Diplomarbeit

Genetische Programmierung im Risikomanagement

Freie wissenschaftliche Arbeit für die Diplomprüfung für Kaufleute an der Wirtschaftswissenschaftlichen Fakultät der Eberhard-Karls-Universität Tübingen.

eingereicht von

Andreas Dewes

geboren am 31.05.1983 in Ludwigsburg

Abgabetermin: 29. Januar 2009

Eingereicht bei Herrn Prof. Dr. Michael Merz

> It is interesting to contemplate an entangled bank, clothed with many plants of many kinds, with birds singing on the bushes, with various insects flitting about, and with worms crawling through the damp earth, and to reflect that these elaborately constructed forms, so different from each other, and dependent on each other in so complex a manner, have all been produced by laws acting around us. These laws, taken in the largest sense, being Growth with Reproduction; inheritance which is almost implied by reproduction; Variability from the indirect and direct action of the external conditions of life, and from use and disuse; a Ratio of Increase so high as to lead to a Struggle for Life, and as a consequence to Natural Selection, entailing Divergence of Character and the Extinction of less-improved forms. Thus, from the war of nature, from famine and death, the most exalted object which we are capable of conceiving, namely, the production of the higher animals, directly follows. There is grandeur in this view of life, with its several powers, having been originally breathed into a few forms or into one; and that, whilst this planet has gone cycling on according to the fixed law of gravity, from so simple a beginning endless forms most beautiful and most wonderful have been, and are being, evolved.

Charles Darwin. [Darwin, 1859]

Inhaltsverzeichnis

Symbolverzeichnis

α	Sicherheitsniveau für die Berechnung des Value at Risk sowie Signifikanzniveau für Fehler 1. Art, Seite 21
β	Skalen-Paramter der generalisierten Pareto-Verteilung, Seite 31
δ	Mischparamter des genetischen Mischoperators, Seite 52
$\hat{\mu}(X)$	Mittelwert der Stichprobe X, Seite 29
$\hat{\sigma}(X)$	Standardabweichung der Stichprobe X, Seite 29
$\mathbb{N}$	Menge der positiven ganzen Zahlen, Seite VII
$\mathbb{R}$	Menge der reellen Zahlen, Seite VII
`cs`	Crossover-Anteil in der genetischen Programmierung, Seite 15
`mr`	Mutationsrate in der genetischen Programmierung, Seite 15
`p`	Populationsgröße in der genetischen Programmierung, Seite 15
`sr`	Shrink-Mutationsrate in der genetischen Programmierung, Seite 15
`ts`	Tournament-Größe in der genetischen Programmierung, Seite 15
ν	Anzahl der Freiheitsgrade der student-t Verteilung, Seite 29
ρ	Der Pearsonsche Korrelationskoeffizient zweier Zufallsvariablen, Seite VII
σ	Die Standardabweichung einer Zufallsvariablen, Seite VII
θ	Shape-Paramter der Gumbel-, Clayton- bzw. Frank-Copula, Seite 49
ξ	Shape-Paramter der generalisierten Pareto-Verteilung, Seite 31
f^a	Angepasste Fitness in der genetischen Programmierung, Seite 9
f^n	Normierte Fitness in der genetischen Programmierung, Seite 9
f^r	Rohe Fitness in der genetischen Programmierung, Seite 9
f^s	Standardisierte Fitness in der genetischen Programmierung, Seite 9
u	Schwellenparameter für die Peak-Over-Treshold Methode, Seite 30
$\mathcal{N}$	Menge der Nichtterminalsymbole in der genetischen Programmierung, Seite 15
$\mathcal{T}$	Menge der Terminalsymbole in der genetischen Programmierung, Seite 15

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einführung

Le risque est l'onde de proue du succès.[1] (Carl Amery)

Der Begriff Risiko ist auf den vulgärlateinischen Ausdruck *risicare/resecare* zurückzuführen, was im ursprünglichen Sinne soviel wie "Gefahr laufen, wagen" bedeutete. Auch heute wird der Begriff im Sprachgebrauch oft als "gefühlte Gefahr" interpretiert, also als eine Gefahr, deren Auftreten nicht vorhersehbar ist. Im Gegensatz zu diesem allgemeinen Risikobegriff zielt der in dieser Arbeit verwendete Begriff des Risikos jedoch eher auf eine mathematische-betriebswirtschaftliche Sichtweise ab. So wird in Anlehnung an [McNeil et al., 2005, S. 4] **Risiko** im folgenden definiert als

> **Risiko:** Ein jegliches Ereignis, welches die Fähigkeit einer Organisation zur Erreichung ihrer Ziele oder der Durchführung ihrer Strategie negativ beeinflussen kann.

Als Organisation ist dabei häufig ein Unternehmen bzw. ein Betrieb im wirtschaftlichen Sinne bezeichnet, es kann sich jedoch z.B. auch um eine natürliche Personen, einen Staat oder eine sonstige Organisationseinheit handeln. Ergänzend sei angemerkt, dass der Eintritt (oder Nichteintritt) des erwähnten Ereignisses üblicherweise nicht mit Sicherheit vorhergesagt werden kann. Wäre dies nicht der Fall, so könnte bereits vor dem Eintreffen des Ereignisses eine sichere Entscheidung über den Umgang mit diesem getroffen werden und eine weitergehende Beschäftigung mit dem zugehörigen "Risiko" wäre somit überflüssig. Da risikobedingte Ereignisse im Allgemeinen jedoch nicht mit absoluter Sicherheit vorhergesagt werden können, ist ein Verfahren zum Umgang mit diesen für jede wirtschaftliche Tätigkeit unerlässlich. Ein solches Verfahren wird im folgenden erläutert.

[1] Risiko ist die Bugwelle des Erfolgs.

1.1. Risikomanagement

Die für den Umgang mit betriebswirtschaftlichen und allgemeinen Risiken entwickelten Methoden werden meist unter dem Begriff **Risikomanagement** zusammengefasst. Im engeren, betriebswirtschaftlichen Sinne ist Risikomanagement dabei definiert als [Wolke, 2008, S. 3]

> **Risikomanagement:** Die unternehmensweite Messung und Steuerung aller betriebswirtschaftlichen Risiken.

Die Gründe für den Einsatz von Risikomanagement sind dabei vielfältig und können praktischer aber auch rechtlicher Natur sein. Allgemein wird dabei zwischen **quantitativem** und **qualitativem Risikomanagement** unterschieden.

1.1.1. Quantitatives Risikomanagement

Das quantitative Risikomanagement befasst sich mit im mathematischen Sinne messbaren Risiken. Oft wird dabei auf die geldwerte Definition eines Risikos abgezielt, davon unabhängig können jedoch auch völlig andere Maße der Risikobewertung zugrunde liegen. Beispiele könnte die Entwicklung eines Aktienwertes, die maximale Windstärke an einem bestimmten geographischen Punkt oder auch der Pegelstand eines bestimmten Flusses Gegenstand quantitativer Risikobetrachtungen sein. Die betrachteten Risikogrößen müssen dabei stets zumindest intervallskaliert vorliegen.

Diese Arbeit wird sich vor allem auf die Untersuchung quantitativer Risiken im Zusammenhang mit der Preisentwicklung von Aktienwerten befassen, qualitative Risikoaspekte werden hierbei nicht berücksichtigt.

1.1.2. Qualitatives Risikomanagement

Das qualitative Risikomanagement befasst sich im Gegensatz zum quantitativen Risikomanagement mit der Behandlung von Risiken, die nicht im mathematischen Sinne quantitativ erfasst werden können. Eine typische Anwendung des qualitativen Risikomanagements ist z.B. die Beurteilung der Kreditwürdigkeit einer Organisation durch eine Rating-Agentur. Die Kreditwürdigkeit wird dabei häufig über sog. Scoring-Modelle berechnet, welche qualitative und vereinzelt auch quantitative Merkmale der betrachteten Organisation zu einem Rating verarbeiten, welches

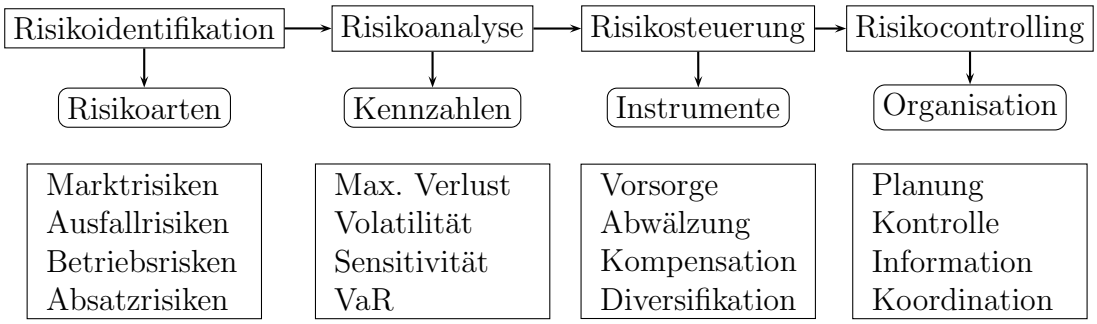


Abbildung 1.1.: Der Risikomanagementprozess nach [Wolke, 2008, S. 4]

als kategorielles Merkmal dann Aufschluss über die Bonität der Organisation gibt. Die betrachteten Kategorien besitzen dabei lediglich eine ordinale Vergleichbarkeit (d.h. man kann alle Kategorien in eine eindeutige Ordnung bringen, jedoch keine Aussage zum "Abstand" zweier unterschiedlicher Kategorien machen).

1.1.3. Risikomanagementprozess

Die praktische Durchführung des Risikomanagements wird üblicherweise als Prozess organisiert, der sich dabei oft an den klassischen Managementprozess anlehnt[Wolke, 2008, S. 4] und wie in Abb. 1.1 dargestellt unterteilt werden kann. Die einzelnen Stufen des Risikomanagementprozesses sind dabei die **Risikoidentifikation**, die **Risikoanalyse**, die **Risikosteuerung** und das **Risikocontrolling**. Im folgenden werden diese Begriffe kurz erläutert.

1.1.3.1. Risikoidentifikation

Mit der Risikoidentifikation erfolgt die Erfassung aller betriebsrelevanten Einzelrisiken innerhalb der untersuchten Organisation. Diese Identifikation kann hierbei z.B. über Analyseraster, Risikotabellen, Interviews bzw. durch Analyse aller Geschäftsprozesse erfolgen[Wolke, 2008, S. 6 ff.]. Üblicherweise wird zusätzlich zur Identifikation auch gleich eine Klassifizierung der gefundenen Risiken vorgenommen. Hierbei wird oft zwischen **naturwissenschaftlichen** und **wirtschaftswissenschaftlichen Risiken** unterschieden, wobei beide Kategorien u.U. nicht scharf trennbar sind (man denke z.B. an Erdbeben als gleichermaßen naturwissenschaftliches als auch wirtschaftswissenschaftliches Risiko für ein Rückversicherungsunter-

nehmen). Weiterhin kann man in betriebs- und volkswirtschaftliche Risiken trennen, wobei betriebswirtschaftliche Risiken vom untersuchten Betrieb selbst verursacht werden, wohingegen volkswirtschaftliche Risiken quasi global durch die gesamte Wirtschaftstätigkeit aller Wirtschaftsakteure erzeugt werden (z.B. Konjunkturkrisen).

1.1.3.2. Risikoanalyse

Aufbauend auf der Risikoidentifikation folgt die **Risikomessung** und die **Risikoanalyse**. Die Risikomessung erfasst dabei entweder qualitativ oder quantitativ die identifizierten Risiken anhand von **Risikofaktoren**. Die Risikoanalyse generiert anschließend aufbauend auf dieser Risikomessung Kennzahlen bzw. sonstige qualitative oder quantitative Maßzahlen, anhand derer das durch einzelne Risikofaktoren gegebene Risiko entscheidungstheoretisch behandelt werden kann. Die Risikoanalyse soll dabei v.a. klären, ob in den späteren Schritten des Risikomanagementprozesses Maßnahmen zur Reduktion einzelner Risiken ergriffen werden müssen. Weiterhin stellt die Risikoanalyse u.U. quantitative Informationen zur späteren Risikoplanung und -steuerung bereit. Einer quantitativen Risikoanalyse entspricht beispielsweise die Modellierung der Volatilität eines Aktienpreises, ein Beispiel für eine qualitative Risikoanalyse wäre hingegen die Erstellung eines Kredit-Ratings durch eine Rating-Agentur.

1.1.3.3. Risikosteuerung

Aufbauend auf der Risikoanalyse stellt die Risikosteuerung Instrumente zur Steuerung der analysierten Risiken bereit. Mögliche Instrumente hierfür sind z.B. die Bereitstellung von Sicherheiten bei der Tätigung von Geschäften, die Einrichtung von Order-Limits sowie die Portfoliooptimierung bei Aktientransaktionen, sowie natürlich die (Rück-)Versicherung einzelner oder zusammengehöriger Risiken.

1.1.3.4. Risikocontrolling

Das Risikocontrolling baut schließlich auf allen vorherigen Stufen des Risikomanagementprozesses auf und zielt auf die Unterstützung der Unternehmensführung beim Risikomanagement durch Planung, Kontrolle und Information ab[Wolke, 2008, S. 239]. Das Risikocontrolling soll somit die Risikoanalyse und die

Risikosteuerung als Prozess im betrachteten Unternehmen verankern und übt eine nicht zu unterschätzende Kontrollfunktion für den Risikomanagementprozess aus.

1.2. Genetische Programmierung

> Am Ende hängen wir doch ab
> Von Kreaturen, die wir machten. (Johann Wolfang Goethe - Faust II)

Genetische Programmierung (GP) ist eine Optimierungstechnik aus dem Bereich der Künstlichen Intelligenz und kann dort dem Unterbereich der evolutionären Algorithmen zugerechnet werden. Erfunden wurde genetische Programmierung durch John Koza[Koza, 1992]. Bereits vor der Erfindung von GP wurden sogenannte genetische Algorithmen (GA) eingesetzt um Optimierungsprobleme mit evolutionären Verfahren zu lösen[Holland, 1962, Holland, 1992b]. Das Optimierungsverfahren der GA wurde dabei dem Reproduktions- und Paarungsmechanismus eines DNA-Strangs nachempfunden. Da genetische Algorithmen als Grundlage für die Entwicklung der genetischen Programmierung angesehen werden können, werden sie im folgenden kurz erläutert.

Generell ist bei den GA für ein gegebenes, bereits modelliertes Problem eine eindimensionale Liste von meist numerischen Parametern zu optimieren. In Anlehnung an die Genetik wird diese Liste oft als Strang und ein einzelner Parameter innerhalb der Liste als Gen bezeichnet. Bei der Durchführung des Algorithmus werden dann zunächst viele unterschiedliche und meist zufällig erzeugte Parameterstränge generiert. Diese werden anschließend auf ihre Eignung als Eingabeparameter zur Lösung des betrachteten Problems geprüft. Einige der in diesem Sinne besten Stränge werden anschließend nach einem bestimmten Verfahren paarweise ausgewählt und zunächst einzeln an einer bestimmten Position "aufgetrennt", womit jeder Strang in zwei Teilstränge zerfällt. Anschließend werden genau zwei dieser Teilstränge zwischen den Gesamtsträngen ausgetauscht. Schließlich werden noch zufällig ausgewählte Elemente einzelner Stränge "mutiert".

