhalten der einzelnen Datenstrukturen.

Listen

Aus Abbildung 24 ist die Linearität in J, N und M ersichtlich. Für gleichbleibendes N und $M > 8$ flacht jedoch die die Kurve extrem ab; ein Blick in die Testdaten zeigt, dass für wachsendes M die Anzahl effektiv durchgeführter Vergleiche ansteigt. Als Ursache wird vermutet, dass mit zunehmender Dimension das dominierte Hypervolumen abnimmt, und somit mehr Stichproben nicht-dominiert sind, i.e. dass mehr Vegleiche benötigt werden, um einen dominierenden Punkt auszumachen. Derselbe Effekt lässt sich auch für die anderen Datenstrukturen beobachten.
Für den adaptiven Stichprobenraum ergeben sich für die verschiedenen Frontenformen keine wesentlichen Unterschiede.
Für die Box als Stichprobenraum zeigen sich für die Frontenformen Unterschiede in der generellen Effizienz (1)(Abhängigkeit in N) und der Kurvenform (2)(Abhängigkeit in M). (1) ergibt sich aus dem stark veränderten, dominierten Hypervolumen. (2) äussert sich so, dass sich Listen für die stärker dominierenden Fronten in tieferen Dimensionen besser verhalten; bei allen Frontenformen pendelt sich die benötigte Zeit ein, der bereits besprochene Effekt wird vermutet.
Es bestätigt sich, dass die Effizienz von Listen unabhängig von der Frontenform für das Zählen dominierter Punkte statisch ist.

Wenn es darum geht schnell eine sehr grosse Population aufzubauen, und darauf eine verhältnismässig geringe Anzahl von Dominanztests durchzuführen (i.e. $N > 10^4, N \gg J$), sind Listen eine geeignete Wahl; z.B. für die Frontentdeckung ($N = J$) scheinen sie dadurch geeignet. Wird nur die Effizienz von Dominanzüberprüfungen betrachtet, so sind Listen allerdings nicht die optimale Wahl, so dass Listen nicht für die Monte Carlo Verfahren (insb. mit grossen J) geeignet scheinen.

Kd-Trees

Kd-Trees erweisen sich in Abbildung 24 für $J \gg N$ als wesentlich effizienter (ca. 10-fach für $J = 10^5$) als Listen. Für wachsendes N machen sich die Aufbaukosten von $O(N^2 logN)$ deutlich bemerkbar. Für steigendes M pendelt sich die Laufzeit schneller ein als bei Listen.
Für den adaptiven Stichprobenraum ergeben sich keine wesentlichen Unterschiede für die Frontenformen. Kd-Trees sind die weitaus effizientesten Datenstrukturen (biszu 100-fach für grosse N); dies zeigt sich deutlichsten für grosse N und kleine M. Für wachsendes M nähern sich die Laufzeiten derjenigen der anderen Datenstrukturen; spiegelt somit die Abfragekomplexität von $O(MN^{1-\frac{1}{M}})$ wieder.
Die relative Effizienz tritt für die Box als Stichprobenraum stärker hervor, und ist interessanterweise für die mässig dominierende Front am grössten. Der Einfluss der Dimension ist weniger stark als bei den restlichen Datenstrukturen.
Für das Zählen dominierter Punkte ergibt sich, dass Kd-Trees für die Frontenformen ähnliches Verhalten an den Tag legen, aber für weniger dominierende Fronten der Unterschied der Effizienz zu den restlichen Datenstrukturen geringer wird. Dies führe ich darauf zurück, dass schwächer dominierende Fronten stärker dominiert werden bzgl. des Stichprobenraumes, und somit mehr Punkte gezählt werden müssen.

Optimierte Kd-Trees zeigen generell ein sehr gutes Verhalten. Die Grösse der Punktmenge beeinflusst das Verhalten für schnellen Aufbau mit vergleichbarer Anzahl Dominanztests wesentlich; Kd-Trees sind insofern für kleine J bedingt zu empfehlen für z.B. Monte Carlo Variante 1, die Eignung für die Frontentdeckung mit sehr grossen Mengen ist allerdings nicht direkt abschätzbar, da die Wartungsoperationen (Einfügen, Entfernen) kontinuierlich stattfinden und nicht wie in der Testumgebung kontrolliert zum optimierten Aufbau aufgerufen werden (i.e. die Optimierung mit $O(N^2 logN)$ fällt weg, der Baum degeneriert ggf.). Auf die Dominanzüberprüfungen reduziert, sind die Kd-Trees von den betrachteten Datenstrukturen optimal, also für Monte Carlo Verfahren mit grosser Anzahl Überprüfungen zu empfehlen.

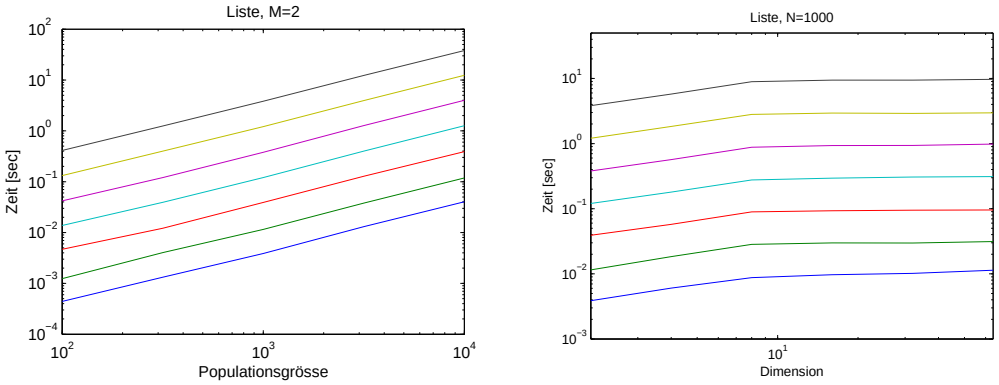
Dominated und Non-Dominated Trees

Nach Abbildung 25 haben DND Trees für $J \gg N$ und M klein eine Listen ähnliche Performanz. Für wachsende N, und M dominiert die Aufbaukomplexität von $O(MN^2 logN)$.
DND Trees wirken für Dominanztests für alle Frontenformen relativ statisch, was das stärker von der Frontenform abhängige Verhalten der anderen Datenstrukturen konstrastiert.
Für den adaptiven Stichprobenraum zeigt sich deutlich eine zu Listen relativ bessere Performanz. Für die Box als Stichprobenraum wird ein geringerer Unterschied sichtbar, welcher für die Frontenformen variert (bei der stark dominierenden Front in wenigen Dimensionen sind Listen sogar besser!).
Für das Zählen dominierter Punkte bei stark dominierender Front in wenigen Dimensionen macht sich der Ausschlussmechanismus der DND Trees bemerkbar, i.e. DND Trees sind um einen Faktor $10^{0.5}$ schneller als Listen. Sie scheinen aber bei den sonstigen Konfigurationen zunehmend zu Listen mit Mehraufwand zu degenerieren.

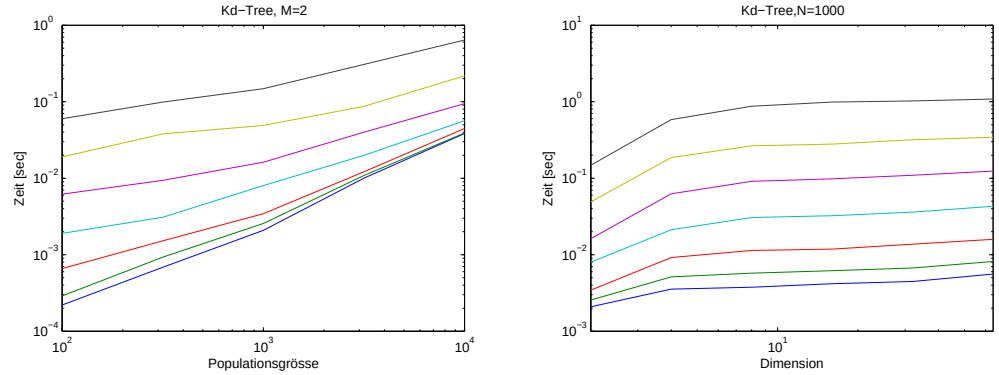
DND Trees sind eindeutig die teuersten Datenstrukturen bzgl. Unterhalt; für grosse N und bei geringer Anzahl Dominanztests dominieren die Kosten der Wartungsoperationen. Nicht beachtet wurde dabei der Unterhalt einer sich stetig verändernden Punktmenge, was zu Degeneration führt; J. Fieldsend et al.[16] empfehlen zur Optimierung entweder einen Algorithmus mit maximaler Komplexität $O(NM - N)$, der die Struktur aber nicht vollständig optimiert, oder den Wiederaufbau der Datenstruktur, was für grosse N zu teuer ist. DND Trees weisen generell aber für Dominanztests eine höhere Effizienz auf als Listen; der Verwendung zum Unterhalt eines Elite Archivs (i.e. Dominanztest mit fortschreitender Front) ist denkbar, was zudem der von J. Fieldsend et al. empfohlene Einsatzbereich ist. Nicht geeignet scheinen DND Trees für das Zählen dominierter Punkte, da sie zu Listen mit Mehraufwand degenerieren können.