Diese Vorgehensweise entspricht grob dem Konzept der geschlechtlichen Fortpflanzung in der Natur und erzeugt zusammen mit einem passenden Selektionsmechanimus in jedem Schritt des Algorithmus bessere Parameterstränge. Ein Beispiel für den Einsatz eines solchen genetischen Algorithmus wäre z.B. die Optimierung

eines Routenplanes für einen reisenden Kaufmann (das sog. **traveling salesman problem**)[Applegate et al., 2007], wobei die entsprechende Abfolge der einzelnen Reiseorte als Parameter in den Parametersträngen gespeichert wäre und das Ziel der Optimierung im Auffinden der kürzesten Reiseroute für den Kaufmann liegt. Zu diesem Problem ähnliche Fragestellungen treten heute in der Logistik häufig auf und sind der Klasse der NP-äquivalenten Probleme zuzuordnen, was eine exakte Lösung für eine große Anzahl an Reiseorten extrem erschwert oder gar unmöglich macht. Wären beispielsweise insgesamt 50 Städte zu bereisen, so müssten theoretisch bereits $50! \approx 3.04 \cdot 10^{64}$ Lösungen auf ihre Optimalität hin untersucht werden, was einen herkömmlichen Suchalgorithmus für die Optimierung quasi ausschließt. Mit genetischen Algorithmen kann hingegen eine nahezu optimale Lösung in recht kurzer Zeit gefunden werden[Grefenstette et al., 1985].

Im Unterschied zu den GA werden bei der GP jedoch nicht nur Parameterstränge für bereits vorgegebene Programme, sondern darüber hinaus auch die zur Lösung des Optimierungsproblems geeigneten Programme selbst erzeugt. Viele der für GA entwickelten Verfahren und Techniken lassen sich dabei unmittelbar auf die GP übertragen[Langdon and Poli, 1998, Eiben and Smith, 1998]. Sowohl GA als auch GP unterliegen dem sog. Schema-Theorem[Holland, 1992a], welches besagt, dass die Anzahl der von einem der beiden Verfahren geprüften Lösungen im Suchraum des jeweiligen Optimierungsproblems im Verlauf des Verfahren exponentiell ansteigt, was diese sehr interessant für eine große Klasse "harter" Optimierungprobleme macht. Im folgenden werden genetische Algorithmen nicht weiter behandelt, da sie im Rahmen der Arbeit nicht verwendet wurden. Stattdessen wird intensiv auf die Grundlagen der genetischen Programmierung eingegangen.

1.2.1. Grundlagen der genetischen Programmierung

Die für die Durchführung der genetischen Programmierung benötigen Techniken werden ausführlich in den als Standardwerken betrachteten Büchern von J. Koza[Koza, 1992, Koza, 1994, Koza, 1999, Koza, 2003] erläutert. Abb. 1.2 zeigt das gebräuchlichste Flussdiagramm zur genetischen Programmierung. Der zur Durchführung der genetischen Programmierung verfolgte Algorithmus wird dabei im folgenden meist als **GP-Algorithmus** bezeichnet.

Zunächst erzeugt der Algorithmus eine große Anzahl zufallsgenerierter **Programme**, welche zur Lösung eines gegebenen Optimierungsproblems eingesetzt

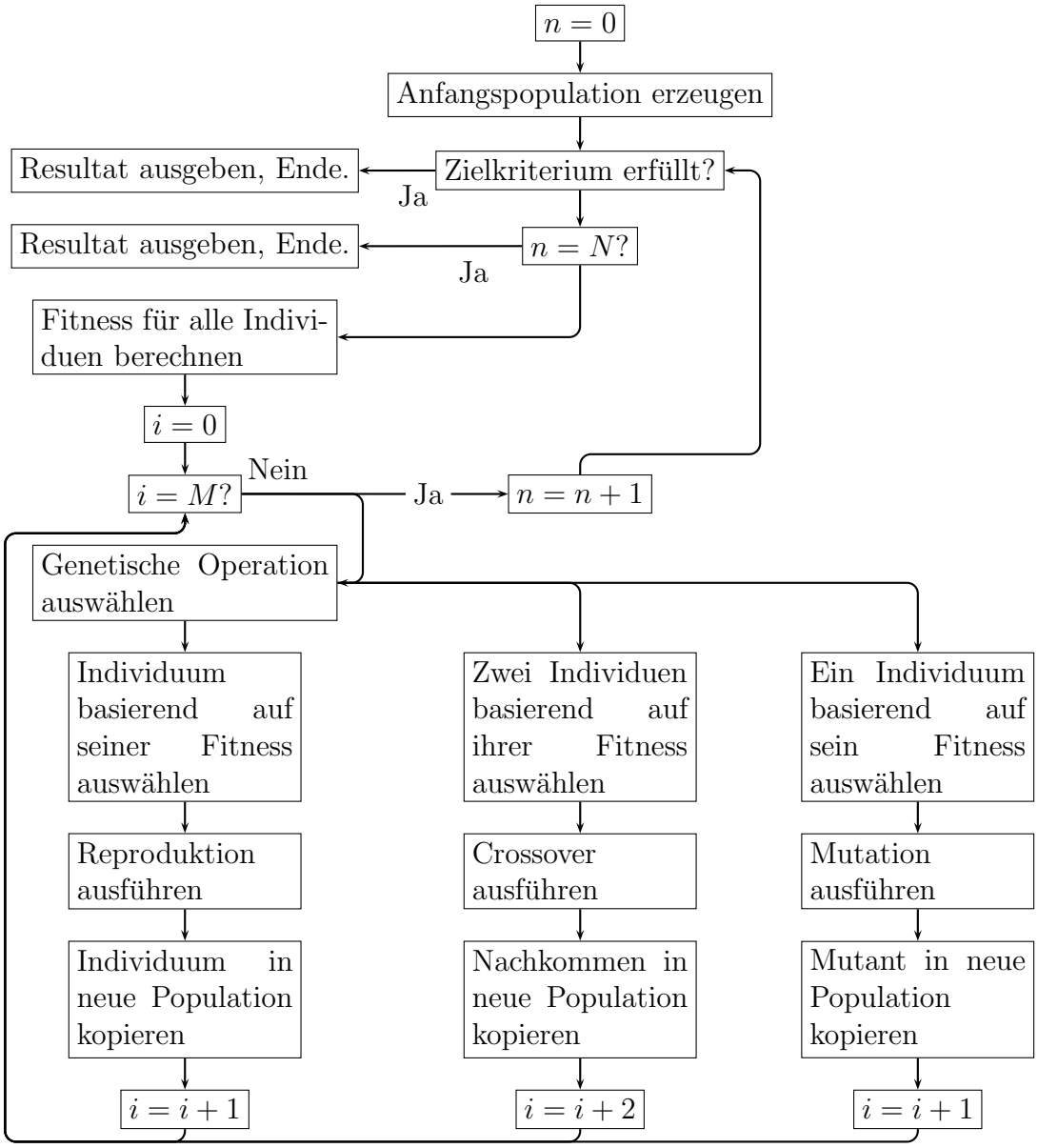


Abbildung 1.2.: Flussdiagramm der genetischen Programmierung nach [Koza, 1992]

werden können. Diese Programme werden oft auch als **Individuen** bezeichnet. Die Gesamtmenge dieser Individuen wird hierbei **Population** genannt. Die einzelnen Individuen können als Computerprogramme ausgeführt werden, erhalten als Eingabe die relevanten Parameter des untersuchten Optimierungsproblems und erzeugen daraus eine oft näherungsweise Lösung des gegebenen Problems. Die Güte der von einem Individuum gefundenen Lösung wird hierbei über die sog. **Fitness** gemessen. Diese Fitness definiert auf der Menge aller Programme eine Quasiordnung im mathematischen Sinne, macht die Individuen also paarweise hinsichtlich ihrer Eignung zur Lösung des gegebenen Problems vergleichbar. Eine höhere Fitness entspricht dabei einer besseren Lösung des Optimierungsproblems. Ist eine initiale Population von Individuen gefunden, so werden im Rahmen des GP-Algorithmus sich wiederholende Optimierungsschritte zur Verbesserung der Fitness der Individuen ausgeführt. Diese Schritte werden im folgenden kurz erläutert.

In jedem Schritt werden zunächst alle Individuen in der vorhandenen Population anhand ihrer Fitness beurteilt. Dabei wird geprüft, ob von einem oder mehreren Individuen ein bereits zuvor gewähltes Optimierungskriterium erfüllt wurde. Falls dies der Fall ist, wird der Algorithmus beendet und die generierten Individuen werden als Lösung ausgegeben, u.U. wird für die Ausgabe dabei auch nur das beste Individuum berücksichtigt. Der Algorithmus endet ebenso, falls eine vorgegebene Anzahl an Optimierungsschritten erreicht wurde. Ist keines dieser beiden Kriterien gegeben, so wird im nächsten Schritt anhand eines vorher festgelegten Verfahrens eine vorgegebene Anzahl von Individuen aus der ursprünglichen Population anhand ihrer Fitness ausgewählt. Die gewählten Individuen werden anschließend zufallsgesteuert bestimmten **genetischen Operationen** unterzogen und zu einer neuen Population zusammengefasst. Die gebräuchlichsten genetischen Operationen sind hierbei

- **Reproduktion**: Ein einzelnes ausgewähltes Individuum wird unverändert in die neue Population übernommen.

- **Mutation**: Ein einzelnes ausgewähltes Individuum wird mutiert, d.h. seine Programmstruktur wird zufällig geändert um ein neues Individuum zu erzeugen.

- **Crossover**: Zwei ausgewählte Individuen werden "gepaart", wobei wiederum

zwei neue Individuen als genetische Nachkommen der ursprünglichen Programme entstehen.

- **Permutation**: Die Reihenfolge einzelner Programmteile innerhalb des ausgewählten Individuums wird zufällig permutiert.
- **Shrink-Mutation**: Das Programm des ausgewählten Individuums wird zufallsgesteuert gekürzt, womit ein neues Individuum erzeugt wird.

Nach der Durchführung dieser Schritte liegt schließlich eine neue **Generation** von Individuen vor und der nächste Optimierungsschritt kann ausgeführt werden. Da bei jedem Schritt möglichst eine Verbesserung der gefundenen Lösung angestrebt wird, muss die Auswahl der Individuen aus der ursprünglichen Generation so erfolgen, dass solche Individuen mit hoher Fitness eine erhöhte Auswahlwahrscheinlichkeit aufweisen. Um dies zu erreichen, eignen sich verschiedene **Selektionsverfahren**. Bei der **fitnessproportionalen Selektion** werden einzelne Individuen statistisch anhand ihrer Fitness ausgewählt. Der Wert der Fitness ist dabei proportional zur Auswahlwahrscheinlichkeit eines Individuums. Dieses Verfahren verlangt die Verwendung der sog. **normierten Fitness** f_i^n, welche stets im Intervall $[0, 1]$ liegt (der Index i bezeichnet hierbei das jeweilige Individuum). Die normierte Fitness wird dabei ausgehend von der **rohen Fitness** f_i^r (welche einen beliebigen Wertebereich besitzen kann), über die **standardisierte** und die **angepasste Fitness** berechnet. Die standardisierte Fitness ist dabei dabei nach obiger Definition der rohen Fitness (ein höherer Wert der rohen Fitness entspricht einer besseren Lösung) definiert als $f_i^s = \max_j f_j^n - f_i^n$, wobei der Index j über alle Individuen einer Generation läuft. D.h. ein Wert $f_i^s = 0$ entspricht stets der Fitness des *besten* Individuums innerhalb der Generation. Die angepasste Fitness schließlich berechnet sich aus der standardisierten Fitness als $f_i^a = 1/\left(1 + f_i^s\right)$, d.h. sie nimmt für das beste Individuum der Generation den Wert $f_i^a = 1$ an und sinkt mit steigendem Wert von f_i^s. Die normierte Fitness berechnet sich schließlich aus der angepassten Fitness als [Koza, 1992, S. 95 ff.]

$$f_i^n = \frac{f_i^a}{\sum_j f_j^a} \tag{1.1}$$

Die normierte Fitness hat einige vorteilhafte Eigenschaften und kann als Wahrscheinlichkeitsmaß zur fitnessbasierten Selektion herangezogen werden.

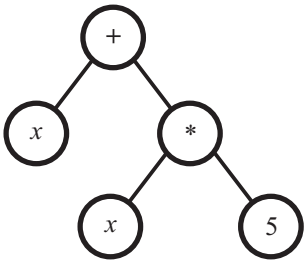


Abbildung 1.3: Die Baumrepräsentation eines genetischen Programms $x + 3 \cdot x$

Im Gegensatz hierzu wählt die in dieser Arbeit eingesetzte **Tournament-Selektion** Individuen in einer Art "Gruppenwettkampf" anhand ihrer relativen Fitness aus. Die Tournament-Selektion läuft dabei folgendermaßen ab:

Algorithmus 1.2.1 (Tournament-Selektion (TS)).

1. Wähle zufällig m Individuen $(i_1, \ldots, i_m)$ aus der Ausgangspopulation aus.
2. Vergleiche die Fitness der ausgewählten Individuen und selektiere das Individuum i_k mit der größten Fitness, so dass $f_{i_k} > f_{i_j}$ für $\forall j = 1, \ldots m$.
3. Entferne das selektierte Individuum i_k aus der Ausgangspopulation.

Die Tournament-Selektion (TS) hat dabei den großen Vorteil, dass die gewählte Fitness nicht normiert sein muss und dass auch kein globaler Vergleich aller Fitness-Werte für die Selektion benötigt wird. Weiterhin bildet TS die biologische Evolution realitätsgetreuer ab als z.B. die fitnessproportionale Selektion, da in der Natur üblicherweise nur kleine Gruppen von Individuen um die Selektion kämpfen und nicht etwa ein Vergleich sämtlicher Individuen einer Population stattfindet (abgesehen vielleicht von einigen Vogel- und Säugetierarten, die sich zur Paarung in einer großen Brutkolonie zusammenfinden). Die Größe der einzelnen Wettkampfgruppen ist dabei ein wichtiger Parameter und sollte sorgfältig gewählt werden. In allen im folgenden durchgeführten Untersuchungen wurde das Tournament-Verfahren zur Selektion eingesetzt.

1.2.2. Repräsentation der Individuen

Zur Repräsentation der Individuen in der genetischen Programmierung wird üblicherweise eine Baumdarstellung gewählt[Banzhaf et al., 2002, S. 105 ff.]. Ein

Baum ist hierbei eine hierarchische Struktur bestehend aus einem oder mehreren **Knoten**, wobei einer dieser Knoten als sogenannte **Wurzel** des Baums fungiert. Jeder Knoten kann einen oder mehrere Unterknoten besitzen. Weiterhin wird zwischen **Ästen (Branches)** und **Blättern (Leafs)** des Baumes unterschieden. Äste besitzen hierbei einen oderer mehrere Unterknoten, Blätter besitzen hingegen keinen solchen Nachfolger. Im Rahmen der Programmierung werden Äste auch oft als **Nichtterminal-Symbole** und Blätter als **Terminal-Symbole** bezeichnet, wobei Terminal-Symbole (engl. "to terminate" = "abschließen") die Enden des Progammbaumes markieren. Abb. 1.3 zeigt beispielhaft die Baumdarstellung des Programms $x + 3 \cdot x$. Die genetischen Operatoren (Crossover, Mutation, Shrink-Mutation) lassen sich in der Baumrepräsentation eines Programms sehr einfach realisieren, was im folgenden gezeigt wird.