Eigene Datenstruktur
Die eigene Datenstruktur scheint im ersten Test (Aufbau und varierte Anzahl Abfragepunkte) vergleichbar mit, aber geringfügig besser als Dominated und Non-Dominated Trees (DND Trees), sowie Listen zu sein, insbesondere hat die Dimension gegenüber den DND Trees praktisch keinen Einfluss.
Für die Dominanztests pendelt die Laufzeit der eigenen Datenstruktur um diejenige der DND Trees. Für mässig dominierende Fronten zeigt sich kein grosser Unterschied im Verhalten. Entgegengesetztes Verhalten äussert sich vorallem für die schwach, respektive stark dominierende Front und Stichprobenräume in vielen Dimensionen: bei adaptivem Stichprobenraum, stark dominierender Front ist der eigene Ansatz besser (Faktor ca. $10^{0.5}$) und bei schwach dominierender Front sind DND Trees effizienter (Faktor ca. $10^{0.2}$); mit der Box als Stichprobenraum ist dieses Verhalten umgekehrt.
Zum Zählen dominierter Punkte ist die eigene Datenstruktur nicht geeignet; sie schneidet für alle Konfigurationen schlechter ab als Listen.
Die Datenstruktur konkurriert die DND Trees, welche sich soweit für den Einsatz bei Dominanztest mit fortschreitender Front (also z.B. das Führen eines Elitearchivs) eignen könnten. Sie stellt allerdings keine Konkurrenz dar für Kd-Trees generell und im Speziellen für das (Auf-)Zählen dominierter Punkte.

Die Skiplist, welche für die Dominated und Non-Dominated Trees, sowie die eigene Datenstruktur eingesetzt wurde, ist nicht notwendigerweise optimal. Die Auswirkung einer angepassten Listenstruktur auf die Performanz wäre von gewissem Interesse.

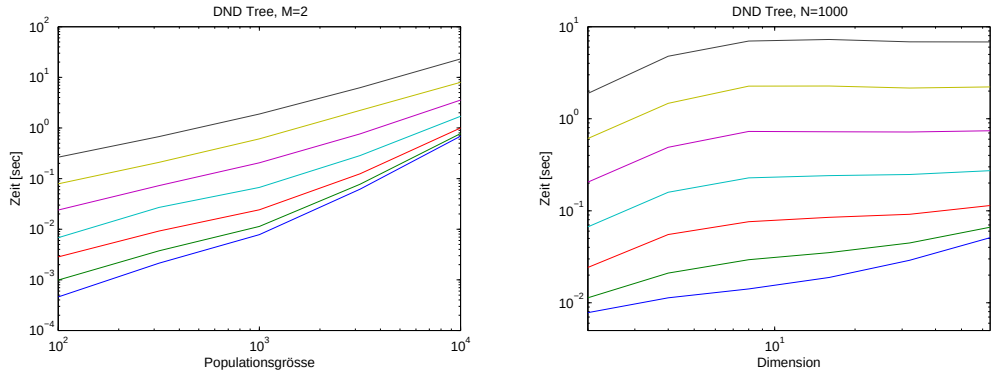


(a)

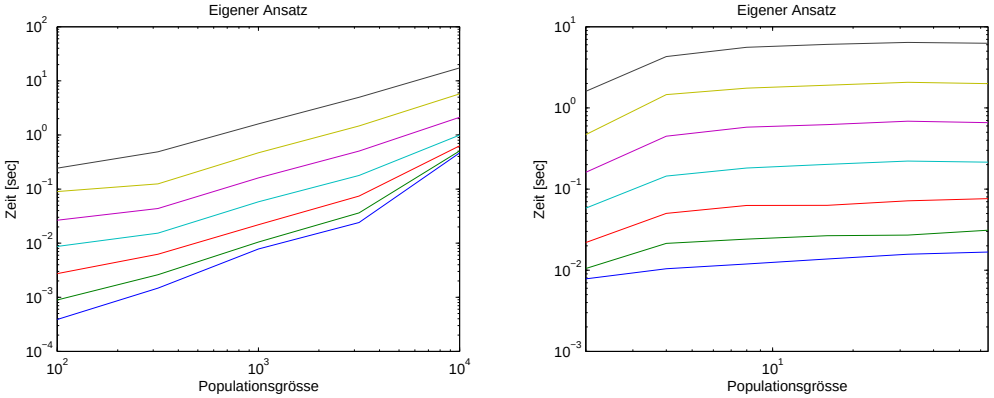


(b)

Abbildung 24: Aufbau und variierte Anzahl Dominanztests: 100 (blau), 317 (grün), 1000 (rot), 3163 (türkis), 10000 (magenta), 31630 (gelb), 100000 (grau). Mässig dominierende Front. Listen und Kd-Trees.



(a)



(b)

Abbildung 25: Aufbau und variierte Anzahl Dominanztests: 100 (blau), 317 (grün), 1000 (rot), 3163 (türkis), 10000 (magenta), 31630 (gelb), 100000 (grau). Mässig dominierende Front. Dominated- und Non-Dominated Trees, eigener Ansatz.

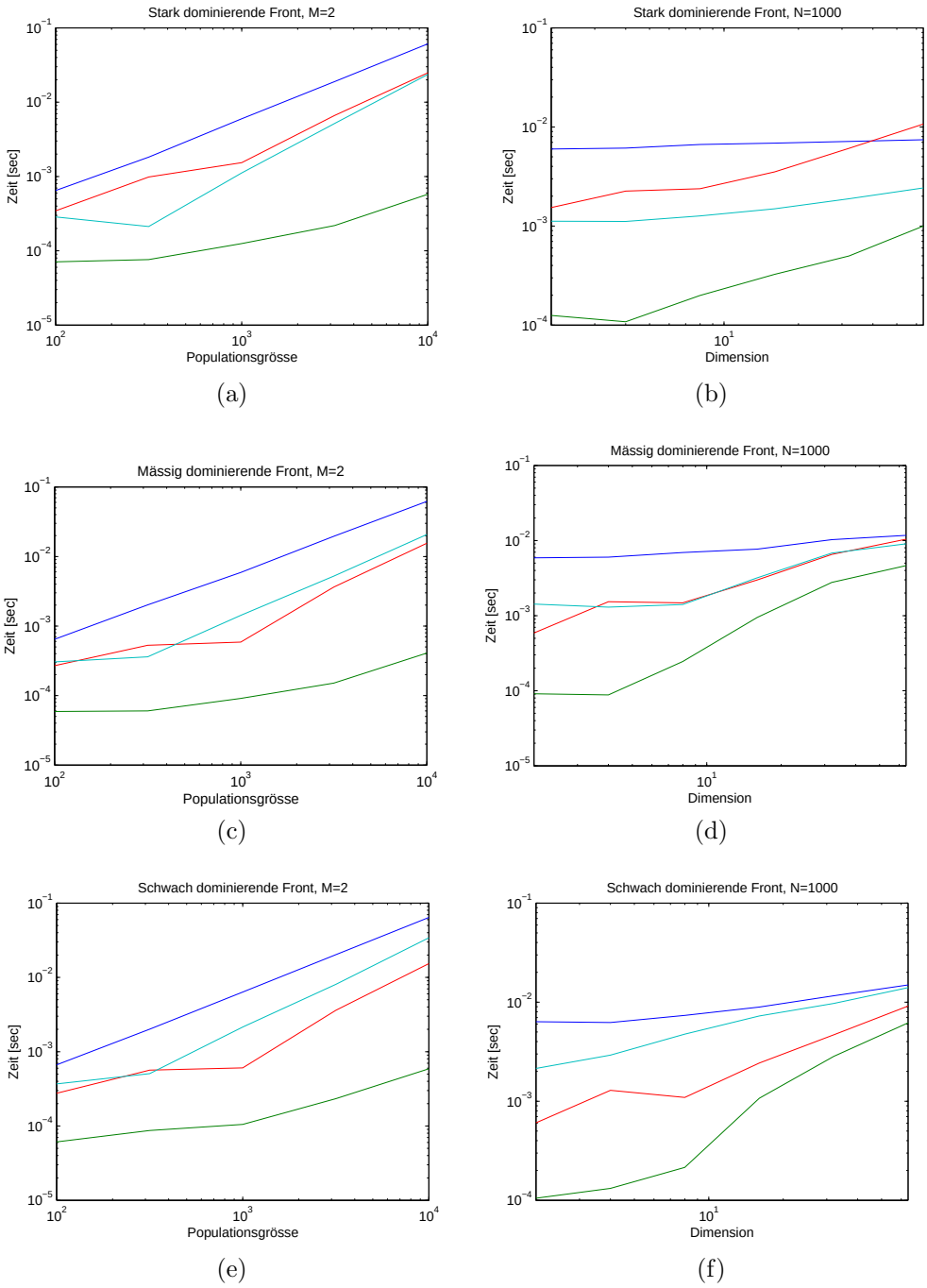


Abbildung 26: Dominanztests mit adaptivem Stichprobenraum für verschiedene Datenstrukturen und Frontenformen. Listen (blau), Kd-Tree (grün), DND Tree (rot), eigener Ansatz (türkis).

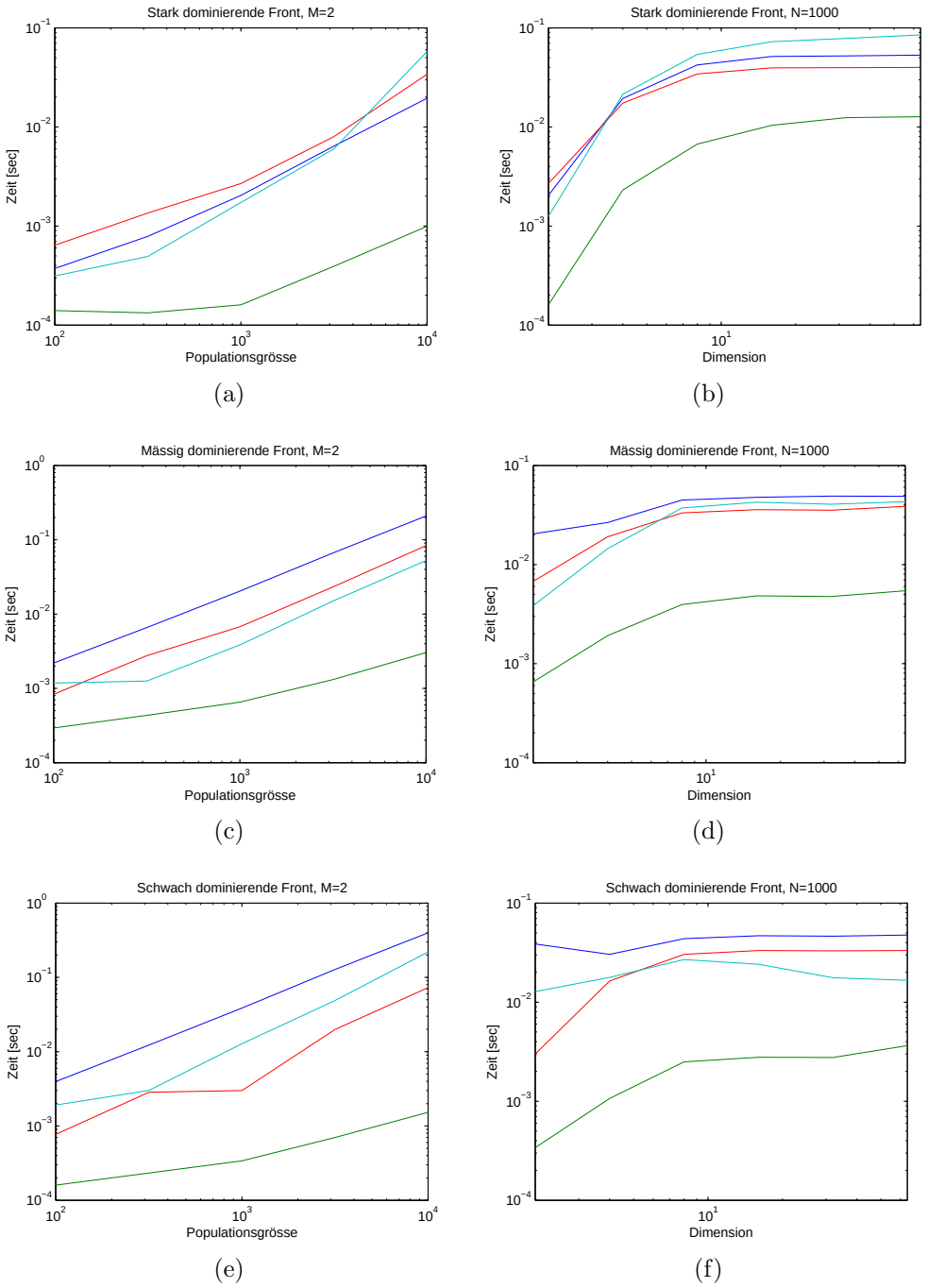


Abbildung 27: Dominanztests mit Box als Stichprobenraum für verschiedene Datenstrukturen und Frontenformen. Listen (blau), Kd-Tree (grün), DND Tree (rot), eigener Ansatz (türkis).