1.2.3. Genetische Operatoren

Die genetischen Operatoren modifizieren einzelne Teile des Programmbaumes, indem sie vorhandene Knoten ersetzen, löschen oder austauschen. Im folgenden seien kurz die einzelnen Operatoren und ihre Wirkung auf die Programmstruktur erläutert.

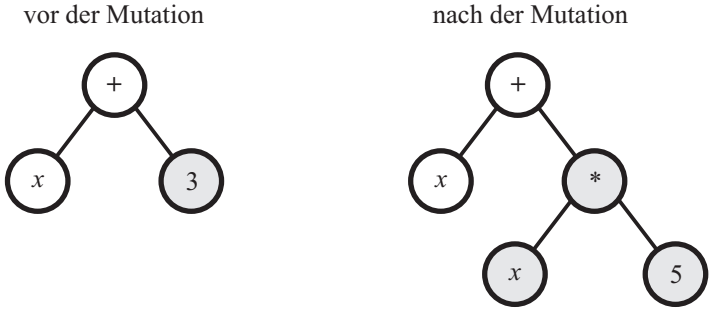


Abbildung 1.4.: Veranschaulichung des genetischen Mutationsoperators anhand des Programms $x + 3$, welches zu $x + x \cdot 5$ mutiert wird. Der zur Mutation ausgewählte Knoten sowie die ihn ersetzende Baumstruktur ist grau unterlegt.

1.2.3.1. Mutation

Abb. 1.4 zeigt beispielhaft die Wirkung des Mutationsoperators auf das Programm $x + 3$. Für die Mutation wird dabei zunächst ein Knoten des Baumes zufällig

ausgewählt (grau unterlegt). Dieser Knoten wird anschließend durch einen neuen, zufallsgenerierten Baum ersetzt. Das so entstehende Programm kann dabei sowohl länger als auch kürzer sein als das ursprüngliche Programm. Unter der **Länge** eines Programms wird dabei die Gesamtzahl der Knoten in seiner Baumdarstellung verstanden.

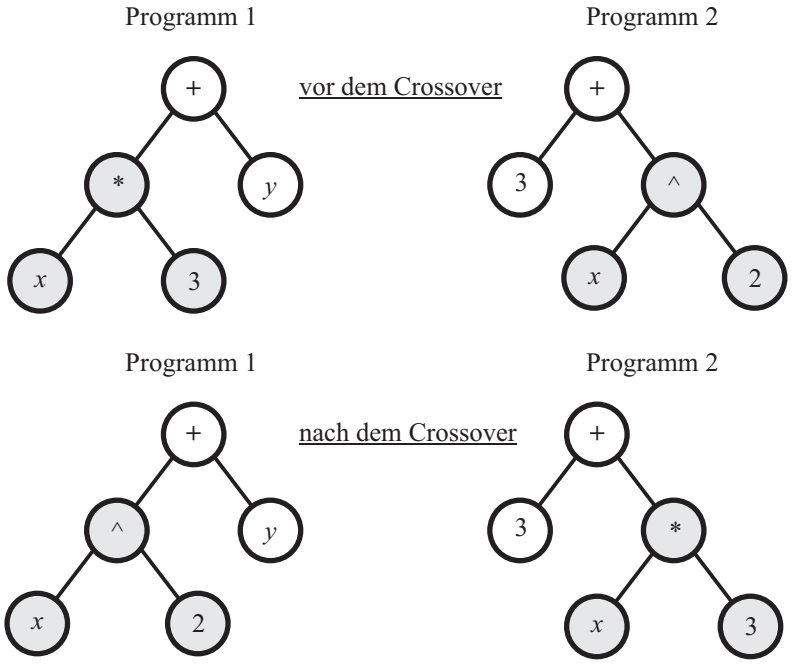


Abbildung 1.5.: Veranschaulichung des genetischen Crossover-Operators. Die beiden Eltern-Programme $x \cdot 3 + y$ und $3 + x^2$ erzeugen den Nachwuchs $x^2 + y$ sowie $3 + x \cdot 3$, der aus einer Mischung des genetischen Materials der beiden Elternteile entsteht.

1.2.3.2. Crossover

Der genetische Crossover erfolgt ähnlich wie die Mutation, jedoch werden hierfür insgesamt zwei Individuen aus der Population selektiert. Abb. 1.5 zeigt beispielhaft den Crossover zwischen den zwei Programmen $x \cdot 3 + y$ und $3 + x^2$. Hierbei wird zunächst von jedem Programm zufällig ein Teil des Baumes für den Crossover ausgewählt (grau unterlegt). Anschließend werden diese Bäume zwischen den beiden Individuen ausgetauscht. Auf diese Weise entstehen zwei neue Baumstrukturen, welche beide jeweils genetische Merkmale beider "Elternteile" tragen. Die

Gesamtlänge der beiden Programme bleibt bei dieser Operation unverändert, ebenso wird kein neues Genmaterial in die Programme eingebracht. Es sei angemerkt, dass die jeweils ausgetauschten Unterbäume nicht zwangsweise die gleiche Länge bzw. die gleiche Position innerhalb des jeweiligen Gesamtbaumes aufweisen müssen.

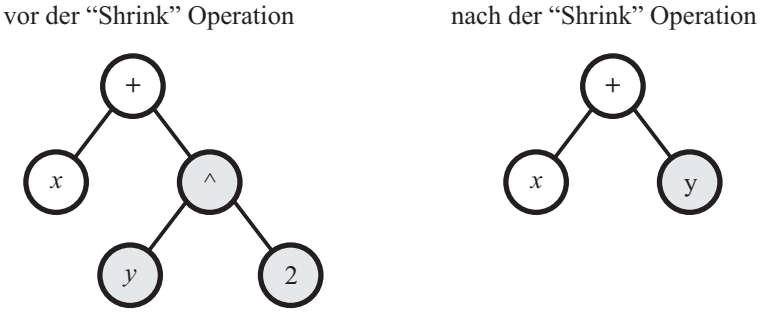


Abbildung 1.6.: Veranschaulichung des genetischen Shrink-Operators. Ein genetisches Program wird in seiner Länge verkürzt, indem ein zufällig ausgewählter Nicht-Terminal/Ast des Programmbaumes durch ein zufallsgeneriertes Terminal/Blatt ersetzt wird.

1.2.3.3. Shrink-Mutation

Die Shrink-Mutation, dargestellt in Abb. 1.6 verläuft analog zur normalen Mutation, jedoch wird nun zufällig ein Ast (und keinesfalls ein Blatt) aus dem Baum ausgewählt (grau unterlegt). Dieser Ast wird im zweiten Schritt durch ein Blatt ersetzt und der Baum wird so insgesamt gekürzt. Die Shrink-Mutation gehört nicht zu den unbedingt notwendigen Operatoren, die für einen funktionalen GP-Algorithmus benötigt werden, ist aber sehr nützlich um die durchschnittliche Programmlänge innerhalb der Population gering zu halten. Diese Verkleinerung ist oft nötig, da die Programmlänge im Verlauf des GP-Algorithmus nahezu exponentiell anwachsen kann, was als **Bloat** bezeichnet wird und oft unerwünscht ist. Alternativ zur Shrink-Mutation kann auch schlicht eine Maximallänge für die Programme einer Population festgelegt werden, wobei zu lange Programme einfach aus der Population entfernt und durch neue, zufallsgenerierte Programme ersetzt werden. Da hierbei jedoch große Mengen an genetischer Information verloren gehen ist die Shrink-Mutation in den meisten Fällen der Längenlimitierung vorzuziehen.

1.2.4. Weiterführende Konzepte

Neben den bereits vorgestellten Konzepten und Operatoren existieren weitere, fortgeschrittenere Konzepte der genetischen Programmierung, welche vorwiegend zur Erweiterung der Funktionalität dieser Methode eingeführt wurden. Ein oft verwendetes Konzept ist dabei die sog. **automatisch definierte Funktion (ADF)** (automatically defined function)[Koza, 1992]. Eine solche ADF stellt im Prinzip ein eigenständiges genetisches Programm dar, welches im Verlauf des GP-Algorithmus wie andere Programme auch durch Anwendung der genetischen Operatoren verändert werden kann. Dabei besitzt jedes reguläre genetische Programm einer Population eine oder mehrere ADFs. Jede ADF kann in einem regulären Programm als Nichtterminal aufgerufen werden. Somit steht eine ADF als Hilfsfunktion für ein reguläres genetisches Programm zur Verfügung und kann von diesem an mehreren Stellen im Programmbaum benutzt werden. Beispielsweise könnte bei der Berechnung der Differenz der Fläche zweier Rechtecke eine automatisch definierte Funktion die Berechnung des Flächeninhaltes eines jeden Rechtecks übernehmen, was die Lösung des Gesamtproblems innerhalb eines regulären Programmes sehr vereinfacht. ADFs sind somit sehr nützlich um komplexe Aufgaben in mehrere, einfacher zu lösende Teilaufgaben zu untergliedern. Diese einzelnen Teilaufgaben können dann jede für sich durch genetische Programmierung gelöst werden. Innerhalb dieser Arbeit werden ADFs jedoch nicht verwendet, daher werden sie auch nicht weitergehend diskutiert. Für weitere Details zu ADFs kann z.B. [Koza et al., 1996] herangezogen werden.

Eine weitere interessante Entwicklung ist die Erweiterung der genetischen Programmierung hin zu einem sog. **generellen Problemlöser** (general problem solver (GPS))[Newell and Simon, 1995, Feigenbaum and Feldman, 1995]. Dabei wird darauf abgezielt, dem GP-Algorithmus unabhängig vom betrachteten Problem einen Satz von Nichtterminalsymbolen zur Verfügung zu stellen, der eine Modellierung eines jeglichen denkbaren Sachverhalts ermöglicht, in Analogie zu dem vollständigen Satz an Operatoren den eine Turing-Maschine[Turing, 1936] zur Darstellung eines beliebigen Computerprogrammes benötigt. Hiermit wäre es prinzipiell möglich, jedes logisch fassbare Problem unter Formulierung einer passenden Fitnessfunktion und eines Satzes von Eingabewerten (Terminalsymbolen) mittels genetischer Programmierung zu lösen. Generell soll der generelle Problemlöser dabei auch menschliches Denken simulieren können[Newell and Simon, 1995]. Aufgrund

des sog. **No Free Lunch Theorems (NFLT)**[Wolpert and Macready, 1997] ist es jedoch prinzipiell nicht möglich, einen Optimierungsalgorithmus zu entwerfen, der auf allen denkbaren Klassen von Optimierungsproblemen eine gleichmäßige Performance (im Sinne der Qualität und der Zeit zum Auffinden guter Lösungen) aufweisen würde. Daher ist der Einsatz von genetischer Programmierung generell nicht für alle denkbaren Optimierungsprobleme zu empfehlen, kann aber trotzdem bei einer großen Anzahl bestimmter Probleme eine signifikant höhere Performance aufweisen als andere Verfahren. Im Rahmen dieser Arbeit wird nicht näher auf das Konzept des GPS eingegangen.

1.2.5. Praktische Aspekte der genetischen Programmierung

Bei der Behandlung eines gegebenen Problems mittels genetischer Programmierung sind zunächst alle relevanten Eingabeparameter (Terminalsymbole) zu erfassen. Weiterhin muss eine Fitnessfunktion gefunden werden, die sich dazu eignet die Qualität der gefundenen Lösungen zu bewerten. Schließlich ist eine Auswahl von Nichtterminalsymbolen zu treffen, welche innerhalb der zu generierenden genetischen Programme eingesetzt werden können. Ein üblicher Satz von Nichtterminalsymbolen für ein numerisches Optimierungsproblem sind die arithmetischen Operatoren $\mathcal{N} = \{+, -, \cdot, /\}$. Abschließend sind noch eine Reihe von Evolutionsparametern festzulegen, welche das Resultat des GP-Algorithmus teilweise sehr stark beeinflussen können. Die wichtigsten Parameter hierbei sind

- **Populationsgröße** (`p`): Gibt die Anzahl an Individuen in einer Population an. Typischer Wert: 10000
- **Mutationsrate** (`mr`): Gibt den prozentualen Anteil der Population an, welcher in jedem Optimierungsschritt einer Mutation unterzogen wird. Typischer Wert: 5%. Wird zusätzlich Shrink-Mutation verwendet, so gibt `sr` die Shrink-Mutationsrate an.
- **Crossover-Anteil** (`cs`): Gibt den prozentualen Anteil der Population an, welcher in jedem Optimierungsschritt der Crossover-Operation unterzogen wird. Typischer Wert: 50%
- **Tournament-Größe** (`ts`): Gibt die Größe einer Tournament-Runde innerhalb des Selektionsmechanismus an. Typischer Wert: 6

Die Wahl der vorgestellten Parameter hat dabei direkten Einfluss auf die Qualität der aus dem GP-Algorithmus erhaltenen Lösungen, die Konvergenzgeschwindigkeit des Verfahrens sowie dessen Stabilität. Im folgenden sei daher im Rahmen einer Sensitivitätsanalyse kurz der Einfluss einzelner Parameter auf den GP-Algorithmus beschrieben.

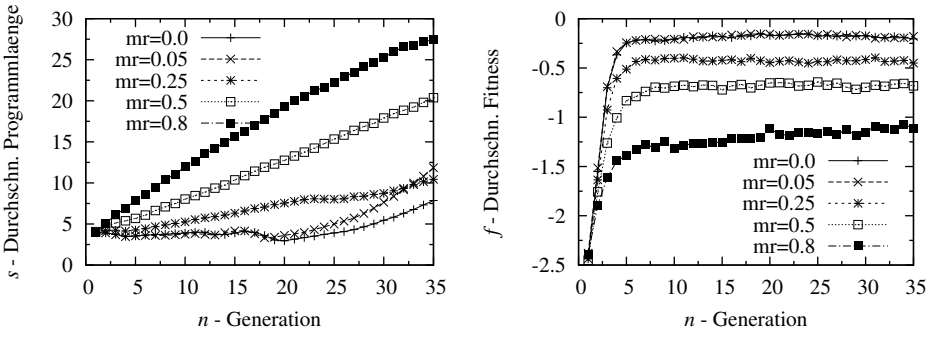


Abbildung 1.7.: (a) Durchschnittliche Länge und (b) durchschnittliche Fitness der genetischen Programme innerhalb einer Population in Abhängigkeit der Mutationsrate und der Generation.

1.2.6. Sensitivitätsanalyse

Die Evolutionsparameter `p`, `mr`, `cs` und `ts` bestimmen maßgeblich den Verlauf des GP-Algorithmus und sollten daher mit Sorgfalt gewählt werden. Die Mutationsrate als vermutlich wichtigster Evolutionsparameter bestimmt dabei maßgeblich die Konvergenzgeschwindigkeit sowie die Wachstumsgeschwindigkeit der Länge der generierten Lösungen. In der Standardimplementierung des Mutationsoperators besitzt dieser die Eigenschaft, die mutierten Individuen im Hinblick auf ihre Länge zu vergrößern, d.h. die Mutation fördert die Komplexität innerhalb der genetischen Population. Weiterhin gelangen durch den Mutationsoperator neue, vorher nicht vorhandene Gene in die Population, gleichzeitig werden jedoch vorhandene Gene u.U. ausgelöscht. Die Mutationsrate sollte daher so gewählt werden, dass einerseits kein zu starkes Komplexitätswachstum auftritt, andererseits aber genügend neue genetische Informationen in die Population gelangt und letztendlich auch bereits vorhandene genetische Information nicht übermäßig geschädigt wird.