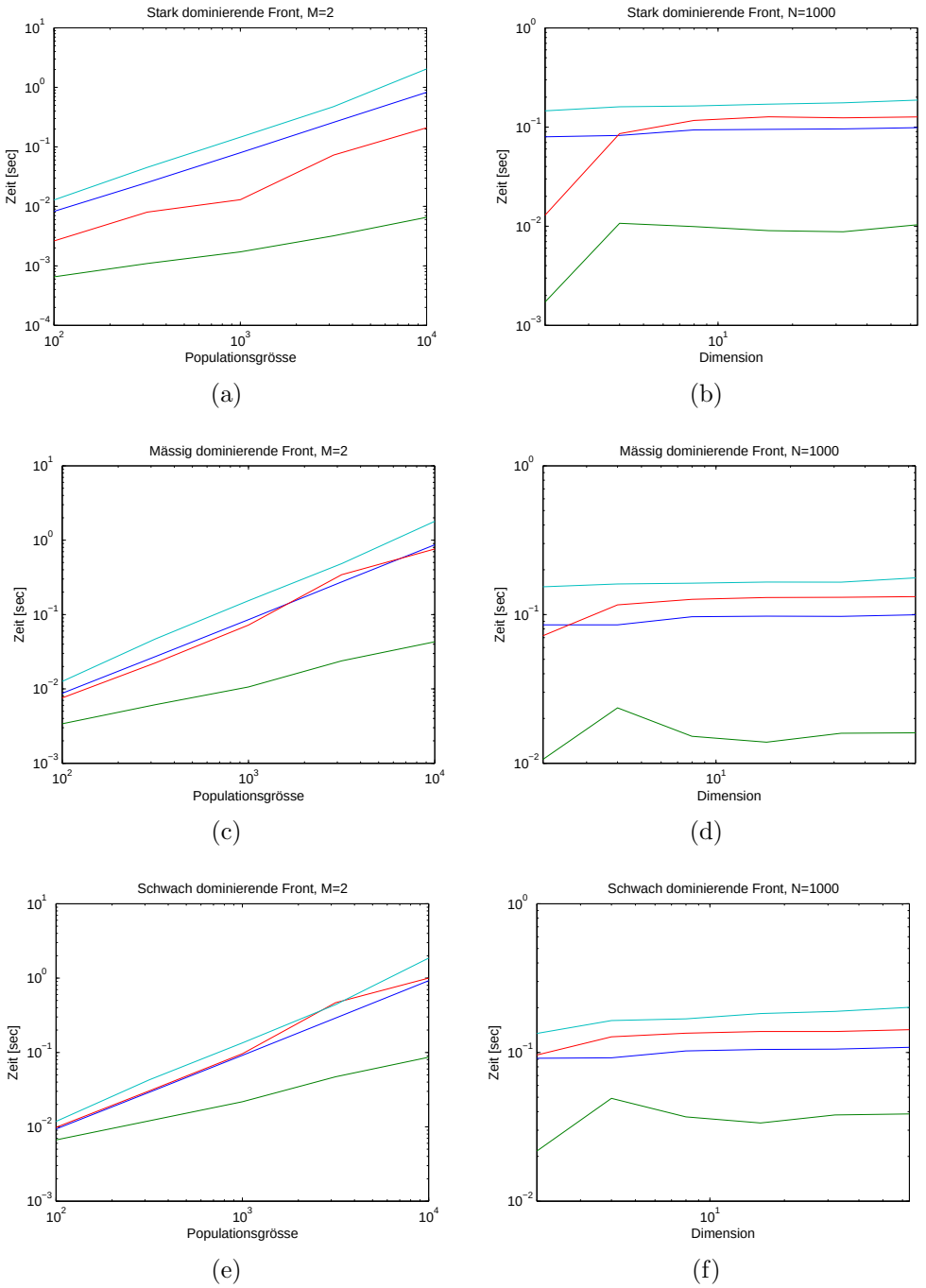


Abbildung 28: Zählen dominierter Punkte mit Box als Stichprobenraum für verschiedene Datenstrukturen und Frontenformen. Listen (blau), Kd-Tree (grün), DND Tree (rot), eigener Ansatz (türkis).

5 Schlussfolgerung und Ausblick

Für Dominanzüberprüfungen existiert eine Vielzahl eingesetzter und insbesondere möglicher Datenstrukturen, wovon allerdings einige bereits durch ihre Struktur eingeschränkt einsetzbar sind.

Eine generelle, theoretische Eignungsabschätzung durchzuführen wird erschwert einserseits dadurch, dass von den Datenstrukturen lediglich empirische Resultate vorliegen, und andererseits da Evolutionäre Algorithmen sehr unterschiedliche Anforderungen an die Datenstrukturen stellen.

Nach einer ersten Sichtung haben sich verschiedene Datenstrukturen als potenzielle Kandidaten für spezifische Anwendungsfälle erwiesen, so z.B. Quad-Trees und Binary Decision Diagrams für Dominanztests in vielen Dimensionen, oder K-Ranges für generelle Probleme in wenigen Dimensionen. Experimente mit einigen generell einsetzbaren Datenstrukturen haben zudem gezeigt, dass Kd-Trees für den unspezifierten Fall weitere Top-Kandidaten sind. Listen als wenig spezifische Datenstruktur bleiben aber eine valable Option für kleine oder sehr grosse Populationen in vielen Dimesionen, insbesondere wenn die Anzahl Dominanzüberprüfungen geringer als oder vergleichbar mit der Populationsgrösse ist.

Eine neue Datenstruktur, welche bisher weniger beachtete Eigenschaften verwendet, kann den spezialisierteren Dominated und Non-Dominated Trees etwa Paroli bieten.

Ausserdem hat sich gezeigt, dass sich gewisse Evolutionäre Algorithmen restrukturieren lassen, um Datenstrukturen für die Dominanzüberprüfung optimaler einzusetzen.

In der Folge sind die in den Experimenten verwendeten Datenstrukturen ausführlicher zu testen; so vorallem nicht-optimierte, oder randomisierte Varianten von Kd-Trees. Weitere Datenstrukturen (z.B. K-Ranges) sind genauer zu untersuchen, und dies insbesondere für spezifischere Anwendungsfälle.

Da sich die theoretische Auseinandersetzung nach wie vor schwer gestaltet, mag ein umfassender empirischer Vergleich eine Alternative darstellen.

Der Einsatz parallelisierter Hardware (z.B. Graphics Processing Units) kann wiederum die Wahl der Datenstruktur beeinflussen, was auch weiter zu betrachten ist.

Die Möglichkeiten von Datenstrukturen scheinen aber noch nicht ausgereizt; können ggf. neue Kriterien eingebracht werden? oder können Verfahren anderer Fachbereiche, etwa der Grafik- oder Datenbankverarbeitung, adaptiert werden?

Appendix

MATLAB Code für Frontenmodell

```
function [Pop] = makePop( N, M, alpha, frontweight, frontdist, D, base )
%   N    Population size
%   M    Dimensions
%   alpha angle between null(ones(1,M)) and outer cone hull
%   frontweight
% weight of specific front w.r.t. N
%   frontdist
% displacement of the front centers
%   D distance distribution (non-negative)
%   base reference point of first fronts

if (nargin<7), base = zeros( 1,M ); end;

nFronts = length( frontweight );
eD = repmat( 1/sqrt(M), 1, M );
alpha = tan(alpha)*eD;

diagx = @(x) eD * (eD * x');
project = @(x) (x - diagx(x));
distx = @(x) norm(project(x),2);

Pop = [];

for nf=1:nFronts
    fw = ceil( N*frontweight(nf) );
    while fw
        d = D();
        vec = project(rand( 1,M ));
        vec = vec/norm(vec);
        new = base + d*(alpha + vec);
        Pop = [Pop; new];
        fw = fw - 1;
    end;
    base = base + frontdist*eD;
end;
```

Literatur

[1] Eiben, A.E., Smith, J.E. *Introduction to Evolutionary Computing*. 1. Edition, Springer, 2003.

[2] K. Deb. *Multi-objective optimization using evolutionary algorithms*. Wiley, Chichester, UK, 2001.

[3] J. Bader, K. Deb, E. Zitzler. *Faster Hypervolume-based Search using Monte Carlo Sampling* in *Conference on Multiple Criteria Decision Making (MCDM 2008)*. Springer, 2008. To Appear.