Abb. 1.7 zeigt den Einfluss der Mutationsrate `mr` auf die durchschnittliche Länge der einzelnen Individuen in einer Population sowie auf deren durchschnittliche Fit-

ness. Die Daten für die Abbildung wurden mithilfe der `ESProgram`-Klasse generiert, auf welche im nachfolgenden Kapitel sowie im Appendix genauer eingegangen wird. Wie man in Abb. 1.7a leicht erkennen kann, steigt die durchschnittliche Länge der Individuen innerhalb einer Population im Verlaufe des Verfahrens üblicherweise umso stärker an, je höher die Mutationsrate liegt. Aus Abb. 1.7b lässt sich weiterhin folgern, dass eine übermäßig hohe Mutationsrate zu einer geringeren durchschnittlichen Fitness der Individuen führt. Generell betrachtet kommt es in der Population dabei unabhängig von der gewählten Mutationsrate zunächst zu einem rapiden Anstieg der durchschnittlichen Fitness. Dieser ist auf die Verbreitung der fittesten Individuen innerhalb der Population bzw. auf das Aussterben der weniger fitten Individuen zurückzuführen[Eiben and Smith, 1998]. Nach dieser schnellen Anpassungsphase tritt üblicherweise ein weniger schnelles, lineares Ansteigen der durchschnittlichen Fitness auf, welches durch die schrittweise genetische Verbesserung einzelner Individuen zustande kommt.

Die Populationsgröße `p` ist ein weiterer wesentlicher Parameter und sollte nicht zu klein gewählt werden. Generell gilt, je größer die Population, desto mehr genetische Information enthält diese und umso effizienter läuft der GP-Algorithmus ab. In den in dieser Arbeit durchgeführten Untersuchungen wurden bspw. keine Populationen mit weniger als `p` = 1000 Individuen untersucht.

Der Crossoveranteil `cs` sollte so gewählt werden, dass es nicht zu übermäßigem genetischen Crossover der Individuen kommt. Ist der Crossoveranteil zu hoch, so kommt es zu einer Überreproduktion der fittesten Individuen (vergleichbar dem Inzest innerhalb einer biologischen Population) und damit zu einer degenerierenden Wirkung des Crossoveroperators. Schließlich wird die Population hierduch immer weniger genetische Information tragen und letztlich frühzeitig zu einer einzigen Lösung konvergieren. Dies wird in der Literatur als **vorzeitige Konvergenz** (premature convergence)[Koza, 1992, S. 104 ff.] bezeichnet und ist unbedingt zu vermeiden.

Die Tournamentgröße `ts` kann über einen recht großen Bereich frei gewählt werden ohne die Resultate des GP-Algorithmus wesentlich zu stören. Bei zu großer Tournamentgröße kann es jedoch wiederum zur vorzeitigen Konvergenz des Algorithmus durch Überselektion der fittesten Individuen kommen.

Generell betrachtet müssen die optimalen Evolutionsparameter für jedes Problem individuell bestimmt werden, alternativ können sie jedoch auch selbst mittels eines genetischen Algorithmus oder eines anderen Optimierungsverfahrens festgelegt werden, was aufgrund der hohen Laufzeit des GP-Algorithmus häufig jedoch nicht durchführbar ist.

1.3. Anwendungen genetischer Programmierung

Genetische Programmierung wird heute in vielen Bereichen der Ingenieurwissenschaften, Informatik und nicht zuletzt auch in der Finanzmathematik eingesetzt. [Koza, 1992] setzte beispielsweise GP ein, um sog. "black art" Probleme zu lösen. Dies schließt beispielsweise das Design analoger elektronischer Schaltungen, elektromagnetischer Antennen, chemischer Reaktionsnetzwerke und optischer Systeme ein. Im Bereich der künstlichen Intelligenz wird genetische Programmierung zur Erzeugung intelligenter Steuerungssysteme eingesetzt. Innerhalb der Finanzmathematik erfreut sich genetische Programmierung mittlerweile einer Vielzahl von Anwendungen wie z.B. der technischen Analyse von Aktienkursen[Subramanian et al., 2006], dem Portfoliomanagement[Chen, 2008], der Vorhersage von Volatilitäten[Chen, 2008, Heigl, 2008], der Simulation von Kreditratings[Kleinau, 2003] und der Berechnung von Optionspreisen[Kleinau, 2003, Heigl, 2008]. Weiterhin wird genetische Programmierung im Bereich der evolutionären Kunst[Bentley and Corne, 2001] und des evolutionären Designs[Bentley, 1999] eingesetzt.

2. Risikoanalyse mit genetischer Programmierung

Das folgende Kapitel beschreibt die konkrete Anwendung genetischer Programmierung im Risikomanagement anhand zweier praxisrelevanter Fragestellungen. Es soll dabei gezeigt werden, dass mittels genetischer Programmierung generierte Risikomodelle vergleichbare bzw. teilweise bessere Ergebnisse erzielen können als mittels Standardverfahren generierte Modelle.

Genetische Programmierung eignet sich besonders gut für den Bereich der quantitativen Risikoanalyse, da die in der Analyse zu generierenden Risikomodelle meist recht einfach anhand von Maßzahlen hinsichtlich ihrer Qualität beurteilt werden können. Weiterhin basieren die meisten Modelle auf quantitativen Eingabeparametern, welche oft recht einfach aus den zu untersuchenden Risikogrößen abgeleitet werden können. Diese beiden Eigenschaften machen es möglich, mittels genetischer Programmierung automatisiert Risikomodelle zu erstellen und gegen reale Daten zu testen. Im Vergleich hierzu ist die Risikoidentifikation beispielsweise ein Prozess, der kaum automatisiert ablaufen kann und stets menschliches Zutun benötigt. Der folgende Teil der Arbeit konzentriert sich daher gänzlich auf den Bereich der Risikoanalyse.

2.1. Risikoanalyse von Zeitreihen

Die Risikoanalyse von Zeitreihen besitzt in der Risikoabschätzung bei Aktien- und Optionspreisen ein überaus wichtiges Anwendungsgebiet und weist heute im Finanzsektor eine überragende Bedeutung auf. Gerade die katastrophale Entwicklung der Finanz- und späteren Wirtschaftskrise im vergangenen und aktuellen Jahr zeigt hierbei deutlich die Wichtigkeit eines adäquaten Risikomanagements. Ausgelöst wurde die Krise nach heutigem Wissensstand vermutlich durch mangelndes

qualitatives, aber auch quantitatives Risikomanagement seitens der Finanzindustrie. Es ist daher zu erwarten, zumindest aber zu erhoffen, dass insbesondere dem quantitativen Risikomanagement und damit auch der Analyse von Wertpapierrisken zukünftig noch größere Bedeutung zugemessen werden wird. Die vorliegende Arbeit setzt den Schwerpunkt der folgenden Abschnitte daher auf die Analyse quantitativer Wertpapierrisken mithilfe genetischer Programmierung (GP). Es wird dabei gezeigt werden, wie mittels GP die heute gebräuchlichsten Risikomaße für univariate Zeitreihen verlässlich abgeschätzt werden können. Die Resultate des GP-Algorithmus werden anschließend mit klassischen Schätzverfahren verglichen und hinsichtlich ihrer Qualität und Praxiseignung beurteilt.

2.1.1. Univariate Risikomaße

Die Abschätzung univariater Risikomaße ist von hoher Bedeutung in einer Vielzahl von Anwendungsgebieten, beispielsweise bei der Bewertung von Aktien und Aktienoptionen. In den vergangenen Jahrzehnten wurden eine Vielzahl mehr oder weniger geeigneter Risikomaße hierfür entwickelt, eine Übersicht der gebräuchlichsten findet sich z.B. in [McNeil et al., 2005, S. 34 ff.].

Betrachtet wird dabei grundsätzlich meist eine Reihe von Realisierungen x_t eines zeitdiskreten, univariaten stochastischen Prozesses der Form

$$X \ : \ \Omega \times T \to \mathbb{R}, \ (\omega, t) \to x_t(\omega) \tag{2.1}$$

welcher auf dem Wahrscheinlichkeitsraum $(\Omega, \mathcal{F}, P)$ definiert ist und wobei $T \in \{\mathbb{N}_0\}$ gilt. Die Realisierungen x_t des Prozesses werden oft als **Risikofaktoränderungen** interpretiert. Üblicherweise ist die genaue Form des datengenerierenden Prozesses dabei unbekannt. Der Begriff **Risiko** bezieht sich in diesem Sinne auf die Wahrscheinlichkeitsverteilung der Realisierungen von X. Im praktischen Kontext oft gestellte Fragen lauten daher in etwa "Wie hoch ist die Auftrittswahrscheinlichkeit einer Realisierung $x_t > z$?" bzw. "Welcher Wert x wird mit einer gegebenen Wahrscheinlichkeit p_0 von den Realisierungen x_t der Zeitreihe nicht erreicht oder überschritten?". Die Beantwortung solcher Fragen führt zur Entwicklung statistischer **Risikomaße**. Ein Risikomaß ist in diesem Sinne eine reellwertige Funktion $\rho : \mathcal{M} \to \mathbb{R}$, wobei $\mathcal{M} \subset X^0(\Omega, \mathcal{F}, P)$ eine Teilmenge der Menge $X^0(\Omega, \mathcal{F}, P)$ der fast sicher beschränkten Realisierungen des betrachteten Wahrscheinlichkeitspro-

zesses bezeichnet. Im folgenden seien einige gebräuchliche Risikomaße exemplarisch genannt:

- **Zentrale Momente**: Das zentrale Moment k-ter Ordnung ist definiert als

$$\mu_k := \mathrm{E}\left(\left[X - \mu\right]^k\right) \tag{2.2}$$

  Hierbei ist μ der Erwartungswert der Zufallsvariablen X. Zentrale Momente können als Kenngrößen von Zufallsvariablen leicht interpretiert werden, gebräuchlich zur Risikoabschätzung ist dabei vor allem das zentrale Moment zweiter Ordnung, auch **Varianz** genannt:

$$\mathrm{var}(X) := \mathrm{E}(X)^2 - \mathrm{E}(X^2) \tag{2.3}$$

  Die Varianz beschreibt also Abweichungen der Zufallsvariable vom Mittelwert, unterscheidet jedoch nicht zwischen positiven und negativen Abweichungen, was den Einsatz als Risikomaß problematisch macht (da Verluste und Gewinne gleich behandelt werden).

- **Partialmomente**: Häufig werden untere Partialmomente zur Risikoabschätzung genutzt, diese beschreiben die negative Abweichung einer Zufallsvariablen von einem gegebenen Wert und besitzen daher nicht den Nachteil der zentralen Momente, welche Abweichungen in beide Richtungen vom Mittelwert berücksichtigen. Das untere Partialmoment k-ter Ordnung ist beispielsweise gegeben als

$$\mu_k^- := \mathrm{E}\left(\max\left\{c - X, 0\right\}^k\right) \tag{2.4}$$

- **Value at Risk (VaR)**: Der VaR gibt den Wert der Zufallsvariablen X an, der mit Wahrscheinlichkeit α nicht überschritten wird. Der VaR berechnet sich als

$$\mathrm{VaR}_\alpha = \inf\left\{x \in \mathbb{R} : P(X > x) \leq 1 - \alpha\right\} = \inf\left\{x \in \mathbb{R} : F_X(x) \geq \alpha\right\} \tag{2.5}$$

  Dabei ist $F_X(x)$ die kumulative Wahrscheinlichkeitsfunktion der Zufallsvariablen X. Der VaR ist heute eines der gebräuchlichsten und am häufigsten eingesetzten Risikomaße.

- **Conditional Value at Risk (CVaR)**: Der CVaR wird auch oft als **Expected Shortfall (ES)** bzw. **Expected Tails Loss (ETL)** bezeichnet. Er gibt den Erwartungswert von X an, unter der Bedingung, dass der Value at Risk zum Sicherheitsniveau α überschritten wurde und ist somit gegeben als

$$\mathrm{CVaR}_\alpha := \mathrm{E}(X|X > \mathrm{VaR}_\alpha) \tag{2.6}$$

  Oft wird stattdessen auch die Größe $\mathrm{CVaR}'_\alpha = \mathrm{CVaR}_\alpha - \mathrm{VaR}_\alpha$ betrachtet, was dem Exzess-Verlust über dem erwarteten VaR entspricht.

Die oben angeführten Risikomaße werden in der Literatur häufig aufgrund ihrer teilweise falschen Modellannahmen kritisiert. Beispielsweise schlägt [Mandelbrot and Franklin, 2008] eine Ersetzung der konventionellen statistischen Risikomaße durch sog. **fraktale Risikomaße** wie z.B. den **Hurst-Koeffizienten** bzw. der **fraktalen Dimension** vor. Problematisch an diesem Ansatz ist jedoch die schwierige Interpretierbarkeit solcher Risikomaße, da diese nicht ohne weiteres in geldwerte Größen umgerechnet werden können.

Unabhängig von der Gültigkeit der zugrunde liegenden Modellannahmen sollte ein gutes, sprich **kohärentes Risikomaß** $\rho(X)$ nach [Artzner et al., 2001] folgende Eigenschaften erfüllen [McNeil et al., 2005, S. 239 ff.]:

- **Translationsinvarianz**: Für alle $X \in \mathcal{M}$ und alle $l \in \mathbb{R}$ gilt $\rho(X + l) = \rho(X) + l$.
- **Subadditivität**: Für alle X_1,$X_2 \in \mathcal{M}$ gilt $\rho(X_1 + X_2) \leq \rho(X_1) + \rho(X_2)$.
- **Positive Homogenität**: Für alle $X \in \mathcal{M}$ und jedes $\lambda > 0$ gilt $\rho(\lambda X) = \lambda\rho(X)$.
- **Monotonie:** Für X_1, $X_2 \in \mathcal{M}$ mit $X_1 \leq X_2$ gilt fast sicher $\rho(X_1) \leq \rho(X_2)$.

Nicht alle der oben angeführten Risikomaße erfüllen die hier gestellten Anforderungen, in den nächsten Abschnitten wird dies am Beispiel des Value at Risk genauer erläutert.

2.1.2. Value at Risk (VaR)

Der Value at Risk (VaR) wie in Gl. (2.5) definiert ist heute eines der weitverbreitetsten und am häufigsten eingesetzten Risikomaße. Eine der ersten Fälle der Verwendung des VaR zur Risikoabschätzung findet sich in [Markowitz, 1952]. In den 1990er Jahren wurde der VaR schließlich u.a. durch J.P. Morgan's RiskMetrics popularisiert [J.P. Morgan, 1996] und gehört seit jener Zeit zum Standardwerkzeug des quantitativen Risikomanagements. Der VaR ist nach [Artzner et al., 2001] kein kohärentes Risikomaß da er nicht zwangsweise subadditiv ist. In [McNeil et al., 2005, S. 57] findet sich eine Liste der gebräuchlichsten Verfahren zur Berechnung des VaR. Diese Verfahren werden im folgenden eingesetzt, um die später mittels genetischer Programmierung abgeschätzten VaR-Werte mit den von den Standardverfahren generierten Werten zu vergleichen und eine Aussage zur Güte des berechneten VaR zu machen. Folgende Standardverfahren zur Berechnung des VaR werden hierbei berücksichtigt [McNeil et al., 2005, Reiss and Thomas, 2007].

- **Varianz-Kovarianz (VC)**: Standardisierte unbedingte Varianz-Kovarianz Methode unter Annahme multivariater gaußscher Risikofaktoränderungen.

- **Historische Simulation (HS)**: Standardisierte unbedingte historische Simulationsmethode.

- **Varianz-Kovarianz, student-t (VC-t)**: Unbedingte Varianz-Kovarianz Methode, bei welcher eine multivariate t-Verteilung an die Risikofaktoränderungen angepasst wird.

- **Historische Simulation, GARCH**$(1,1)$ **(HS-GARCH)**: Bedingte Version der historischen Simulationsmethode, bei der ein normalverteiltes GARCH$(1,1)$-Modell an die historisch simulierten Risikofaktoränderungen angepasst wird, um die Volatilität der nächsten Risikofaktoränderung abzuschätzen.

- **Varianz-Kovarianz, GARCH**$(1,1)$ **(VC-GARCH)**: Schätzung des VaR mit der Varianz-Kovarianz Methode, wobei ein GARCH$(1,1)$-Modell an die Volatilität angepasst wird.

- **Historische Simulation, EWMA (HS-EWMA)**: Bedingte Version der historischen Simulationsmethode, bei der die Volatilität mittels exponentieller Glättung prognostiziert wird.

- **Varianz-Kovarianz, EWMA (VC-EWMA)**: Varianz-Kovarianz Verfahren mit exponentieller Glättung der Volatilität.

- **Historische Simulation, GARCH-$t(1,1)$ (HS-GARCH-t)**: Historische Simulationsmethode mit GARCH-$(1,1)$ Modellierung der Volatilität unter Annahme t-verteilter Risikofaktoränderungen

- **Varianz-Kovarianz, GARCH-$t(1,1)$ (VC-GARCH-t)**: Varianz-Kovarianz Methode mit multivariater GARCH$(1,1)$ Modellierung der Volatilität und t-verteilten Risikofaktoränderungen

- **Extremwerttheorie (EVT)**: Bedingte Schätzung mit GARCH$(1,1)$ Modellierung unter Zuhilfenahme der Extremwerttheorie.

Die hier aufgeführten Verfahren sind die heute gebräuchlichsten zur Berechnung des VaR und werden im folgenden Abschnitt detailliert erläutert.

2.1.3. Abschätzung des VaR

Im folgenden soll der tagesbasierte Value at Risk sowohl mittels klassischer Verfahren als auch mithilfe genetischer Programmierung für ausgesuchte Aktienwerte modelliert werden. Anschließend soll ein Vergleich zwischen den einzelnen Verfahren durchgeführt werden, um sie auf ihre Praxiseignung zu untersuchen. Getestet wird dabei unter Verwendung von Aktienpreisdaten, welche von [Yahoo! Deutschland GmbH, 2008] bezogen wurden. Diese Daten sind bereits von Dividendenausschüttungen, Aktienaufteilungen und anderen störenden Faktoren bereinigt und können somit direkt zur Analyse herangezogen werden. Der Aktienpreis zum Zeitpunkt $t \in \mathbb{N}$ (wobei t sich auf Tageszeiträume beziet) wird dabei im folgenden als p_t bezeichnet, untersucht wurden jedoch vielmehr die relativen negativen Änderungen dieses Preises, welche gegeben sind als

$$x_t := -\frac{p_t - p_{t-1}}{p_{t-1}} \tag{2.7}$$

Zur Berechnung des Value at Risk $\mathrm{VaR}_\alpha(x_t|x_1,\ldots,x_{t-1})$ zum Zeitpunkt t und zum Sicherheitsniveau α mittels genetischer Programmierung muss zunächst eine Fitnessfunktion sowie eine Menge von Terminal- und Nichtterminalsymbolen definiert werden. Die Wahl der Fitnessfunktion sollte dabei so erfolgen, dass sich darin alle aus Gl. (2.5) ergebenden Eigenschaften des VaR widerspiegeln. Folgende drei Eigenschaften eines guten VaR-Modells lassen sich direkt aus Gl. (2.5) ableiten.

1. Die Anzahl der Überschreitungen $x_t > \mathrm{VaR}_\alpha(x_t|x_1,\ldots,x_{t-1})$ für $t = 1,\ldots,T$ liegt möglichst nahe am erwarteten Wert $P(X > \mathrm{VaR}_\alpha) \cdot T = (1-\alpha)\cdot T$.

2. Der durchschnittliche Wert $1/T \cdot \sum_{t=1}^{T} \mathrm{VaR}_\alpha(x_t|x_1,\ldots,x_{t-1})$ des VaR ist möglichst gering.

3. Die einzelnen Überschreitungen sind zeitlich unabhängig voneinander verteilt.

Die erste Forderung ergibt sich unmittelbar aus Gl. (2.5), die zweite aus den Eigenschaften des Infimums in Gl. (2.5).

Die dritte Forderung folgt ebenfalls direkt aus Gl. (2.5) und kann überprüft werden, indem die Verteilung der Wartezeiten zwischen aufeinanderfolgenden VaR-Überschreitungen untersucht wird. Hierauf wird später detailliert eingegangen werden, zunächst wird jedoch auf eine genauere Betrachtung verzichtet und lediglich eine Fitnessfunktion definiert, welche die beiden ersten Forderungen der obigen Liste berücksichtigt und somit gegeben ist als

$$\begin{aligned} f_{\mathrm{VaR}} \quad := \quad & -\ln\left[1 + \frac{1}{T}\cdot\sum_{t=1}^{T}\mathrm{VaR}_\alpha(x_t|x_1,\ldots,x_{t-1})\right] \\ & -\gamma\ln\left[1+\left((1-\alpha)\cdot T - \sum_{t=1}^{T}\mathbf{1}\left\{x_t > \mathrm{VaR}_\alpha(x_t|x_1,\ldots,x_{t-1})\right\}\right)^2\right] \end{aligned} \tag{2.8}$$

Die Logarithmierung der einzelnen Terme entspricht hierbei einer multiplikativen Minimierung der einzelnen Zielkriterien, wobei der Faktor γ das relative Gewicht

der einzelnen Kriterien festlegt. Die Fitness sehr guter VaR-Modelle liegt somit nahe bei $f_{\mathrm{VaR}} = 0$, schlechte Modelle hingegen weisen Werte $f_{\mathrm{VaR}} \ll 0$ auf.

Als Nichtterminalsymbole werden die gebräuchlichsten arithmetischen Operatoren sowie der Logarithmus $\ln x$, die Exponentialfunktion $\exp x$, das Argumentenminimum $\min(a, b)$, Argumentenmaximum $\max(a, b)$ und der Argumentendurchschnitt $\mathrm{avg}(a, b) = (a + b)/2$ gewählt. Damit gilt $\mathcal{N} = \{+, -, \cdot, /, \exp, \log, \min, \max, \mathrm{avg}\}$. Als Terminalsymbole werden lediglich die gebräuchlichsten statistischen Kennzahlen der jeweiligen Zeitreihe herangezogen. Dies sind

- Blockmaxima (`max`)und Blockminima (`min`): Größte bzw. kleinste Beobachtungen von x_t in einem Zeitfenster h

$$\mathtt{max}_t := \max_{i=t-h-1}^{t-1} x_i \tag{2.9}$$

$$\mathtt{min}_t := \min_{i=t-h-1}^{t-1} x_i \tag{2.10}$$

- Klassische Standardabweichung (`sigma`): Standardabweichung im Zeitfenster h

$$\begin{aligned} \mu_t^h &= \frac{1}{h} \sum_{i=t-h-1}^{t-1} x_i \\ \mathtt{sigma}_t &:= \sqrt{\sum_{i=t-h-1}^{t-1} \left(x_i - \mu_t^h\right)^2} \end{aligned} \tag{2.11}$$

- Klassischer Mittelwert (`mu`): Mittelwert über alle Realisierungen bis zum Zeitpunkt t:

$$\mathtt{mu}_t := \frac{1}{t-1} \sum_{i=1}^{t-1} x_i \tag{2.12}$$

- EWMA-Mittelwert (`mu_ma`) und EWMA-Standardabweichungen (`sigma_ma`):

Exponentiell gewichtete Standardabweichung, rekursiv berechnet als

$$\begin{aligned} \texttt{mu\_ma}_2 &= x_1 \\ \texttt{sigma\_ma}_2^2 &= x_1^2 \\ \texttt{mu\_ma}_t &= \gamma \cdot \texttt{mu\_ma}_{t-1} + (1-\gamma) \cdot x_{t-1} \qquad (2.13) \\ \texttt{sigma\_ma}_t^2 &= \gamma \cdot \texttt{sigma\_ma}_{t-1}^2 + (1-\gamma) \cdot (x_{t-1} - \texttt{mu\_ma}_{t-1})^2 \qquad (2.14) \end{aligned}$$

- GARCH- (`sigma_garch`) und GARCH-t-Standardabweichungen (`sigma_garch_t`): Standardabweichungen, welche mittels eines GARCH(1,1) Modells berechnet werden unter der Annahme normal- bzw. t-verteilter Risikofaktoränderungen. Ein GARCH(1,1) Modell für σ_t^2 hat die Form

$$\sigma_t^2 = a + bx_{t-1}^2 + c\sigma_{t-1}^2 \qquad (2.15)$$

  Da σ_t^2 nicht direkt beobachtet werden kann, müssen die Parameter a, b und c über eine (quasi-) maximum-likelihood Methode (QMLE) [Hamilton, 1994, S. 117 ff.] geschätzt werden. Hierzu wird die Wahrscheinlichkeit[McNeil et al., 2005, S. 150 ff.]

$$\begin{aligned} L(a,b,c;\mathbf{X}) &= \prod_{t=1}^{T} \frac{1}{\sigma_t} g\left(\frac{x_t}{\sigma_t}\right) \qquad (2.16) \\ \sigma_t &= \sqrt{a + bx_{t-1}^2 + c\sigma_{t-1}^2} \end{aligned}$$

  numerisch maximiert, wobei $g(x)$ entweder als Normal- oder als t-Verteilung angesetzt wird. Üblicherweise gilt $\sqrt{a} \ll \sigma_t$, daher wird im folgenden stets $a = 0$ gesetzt, was zu besseren Resultaten beim Einsatz der QMLE-Methode führt.

- Handelsvolumen (`vol`): Das normierte Handelsvolumen, rekursiv definiert mithilfe des Tagesvolumens V_t als

$$\begin{aligned} \overline{V}_2 &= V_1 \\ \overline{V}_t &= \gamma \cdot \overline{V}_{t-1} + (1-\gamma) \cdot V_{t-1} \\ \texttt{vol}_t &= \frac{V_{t-1}}{\overline{V}_t} \qquad (2.17) \end{aligned}$$

Weiterhin werden Gesamtmaxima- (`max_tot`), -minima (`min_tot`) und die Gesamtstandardabweichung (`sigma_tot`) der Zeitreihe verwendet. Es gilt somit $\mathcal{T} = \{\texttt{max}, \texttt{min}, \texttt{mu}, \texttt{mu\_ma}, \texttt{sigma\_ewma}, \texttt{sigma\_garch}, \texttt{sigma\_garch\_t}, \texttt{vol}, \texttt{max\_tot}, \texttt{min\_tot}, \texttt{sigma\_tot}\}$

Weiterhin werden zufallsgenerierte Zahlen im Intervall $[-10, 10]$ als Terminalsymbole verwendet, diese treten bei der Zufallsgenerierung eines Terminals mit einer gewählten Wahrscheinlichkeit von 50 % auf. Abb. 2.1 zeigt exemplarisch einige der oben angeführten Terminalsymbole am Beispiel des `S & P` Aktienindexes. Zusätzlich zu den hier berücksichtigten Terminalsymbolen können prinzipiell auch die mithilfe der klassischen Verfahren berechneten VaR-Werte als Terminals auftreten. Durch die Spezifizierung der Terminal- und Nichtterminalsymbole sowie durch die Festlegung einer Fitnessfunktion ist das genetische Programmierungsproblem vollständig spezifiziert, lediglich die Evolutionsparameter müssen noch festgelegt werden. Für alle im folgenden beschriebenen Untersuchungen wurde dabei $\texttt{mr} = 0.05$, $\texttt{sr} = 0.025$, $\texttt{cs} = 0.5$, $\texttt{ts} = 6$ sowie $\texttt{p} = 1000$ oder $\texttt{p} = 10000$ gewählt.

Ein häufig auftretendes Problem bei der Generierung numerischer Modelle anhand von Testdaten ist die sogenannte **Überanpassung** (engl. **Overfitting**). Überanpassung tritt sehr leicht bei der Minimierung einer Zielfunktion auf, wenn die generierten Modelle hinsichtlich ihrer Parameteranzahl nicht beschränkt sind (was bei der genetischen Programmierung naturgemäß der Fall ist). So erklärt ein überangepasstes Modell zwar scheinbar perfekt die beobachteten und zur Modellerzeugung herangezogenen Daten, versagt aber bei Anwendung auf Datensätze, die nicht mit dem Testdatensatz übereinstimmen. Um Überanpassung zu vermeiden, wurde die Optimierung der mittels GP generierten VaR-Modelle lediglich basierend auf 60 % der vorhandenen Datensätzen durchgeführt. Die verbleibenden 40 % der Datensätze wurden anschließend verwendet, um die generierten Modelle mittels Backtesting auf ihre Tauglichkeit zu überprüfen.

Zur Berechnung des VaR mithilfe der klassischen Verfahren wurden die folgende Algorithmen eingesetzt.

Algorithmus 2.1.1 (Berechnung von $\mathrm{VaR}_\alpha^{\mathrm{HS}}$).

(1) Sei $\{a_1, \ldots, a_n\}$ mit $n = 1 \ldots T$ gegeben, so dass $y_n = x_{a_n}$ und $y_n \geq y_{n+1}$. Dann gilt $y_T = \min_t x_t$ sowie $y_0 = \max_t x_t$

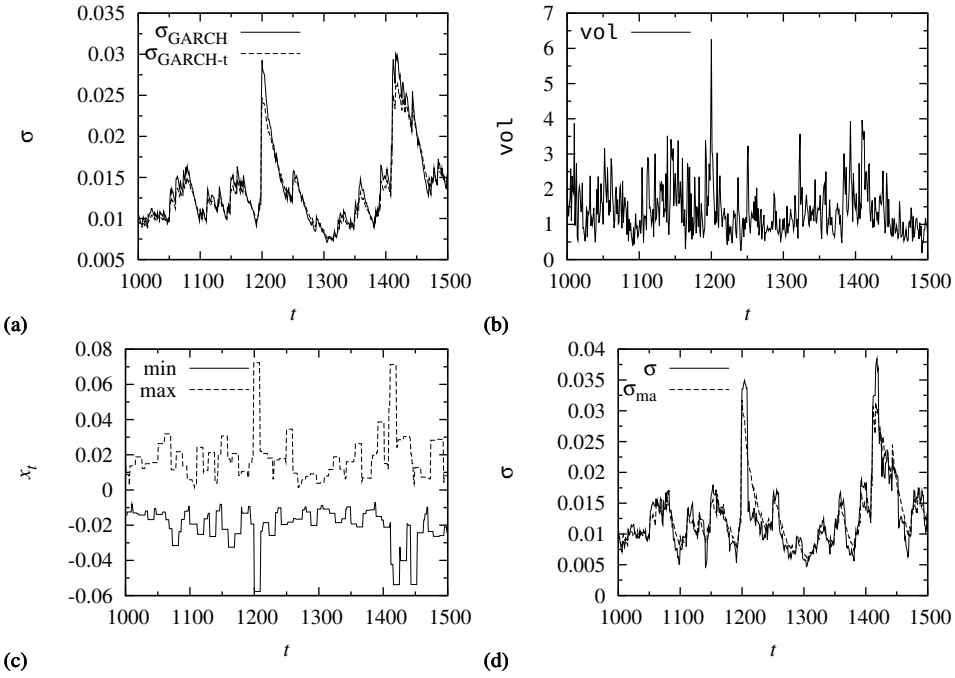


Abbildung 2.1.: Werte einiger Terminalsymbole für den `S & P` Aktienindex. Gezeigt sind (a) die Standardabweichung berechnet mit einem GARCH(1,1) und GARCH-t(1,1) Ansatz nach Gl. (2.16), (b) das Handelsvolumen berechnet nach Gl. (2.17), (c) Blockminima und -maxima berechnet nach Gln. (2.10) und (2.9), (d) die klassisch nach Gl. (2.11) sowie mittels EWMA-Verfahren nach Gl. (2.14) berechnete Standardabweichung.