[4] L. While, P. Hingston, L. Barone, S. Huband. *A Faster Algorithm for Calculating Hypervolume*. In *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 1, pp. 29-38, 2006.

[5] K. Deb, S. Agrawal, A. Pratap, T. Meyarivan. *A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II*, in M. Schoenauer et al., editors, *Conference on Parallel Problem Solving from Nature (PPSN VI)*, volume 1917 of LNCS, pp. 849-858. Springer, 2000.

[6] E. Zitzler, M. Laumanns, L. Thiele. *SPEA2: Improving the Strength Pareto Evolutionary Algorithm*. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Zurich, Switzerland, 2001.

[7] E. Zitzler, S. Künzli. *Indicator-based Selection in Multiobjective Search*, in *8th International Conference on Parallel Problem Solving from Nature (PPSN VIII)*, pp. 832-842. Springer-Verlag, 2004.

[8] E. Zitzler, D. Brockhoff, L. Thiele. *The Hypervolume Indicator Revisited: On the Design of Pareto-compliant Indicators Via Weighted Integration*, in S. Obayashi et al., editors, *Conference on Evolutionary Multi-Criterion Optimization (EMO 2007)*, volume 4403 of LNCS, pp. 862-876. Berlin, 2007. Springer.

[9] T. Cormen, C. Leiserson, R. Rivest, C. Stein. *Introduction to Algorithms*. 2nd edition, 2001, The MIT Press, Cambridge Massachusetts.

[10] T. Ottmann, P. Widmayer. *Algorithmen und Datenstrukturen*. Reihe Informatik, 3rd rev. edition, 1996, Spektrum Akdemischer Verlag.

[11] M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. 2nd, rev. edition, Springer, 2000.

[12] F. Preparata, M. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, 1988.

[13] C. Aragon, R. Seidel. *Randomized search trees*. In *Algorithmica*, vol. 16, pp. 464-497. 1996.

[14] A. Duch, V. Estivill-Castro, C. Martinez. *Randomized k-dimensional binary search trees*. Technical Report, Departament de Llenguatges i Sistemes Informatics, Polytechnic University of Catalonia, Barcelona, 1998.

[15] P. Chanzy and L. Devroye, C. Zamora-Cura. *Analysis of range search for random k-d trees*. Technical Report, School of Computer Science, McGill University, Montreal, 1999.

[16] J. Fieldsend, R. Everson, S. Singh. *Using Unconstrained Elite Archives for Multiobjective Optimization*, in *IEEE Trans. on Evol. Comput.*, vol. 7, no. 3, pp. 305-323, June 2003.

[17] S. Mostaghim, J. Teich. *Quad-trees: A Data Structure for Storing Pareto Sets in Multiobjective Evolutionary Algorithms with Elitism*, in *Evolutionary Multiobjective Optimization*, pp. 81-104. Springer, 2005.

[18] S. Mostaghim, J. Teich, A. Tyagi. *Comparison of data structures for storing Pareto sets in MOEAs*, in *Proceedings of the 2002 Congress on Evolutionary Computation, Part of the 2002 IEEE World Congress on Computational Intelligence, Hawaii*. Piscataway, NJ: IEEE Press, 2002.

[19] M. Sun, R. Steuer. *InterQuad: An interactive quad tree based procedure for solving the discrete alternative multiple criteria problem*, in *European journal of Operational Research*, vol. 89, pp. 462-472, 1996.

[20] O. Schütze. *A New Data Structure for the Nondominance Problem in Multi-Objective Optimization*, *Proceedings of EMO 2003*. Faro, Portugal, 2003.

[21] M. Lukasiewycz, M. Glaß, C. Haubelt, J. Teich. *Symbolic Archive Representation for a Fast Nondominance Test*, in *Proceedings of the Fourth International Conference on Evolutionary Multi-Criterion Optimization (EMO 2007)*. Sendai, Japan, pp. 111-125, March 5-8, 2007.

[22] R. Drechsler, B. Becker. *Binary Decision Diagrams: Theory and Implementation*. Boston, Kluwer Academic Publishers, 1998.

[23] R. Rudell. *Dynamic Variable Ordering for Ordered Binary Decision Diagrams*, in *Proceedings of the 1993 IEEE/ACM International Conference on Computer-Aided Design (ICCAD'93)*, pp. 42-47. Los Alamitos, CA, USA, IEEE Computer Society Press, 1993.