(2) Berechne $m = \text{int}\left([1-\alpha]\,T\right)$

(3) Setze $\text{VaR}_\alpha^{\text{HS}} = y_m$

Algorithmus 2.1.2 (Berechnung von $\text{VaR}_\alpha^{\text{VC}}$).

(1) Berechne $\hat{\sigma} = \sqrt{\hat{\text{var}}(X)}$ und $\hat{\mu} = \hat{\text{E}}(X)$.

(2) Setze $\text{VaR}_\alpha^{\text{VC}} = \hat{\mu} + \hat{\sigma}\Phi^{-1}(\alpha)$, wobei $\Phi^{-1}(u)$ die inverse kumulative Verteilungsfunktion der Normalverteilung ist.

Algorithmus 2.1.3 (Berechnung von $\text{VaR}_\alpha^{\text{VC-}t}$).

(1) Berechne $\hat{\sigma} = \sqrt{\hat{\text{var}}(X)\,(\nu-2)\,/\nu}$ und $\hat{\mu} = \hat{\text{E}}(X)$.

(2) Setze $\text{VaR}_\alpha^{\text{VC}_\text{t}} = \hat{\mu} + \hat{\sigma} t_\nu^{-1}(\alpha)$, wobei $t_\nu^{-1}(\alpha)$ die inverse kumulative Verteilungsfunktion der student-t Verteilung mit ν Freiheitsgraden ist.

Algorithmus 2.1.4 (Berechnung von $\mathrm{VaR}_\alpha^{\text{HS-GARCH}}$).

(1) Berechne $\hat{\mu} = \hat{\mathrm{E}}(X)$.

(2) Passe ein GARCH(m, n) Modell nach Gl. (2.16) mit normalverteilten Risikofaktoränderungen an die Zeitreihe x_t an, s.d. $\hat{\sigma}^2(x_t|x_{t-1}, x_{t-2}, \ldots) \sim$ GARCH(m, n).

(2) Setze $x'_t = (x_t - \hat{\mu}) / \hat{\sigma}(x_t|x_{t-1}, x_{t-2}, \ldots)$

(2) Berechne $\mathrm{VaR}^{\mathrm{HS}'}$ der normierten Zeitreihenwerte x'_t nach Alg. 2.1.1 und setze $\mathrm{VaR}_\alpha^{\text{HS-GARCH}}(x_t) = \hat{\mu} + \mathrm{VaR}^{\mathrm{HS}'} \cdot \hat{\sigma}(x_t|x_{t-1}, x_{t-2}, \ldots)$

Algorithmus 2.1.5 (Berechnung von $\mathrm{VaR}_\alpha^{\text{HS-EWMA}}$). Die Berechnung von $\mathrm{VaR}_\alpha^{\text{HS-EWMA}}$ erfolgt analog zu Algorithmus 2.1.4, anstatt eines GARCH(m, n)-Modells wird jedoch ein exponentiell gewichteter laufender Durchschnitt nach Gl. (2.14) verwendet um $\hat{\sigma}(x_t|x_{t-1}, x_{t-2}, \ldots)$ zu schätzen.

Algorithmus 2.1.6 (Berechnung von $\mathrm{VaR}_\alpha^{\text{VC-GARCH}}$). Die Berechnung von $\mathrm{VaR}_\alpha^{\text{VC-GARCH}}$ erfolgt analog zu Algorithmus 2.1.2, anstatt der unbedingten empirischen Varianz $\hat{\sigma}^2 = \hat{\mathrm{var}}(X)$ wird jedoch ein GARCH(m, n)-Modell nach Gl. (2.16) an die historischen Werte von x_t angepasst um $\hat{\sigma}(x_t|x_{t-1}, x_{t-2}, \ldots)$ zu schätzen.

Algorithmus 2.1.7 (Berechnung von $\mathrm{VaR}_\alpha^{\text{VC-EWMA}}$). Die Berechnung von $\mathrm{VaR}_\alpha^{\text{VC-EWMA}}$ erfolgt analog zu Algorithmus 2.1.6, jedoch wird exponentielles Glätten zur Schätzung von $\hat{\sigma}(x_t|x_{t-1}, x_{t-2}, \ldots)$ verwendet.

Algorithmus 2.1.8 (Berechnung von $\mathrm{VaR}_\alpha^{\text{VC-GARCH-t}}$). Analog zu Algorithmus 2.1.6 unter Annahme student-t verteilter Risikofaktoränderungen innerhalb des GARCH(m, n)-Modells nach Gl. (2.16).

Algorithmus 2.1.9 (Berechnung von $\mathrm{VaR}_\alpha^{\text{HS-GARCH-t}}$). Analog zu Algorithmus 2.1.4, jedoch Modellierung des GARCH(m, n) Modells nach Gl. (2.16) mit student t-verteilten Risikofaktoränderungen.

Algorithmus 2.1.10 (Berechnung von $\mathrm{VaR}_\alpha^{\mathrm{EVT}}$). Betrachtet wird die **Exzess-Verteilung**

$$F_u(x) = P(X - u \leq x | X > u) = \frac{F(x+u) - F(u)}{1 - F(u)} \tag{2.18}$$

der Überschreitungen von x über einen Schwellenwert u. An diese Exzess-Verteilung wird eine **generalisierte Pareto-Verteilung** der Form[McNeil et al., 2005, S. 275]

$$G_{\xi,\beta}(x) = \begin{cases} 1 - (1 - \xi \cdot x/\beta)^{-1/\xi} & , \quad \xi \neq 0 \\ 1 - \exp(-x/\beta) & , \quad \xi = 0 \end{cases} \tag{2.19}$$

angepasst. Der Value at Risk (und der Conditional Value at Risk) berechnet sich anschließend als

$$\mathrm{VaR}_\alpha^{\mathrm{EVT}} = u + \frac{\beta}{\xi}\left(\left[\frac{1-\alpha}{\bar{F}(u)}\right]^{-\xi} - 1\right) \tag{2.20}$$

$$\mathrm{CVaR}_\alpha^{\mathrm{EVT}} = \frac{\mathrm{VaR}_\alpha^{\mathrm{EVT}}}{1-\xi} + \frac{\beta - \xi \cdot u}{1-\xi} \tag{2.21}$$

Zusätzlich kann GARCH(m,n)- bzw. EWMA-Modellierung zur Berücksichtigung einer zeitabhängigen Volatilität verwendet werden.

Abb.2.2 zeigt exemplarisch die mittels obiger Verfahren berechneten 99% VaR-Werte für den deutschen Aktienindex. Zusätzlich wird der mittels genetischer Programmierung berechnete VaR gezeigt.

2.1.4. Statistische Tests für den VaR

Um zu prüfen, ob die vom GP-Algorithmus bzw. von den klassischen Algorithmen gefundenen VaR-Modelle korrekt sind, kann ein statistischer Test angewandt werden. Aus Gl. (2.5) ergibt sich unmittelbar, dass für eine Realisierung x_t die Wahrscheinlichkeit einer Übertretung der VaR-Schranke als $P\left[x_t > \mathrm{VaR}_\alpha(x_t|x_1,\ldots,x_{t-1})\right] = 1-\alpha$ gegeben ist. Die Variable $Y_t := \mathbf{1}\left[x_t > \mathrm{VaR}_\alpha(x_t|x_1,\ldots,x_{t-1})\right]$ ist somit Bernoulli-verteilt mit $P(Y_t = y_t) = (1-\alpha)^{y_t} \cdot \alpha^{1-y_t}$ und $y_t \in \{0,1\}$. Die Gesamtzahl der Abweichungen $Y := \sum_{t=1}^{T} Y_t$ ist daher binomial-verteilt mit[Reich, 2004]

$$F_Y(y) = P(Y \leq y) = \sum_{k=0}^{\lfloor y \rfloor} \binom{T}{k} (1-\alpha)^k \cdot \alpha^{T-k} \tag{2.22}$$

Mit obiger Gleichung kann bei gegebenem Signifikanzniveau leicht der kritische

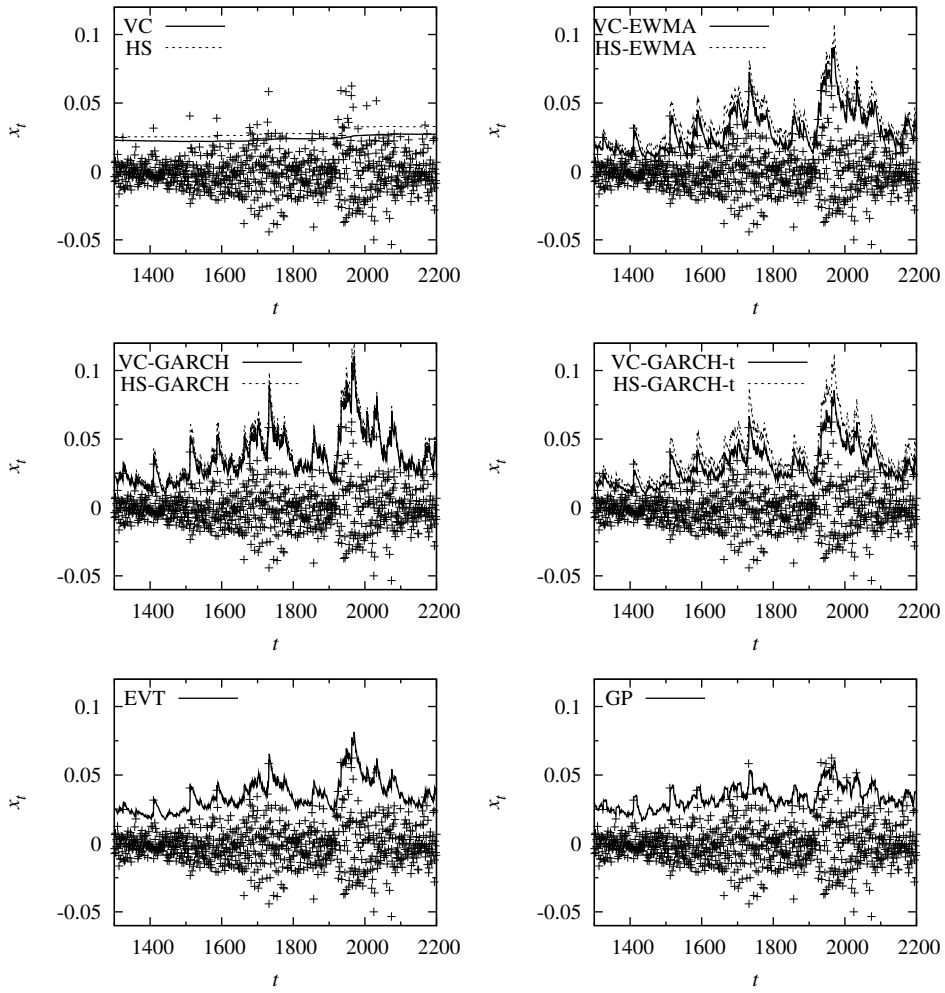


Abbildung 2.2.: Der Value-at-Risk Zoo. Gezeigt sind die mittels Varianz-Kovarianz, historischer Simulation, Extremwerttheorie und genetischer Programmierung berechneten 99% VaR-Modelle für den `deutschen Aktienindex`. Die Volatilität wurde bei den bedingten VaR-Modellen mittels exponentiellem Glätten sowie GARCH(1,1)-Modellierung geschätzt.

(d.h. die im statistischen Sinne gerade noch akzeptable) Wertebereich für positive bzw. negative Abweichungen vom Erwartungswert an der Anzahl an Übertretungen für ein gegebenes Modell und eine gegebene Anzahl an Beobachtungen T berechnet werden, womit letztendlich Hypothesen über korrekte bzw. inkorrekte VaR-Modelle getestet werden können. Dieses als **Backtesting** bekannte Verfahren wird standardmäßig eingesetzt, um die Qualität von VaR-Modellen zu beurteilen. Im Rahmen von Basel II [Basler Ausschuss für Bankenaufsicht, 2004, Philippe, 2002]

wurden Kriterien für die Güte eines VaR-Modells formuliert. Danach ist der obigen Binomialteststatistik ein 5 %-iges Signifikanzniveau zugrunde zu legen. Weist dabei z.B. ein betrachtetes 99 % VaR-Modell mehr als 4 Überschreitungen innerhalb eines Zeitraumes von 250 Tagen auf, so muss der VaR-Wert mittels entsprechender Korrekturfaktoren erhöht werden. In den folgenden Untersuchungen werden die Übertretungen des VaR für die gesamte untersuchte Zeitperiode (und nicht lediglich für einen 250-Tageszeitraum) betrachtet, hiervon abgesehen werden jedoch die gleichen Kriterien zur Beurteilung der Modelle herangezogen.

2.1.5. Wartezeitenverteilung

Wie bereits angesprochen, sollten Übertretungen des VaR zeitlich unabhängig voneinander auftreten. Um dies zu überprüfen, kann ebenfalls ein statistischer Test herangezogen werden. Interessanterweise scheint in der Literatur die Überprüfung eines VaR-Modells meist lediglich hinsichtlich der Anzahl der Überschreitungen zu erfolgen, nicht jedoch anhand der spezifischen zeitlichen Verteilung dieser Überschreitungen. Jedoch gerade eine mangelnde Erfassung der Heteroskedastizität einer betrachteten Zeitreihe kann durch die bloße Betrachtung der Anzahl an Überschreitungen nicht beurteilt werden. Denn selbst ein durch entsprechende Korrekturfaktoren angepasstes VaR-Modell (welches eine akzeptable Anzahl an Überschreitungen liefert) kann noch falsche VaR-Werte produzieren, was sich dann jedoch sehr leicht an der Verteilung der Wartezeiten zwischen aufeinanderfolgenden Überschreitungen erkennen lässt. Dies soll im folgenden kurz erläutert werden. Die Indikatorfunktion einer VaR-Überschreitung

$$\mathbf{1}\left\{x_t > \mathrm{VaR}_\alpha(x_t|x_1,\ldots,x_{t-1})\right\} \tag{2.23}$$

ist Bernoulli-verteilt und somit gilt für die Wahrscheinlichkeit, dass die Wartezeit t_w zwischen zwei aufeinanderfolgenden Überschreitungen den Wert k annimmt die Wahrscheinlichkeit

$$P(t_w = k) = \alpha^k \cdot (1-\alpha) \tag{2.24}$$

Die Verteilungsfunktion der Wartezeitenverteilung lässt sich aus Gl. (2.24) leicht berechnen und ist gegeben als

$$F_w(k) = P(t_w \leq k) = 1 - \alpha^k \tag{2.25}$$

Ist das betrachtete VaR-Modell nun korrekt, so sollte die empirische Verteilungsfunktion der Wartezeiten gegen Gl. (2.25) konvergieren. Dies kann leicht über einen Kolmogorov-Smirnow bzw. einen χ^2-Test geprüft werden, was im folgenden für die mittels genetischer Programmierung erzeugten Modelle sowie für die klassischen VaR-Modelle gezeigt wird. Zur Durchführung des Kolmogorov-Smirnow Tests muss dabei zunächst die empirische Verteilungsfunktion $F_n(k)$ der Wartezeiten ermittelt werden. Anschließend wird die maximale Differenz

$$D_{\max} = \|F_n - F_w\| = \sup_k |F_n(k) - F_w(k)| \tag{2.26}$$

zwischen empirischer und hypothetischer Verteilungsfunktion ermittelt. Der so gefundene Wert wird anschließend bei gegebenem Konfidenzniveau mit dem tabellierten kritischen Wert der Kolmogorov-Smirnow Verteilung verglichen. Überschreitet er diesen, so kann die Hypothese, dass beide Verteilungen identisch sind, abgelehnt werden. Liegen mehr als $n = 40$ Beobachtungen vor, so ist der kritische Wert näherungsweise gegeben als $D_{\text{crit}}(n) \approx 1.36/\sqrt{n}$.

Für den χ^2-Test unterteilt man die beobachteten Wartezeiten zunächst in m Intervalle. n_i bezeichne dann die Anzahl der Beobachtungen im i-ten Intervall. Anschließend berechnet man anhand der hypothetischen Verteilungsfunktion nach Gl. (2.25) und der gegebenen Intervallgrenzen die erwartete Anzahl an Beobachtungen in jedem Intervall n_{i0} und hieraus schließlich die Prüfgröße

$$\chi^2 = \sum_{i=1}^{m} \frac{(n_i - n_{i0})^2}{n_{i0}} \tag{2.27}$$

Sind die einzelnen n_j hinreichend groß, so ist diese Prüfgröße $\chi^2(x, m-1)$ verteilt mit $m-1$ Freiheitsgraden. Durch Vergleich des ermittelten Wertes von χ^2 mit dem kritischen Wert der $\chi^2(x, m-1)$ Verteilung kann so bei gegebenen Konfidenzniveau die Verteilungshypothese überprüft werden. In den folgenden Untersuchungen werden für alle VaR-Modelle die berechneten χ^2-Werte angegeben. Gl. (2.27) eignet sich darüber hinaus auch sehr gut, um die Güte des VaR-Modells zu messen, denn die Verteilung nach Gl. (2.25) "prüft" sowohl auf eine korrekte Übertretungswahrscheinlichkeit $1 - \alpha$, als auch auf die korrekte Verteilung der Wartezeiten zwischen

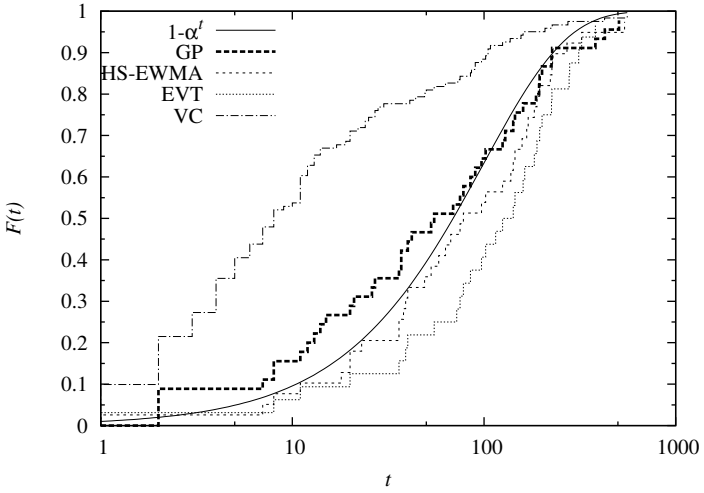


Abbildung 2.3.: Kumulative Verteilungsfunktionen der Wartezeitenverteilungen für verschiedene VaR-Modelle am Beispiel des deutschen Aktienindex. Die durchgezogene Linie zeigt dabei die theoretisch zu erwartende Verteilung.

aufeinanderfolgenden Überschreitungen. Daher kann Gl. (2.27) direkt zur Berechnung der Fitness eines genetisch generierten VaR-Modells herangezogen werden, sollte aber um weitere Kriterien ergänzt werden, um eine gute Konvergenz des GP-Algorithmus sicherzustellen. Die im folgenden generierten VaR-Modelle verwenden dabei lediglich die Fitnessfunktion nach Gl. (2.8). Abb. 2.3 zeigt exemplarisch die Wartezeitenverteilung für verschiedene VaR-Modelle am Beispiel des deutschen Aktienindex.

2.1.6. Test des GP-Algorithmus

Um die Funktionsfähigkeit des GP-Algorithmus zu testen, wurden zunächst numerisch Testzeitreihen mit bekannter Verteilung erzeugt. Diese Testzeitreihen wurden dabei mithilfe der inversen Verteilungsfunktion $F^{-1}(u) : [0,1] \rightarrow \mathbb{R}$ der gewünschten Verteilung generiert. Hierzu wurden zunächst gleichmäßig verteilte Pseudozufallszahlen $u_i \in [0,1]$ berechnet. Anschließend wurden durch Anwendung der inversen Verteilungsfunktion $x_i = F^{-1}(u_i)$ die gewünschten Zufallszahlen mit der entsprechenden Verteilung generiert. Zur Erzeugung der gleichmäßig verteilten Zufallszahlen wurde dabei die Funktion `ran3` der Algorithmensammlung [Press et al., 2002] benutzt.

Für die Testzeitreihen wurden anschließend Werte des 99 % VaR mittels gene-

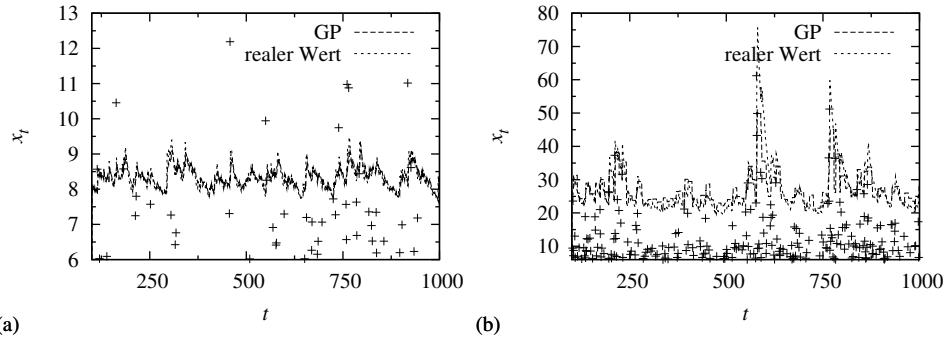


Abbildung 2.4.: Mittels genetischer Programmierung generierte VaR-Modelle für numerische generierte Zeitreihen mit einem GARCH(1,1) Varianzmodell und einer Normal- ($\sigma = 1$, $\mu = 0$) bzw. Extremwertverteilung ($\xi = 0$, $\sigma = 1$). (a) Normalverteilte Zufallsvariable, bestes Modell nach 19 Generationen: `avg(max_tot, sigma_t`^{2} `)` (b) Extremwertverteilte Zufallsvariable, bestes Modell nach 12 Generationen: `avg(max, avg( exp( sqrt( sqrt( log(8.886) +max_tot ) ) ) ,max_tot) )`

tischer Programmierung und mithilfe der klassischen Verfahren berechnet. Diese Werte konnten anschließend mit den tatsächlichen VaR-Werten verglichen werden, welche aufgrund der bekannten Verteilung der Testdaten vorlagen. Zum Testen wurden sowohl **normal-** als auch **extremwertverteilte** Zeitreihenmodelle eingesetzt. Abb. 2.4 zeigt exemplarisch jeweils eine generierte normal- und extremwertverteilte Zeitreihe mit dem realen und dem durch den GP-Algorithmus berechneten VaR. Wie man sieht, stimmt der vom GP-Algorithmus berechnete 99%-VaR in beiden Fällen sehr gut mit dem realen VaR überein. Im Falle der Normalverteilung wurde vom genetischen Algorithmus das VaR-Modell `avg(max_tot, sigma_t`^{2} `)` generiert, welches auf der GARCH-t(1,1) modellierten Standardabweichung beruht. Im Falle der extremwertverteilten Zeitreihe hat der Algorithmus das Modell `avg(max, avg( exp( sqrt( sqrt( log(8.886) +max_tot ) ) ) ,max_tot) )` erzeugt, welches statt der Volatilität nun Blockmaxima zur Prognose des VaR einsetzt. Beide Modelle liefern sehr gute Vorhersagewerte für den VaR mit jeweils 97 bei erwarteten 99 Übertretungen für die normalverteilte Zeitreihe und 91 bei 99 erwarteten Übertretungen für die extremwertverteilte Zeitreihe. Die Abweichungen vom Erwartungswert der Übertretungen liegen dabei beide innerhalb des zu tolerierenden Bereichs bei einem 5% Signifikanzniveau, somit kann die Nullhypothese eines korrekten VaR-Modells nicht zurückgewiesen werden. Die klassischen VaR-Modelle lieferten für die normalverteilte Zeitreihe 83 (HS,VC), 91 (HS-GARCH), 90 (VC-GARCH), 92 (HS-EWMA), 132 (VC-EWMA), 88 (HS-

GARCH-t), 112 (VC-GARCH-t), und 56 (EVT) Überschreitungen sowie für die extremwertverteilte Zeitreihe 86 (HS), 319 (VC), 195 (HS-GARCH), 245 (VC-GARCH), 164 (HS-EWMA), 422 (VC-EWMA), 189 (HS-GARCH-t), 338 (VC-GARCH-t) sowie 121 (EVT) Überschreitungen.

2.1.7. VaR-Modelle für Aktienwerte

Im nächsten Schritt wurde der GP-Algorithmus mit den Preisdaten verschiedener Aktien und Aktienindizes getestet. Die Ergebnisse des Algorithmus wurden anschließend zusammen mit den klassischen VaR-Verfahren auf ihre Qualität hin überprüft. Dabei wurden die betrachteten Modelle sowohl hinsichtlich der Anzahl als auch der Verteilung der VaR-Überschreitungen getestet. Untersucht wurden die drei Aktienwerte `American Express (Amex)`, `Morgan Stanley`, `Freddie Mac` sowie der `Standard & Poor's` Aktienindex und der `deutsche Aktienindex (Dax)`.

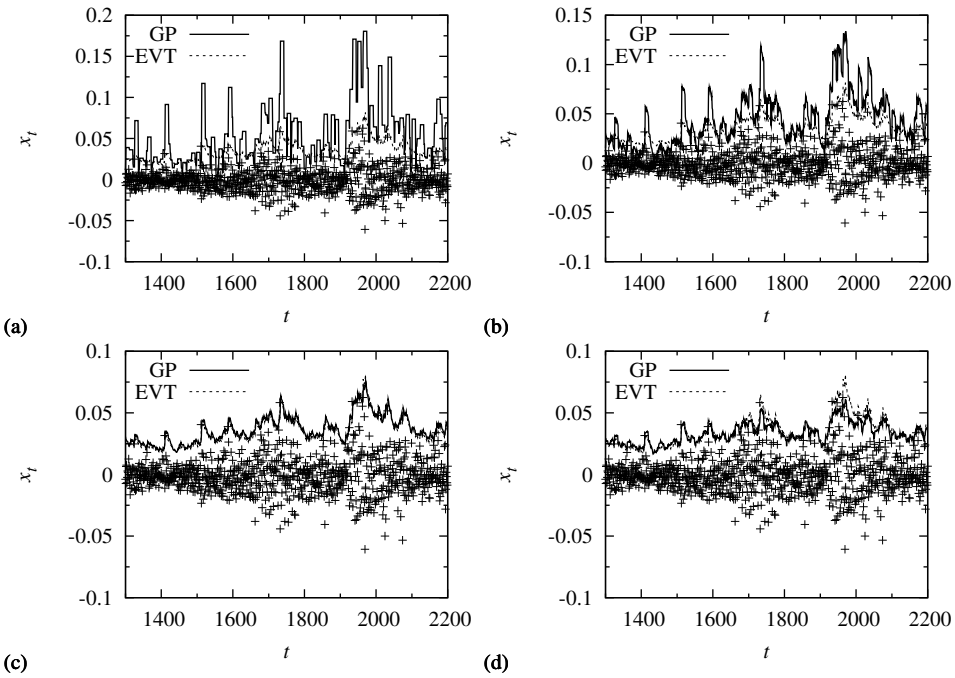


Abbildung 2.5.: Value at Risk (VaR) des deutschen Aktienindex, berechnet mittels genetischer Programmierung (GP). Gezeigt ist das jeweils beste VaR-Modell nach (a)1, (b) 5, (c) 8 und (d) 12 Generationen.

Aktie	Erw.	GP	VC	HS	VC_{EWMA}	HS_{EWMA}
Amex	79 ± 17	*79*(0)	137(58)	119(40)	113(34)	56(23)
M. Stanley	38 ± 12	*47*(9)	82(44)	62(24)	64(26)	39(1)
S & P	39 ± 12	*36*(3)	123(84)	64(25)	83(44)	27(12)
Fr. Mac	49 ± 14	*55*(6)	117(68)	97(48)	75(26)	51(2)
Dax	45 ± 13	*47*(2)	123(78)	74(29)	77(32)	41(4)
Aktie	Erw.	VC_{GARCH}	HS_{GARCH}	$VC_{GARCH-t}$	$HS_{GARCH-t}$	EVT
Amex	79 ± 17	118(39)	84(5)	140(61)	79(0)	64(15)
M. Stanley	38 ± 12	148(110)	58(20)	163(125)	56(18)	31(7)
S & P	39 ± 12	59(20)	24(15)	77(38)	25(14)	22(17)
Fr. Mac	49 ± 14	61(12)	47(2)	73(24)	48(1)	42(7)
Dax	45 ± 13	44(1)	36(9)	77(32)	38(7)	34(11)

Tabelle 2.1.: Schätzungen des 99%-VaR für drei Aktienwerte (American Express, Morgan Stanley, Freddie Mac) und zwei Aktienindizes (Dax, Standard & Poor's). Angeführt sind jeweils die erwartete sowie beobachtete Anzahl an Übertretungen für jedes Verfahren. Die Werte in Klammern geben die Abweichung von beobachteter zu erwarteter Anzahl an Übertretungen an. Ein Konfidenzniveau von $\alpha = 0.05$ wurde dabei für die Berechnung des zulässigen Betrags der Abweichungen zugrunde gelegt.

2.1.7.1. Evolution der VaR-Modelle

Abb. 2.5 zeigt beispielhaft die Entwicklung eines 99 % VaR-Modells für den deutschen Aktienindex im Verlauf des GP-Algorithmus. Abb. 2.5a zeigt das beste VaR-Modell der 1. Generation, 2.5b das beste Modell der 5. Generation, 2.5c der 8. Generation und 2.5d der 12. Generation. An den gezeigten Abbildungen lässt sich dabei sehr anschaulich die Evolution der generierten Modelle ablesen. Zunächst wird vom GP-Algorithmus ein recht grobes VaR-Modell erzeugt, dieses weist eine hohe Volatilität und einen ebenfalls recht hohen Durchschnittswert auf. Dieses grobe Modell wird vom GP-Algorithmus schrittweise durch Mutation und Crossover verbessert. Wie man sieht, sinkt dabei sowohl die Volatilität des VaR-Modells als auch dessen durchschnittlicher Wert. Letztendlich entsteht ein sehr gutes VaR-Modell, welches in großen Teilen dem mittels Extremwerttheorie erzeugten Modell ähnelt und die Heteroskedastizität der betrachteten Zeitreihe sehr gut berücksichtigt.

2.1.7.2. Backtesting der VaR-Modelle

Tab. 2.1 zeigt die Resultate des Backtesting der 99 % VaR-Modelle für die untersuchten Aktientitel. Gezeigt ist jeweils der Erwartungswert der Anzahl an Überschreitungen des korrekten VaR-Modells sowie die innerhalb eines 5 %-igen Konfi-

denzintervalls nach der Binomialverteilung nach Gl. (2.22) maximal zu tolerierende positive bzw. negative Abweichung von diesem Erwartungswert. Weiterhin sind für jedes berechnete VaR-Modell die Anzahl an Überschreitungen angegeben, die geklammerten Werte zeigen hierbei die absolute Abweichung vom Erwartungswert.

Wie sich zeigt, sind die bedingten VaR-Modelle den unbedingten Modellen (HS,VC) klar überlegen. Weiterhin zeigt sich für (bedingte wie unbedingte) Varianz/Kovarianz-Modelle eine signifikante Unterschätzung des zugrunde liegenden Preisrisikos aufgrund der unterschätzten Wahrscheinlichkeitsmasse in den Randbereichen der empirischen Verteilung, den sog. "fat tails". Die bedingte historische Simulationsmethode weist hierbei einen klaren Vorteil auf, da sie sowohl die heteroskedastische Natur der zugrunde liegenden Zeitreihe als auch die empirische Verteilung der (normierten) Zeitreihenwerte berücksichtigt. Daher weisen sämtliche HS-Modelle eine geringere Fehlerrate im Vergleich zu den VC-Modellen auf. Übertroffen werden die bedingten HS-Modelle lediglich durch die bedingte VaR-Modellierung mithilfe der Extremwerttheorie, welche explizit die hohe Wahrscheinlichkeitsmasse im Randbereich der empirischen Verteilungen berücksichtigt. Besser noch als die Extremwertmethode schneidet der genetische Algorithmus hinsichtlich der Erwartungstreue der Überschreitungen ab.

2.1.7.3. Verteilung der VaR-Überschreitungen

Im nächsten Schritt wurde die statistische Verteilung der Überschreitungen für die einzelnen VaR-Modelle untersucht. Dazu wurde die empirische Verteilung der Wartezeiten zwischen aufeinanderfolgenden Überschreitungen untersucht und mit der theoretisch zu erwartenden, geometrischen Verteilung nach Gl. (2.25) verglichen. Der Vergleich erfolgte dabei sowohl anhand des χ^2-Tests nach Gl. (2.27) als auch anhand des Kolmogorov-Smirnow Test gem. Gl. (2.26). Die Ergebnisse der beiden Tests für die einzelnen Aktienwerte sind in Tab. 2.2 sowie Tab. 2.3 zusammengefasst. Wie sich zeigt, lässt sich für die mittels genetischer Programmierung generierten VaR-Modelle die Hypothese der korrekten Verteilung der Wartezeiten für drei der fünf Aktienwerte nicht zurückweisen, lediglich für die beiden Werte `American Express` und `Morgan Stanley` weisen die Teststatistiken leicht überkritische Werte auf. Im allgemeinen ist ersichtlich, dass die mittels GP erzeugten VaR-Modelle hinsichtlich der Wartezeitenverteilung keine schlechteren Eigenschaften als die konventionellen Modelle zeigen und diese oftmals sogar übertreffen. Eindeutig

Aktie	GP	VC	HS	VC_{EWMA}	HS_{EWMA}
Amex	0.93	10.91	97.13	1.48	1.22
M. Stanley	1.28	3.50	0.76	2.38	0.64
S & P	0.40	3.67	0.66	1.30	0.91
Fr. Mac	0.57	4.71	3.15	1.14	0.72
Dax	0.35	5.77	0.71	0.91	0.36
Aktie	VC_{GARCH}	HS_{GARCH}	$VC_{GARCH-t}$	$HS_{GARCH-t}$	EVT
Amex	1.04	0.37	5.90	94.41	—
M. Stanley	8.08	1.45	8.62	1.32	0.51
S & P	0.72	0.24	1.09	0.17	0.88
Fr. Mac	0.85	0.40	1.12	0.39	—
Dax	0.37	0.36	1.03	0.60	0.69

Tabelle 2.2.: Werte der χ^2-Teststatistik für die geschätzten VaR-Modelle. Alle Angaben in Einheiten des kritischen Wertes $\chi^2(0.95, n)$.

lässt sich aufgrund überkritischer Teststatistiken weiterhin das fehlerhafte Verhalten der unbedingten VaR-Modelle für die untersuchten Zeitreihen nachweisen. Auch für andere Verfahren lassen sich Rückschlüsse auf die Gültigkeit der entsprechenden Modelle ziehen. So schneiden wie bereits vorher die bedingten Modelle weitaus besser als die unbedingten ab, weiterhin weisen die mittels historischer Simulation generierten VaR-Modelle im Vergleich mit den VC-Modellen fast durchgängig kleinere Werte der Teststatistiken auf. Die für das Extremwertverfahren ermittelten Teststatistiken sind dabei qualitativ mit denen der anderen bedingten Verfahren vergleichbar und lassen im Vergleich zum GP-Verfahren keinerlei signifikante Rückschlüsse auf die relative Qualität der beiden Verfahren zu.

2.1.7.4. Interpretation der VaR-Modelle

Einer der größten Vorteile der genetischen Programmierung gegenüber anderen Verfahren der künstlichen Intelligenz wie z.B. den neuronalen Netzen ist die unmittelbare Interpretierbarkeit der erzeugten Programme bzw. Modelle. So ist es möglich, die erzeugten Programme in direkt lesbarer Form zu betrachten und ihre Struktur auszuwerten. Dies soll im folgenden für die gefundenen VaR-Modelle durchgeführt werden. Tab. 2.4 zeigt die für die untersuchten Aktienwerte gefundenen Modelle mit der jeweils höchsten Fitness innerhalb der betrachteten Endpopulation. Wie sich zeigt, bringt der GP-Algorithmus vorwiegend VaR-Modelle hervor, welche auf der geschätzten Standardabweichung der Zeitreihen basieren.

Aktie	GP	VC	HS	VC_{EWMA}	HS_{EWMA}
Amex	2.08	3.32	2.80	1.77	1.00
M. Stanley	1.85	3.40	2.59	1.99	1.57
S & P	0.83	4.00	2.21	1.97	1.01
Fr. Mac	0.65	4.25	3.93	1.78	1.18
Dax	0.63	4.35	2.92	1.38	0.62
Aktie	VC_{GARCH}	HS_{GARCH}	$VC_{GARCH-t}$	$HS_{GARCH-t}$	EVT
Amex	1.86	0.71	5.83	2.15	—
M. Stanley	4.91	2.06	5.09	1.82	1.43
S & P	1.06	1.01	1.61	0.90	0.66
Fr. Mac	1.17	1.02	1.54	1.10	2.33
Dax	0.51	0.71	1.39	0.73	0.97

Tabelle 2.3.: Werte $d_{\max}$ der Kolmogorov-Smirnov Teststatistik für die geschätzten VaR-Modelle. Alle Angaben in Einheiten des kritischen Wertes $d_{\max}^{\text{crit}}$.

Einige der Modelle enthalten darüber hinaus zusätzliche Komponenten wie z.B. Blockmaxima und -minima sowie die Mittelwerte verschiedener Kennzahlen. Generell gesehen wirken die gefundenen Modelle für den Betrachter eher exotisch und schwer erklärbar. Gerade im Auffinden solch unkonventioneller Modelle liegt jedoch die Stärke des GP-Algorithmus, der frei von menschlicher Beeinflussung aus bekannten Modellen auf zufällige (durch Mutation) sowie auf deterministische (durch Crossover) Weise neue Modelle erzeugt. Dieses Vorgehen ähnelt dabei sehr stark dem kreativen Lösungsfindungsprozess eines Menschen, der ebenfalls aus bereits vorhandenem Wissen neue Modelle erzeugt und teilweise auch durch Nachdenken oder Intuition dem bekannten Modell neue Elemente hinzufügt. Nachdem die genetischen VaR-Modelle einmal erzeugt wurden, können sie sehr leicht zur Berechnung zukünftiger VaR-Werte verwendet werden. In der Praxis können die gefundenen VaR-Modelle dabei auch nach jedem betrachteten Handelstag -oder nach einem längeren Zeitraum- durch einen erneuten Durchlauf des GP-Algorithmus verbessert und eventuell an neue Gegebenheiten (Strukturbrüche, extreme Ereignisse) angepasst werden, was für die klassischen Modelle nicht möglich ist. Der hieraus entstehende Feedback-Mechanismus ist dabei wesentlich flexibler als die z.B. in [Basler Ausschuss für Bankenaufsicht, 2004] vorgeschlagene, auf der Anzahl an Übertretungen basierende Skalierung des VaR zur Behandlung von Modellverletzungen.

Ein weiterer klarer Vorteil des genetischen Verfahrens ist, dass jederzeit neue

Dax	$\texttt{sigma}/(1 - \ln(\texttt{max\_tot}))$
Morgan Stanley	$\texttt{max\_tot}/(\texttt{max\_tot} - \ln(\texttt{sigma}))$
Freddie Mac	$0.08 + \texttt{ma\_sigma}^2$
S & P	$\text{avg}(0.072, \texttt{sigma\_t}) * \sqrt{\texttt{sigma\_t} + \min(\texttt{sigma\_tot}, \texttt{min})}$
Amex	$\min(\texttt{min}/\texttt{mu\_tot}, \text{avg}(\texttt{sigma}, \texttt{max}))^{\texttt{max}+\texttt{max}+2.906^{\texttt{sigma}}}$

Tabelle 2.4.: Mittels genetischer Programmierung gefundene VaR-Modelle für die untersuchten Aktienwerte.

Terminal- und Nichtterminalsymbole für die Berechnung des VaR eingeführt werden können. Würde sich z.B. herausstellen, dass der VaR einer betrachteten Zeitreihe mit einer bestimmten externen Kennzahl oder Größe auf eine bestimmte Weise korreliert ist, so könnte diese Kennzahl bzw. Größe dem Algorithmus sehr einfach zur Verfügung gestellt und somit zur Modellierung des VaR verfügbar gemacht werden. Solche externen Kennzahlen könnten dabei z.B. Wechselkurse, Klimadaten, Absatzzahlen oder auch Expertenschätzungen sein. Eine Berücksichtigung solcher Kennzahlen bzw. Größen ist im Rahmen der klassischen Verfahren zur Berechnung des VaR dabei sehr schwierig bzw. oft gänzlich unmöglich.

2.1.8. Conditional Value at Risk (CVaR)

Wie bereits angesprochen, ist der VaR kein kohärentes Risikomaß, da er sich nicht zwangsläufig subadditiv verhält und somit Diversifikationseffekte unterschätzen kann. Ausgehend vom VaR kann jedoch der sog. Conditional Value at Risk (CVaR), oft auch als Expected Shortfall (ES) bezeichnet, abgeschätzt werden. Der CVaR ist im Gegensatz zum VaR ein kohärentes Risikomaß und kann ebenfalls sehr leicht interpretiert werden, da er als Erwartungswert des Verlustes bei der Überschreitung des VaR definiert ist. Um den CVaR ausgehend vom VaR zu berechnen werden im folgenden zwei unterschiedliche Verfahren eingesetzt. Diese sind

- die Berechnung des CVaR mittels Anpassung eines linearen Modells an die bestehenden VaR-Werte,
- die Berechnung des CVaR mittels genetischer Programmierung, ausgehend vom bestehenden VaR-Modell.

Die zu optimierende Fitnessfunktion ist in beiden Fällen die negative Abweichung des CVaR vom tatsächlich beobachteten Zeitreihenwert, d.h.

$$f_{\mathrm{CVaR}} = -\sum_{i=1}^{m} (x_{a_i} - \mathrm{CVaR}_{a_i})^2 \tag{2.28}$$

, wobei m die Anzahl der VaR-Überschreitungen bezeichnet und $\{a_1, \ldots, a_m\}$ die Zeitindizes der einzelnen Überschreitungen enthält. Beide Verfahren werden im folgenden kurz erläutert.

2.1.8.1. Berechnung des CVaR als lineare Funktion des VaR

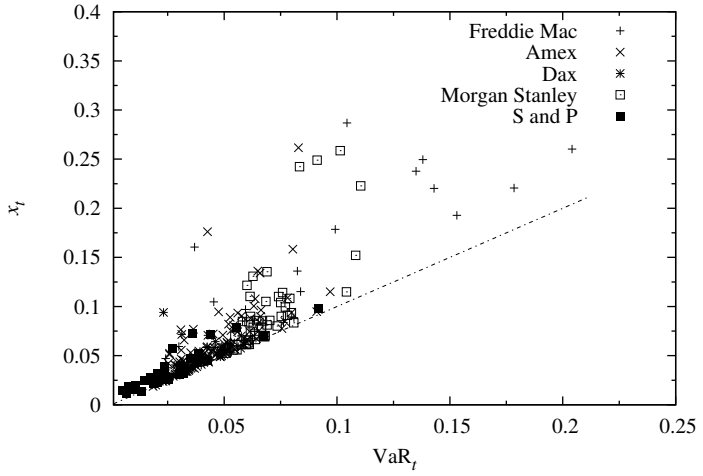


Abbildung 2.6.: Relative Preisänderung x_t bei Überschreitung von $\mathrm{VaR}_\alpha(x_t|x_{t-1}, \ldots)$ als Funktion des selbigen, dargestellt für die fünf untersuchten Aktienwerte.

In diesem Ansatz wird davon ausgegangen, dass sich der CVaR als

$$\mathrm{CVaR}(x_t|\mathrm{VaR}_1, \ldots, \mathrm{VaR}_t, x_1, \ldots, x_{t-1}) = a' \cdot \mathrm{VaR}_t + b' \tag{2.29}$$

schreiben lässt, also eine lineare Funktion des VaR ist. Dieser Ansatz ist für das EVT-Modell sowie die VC-Modelle gerechtfertigt und stellt auch für die HS-Modelle eine gute Näherung dar. Abb. 2.6 zeigt die Werte x_{a_i} der VaR-Überschreitungen der untersuchten Zeitreihen, dargestellt als Funktion von VaR_{a_i}. Wie man sieht, scheint die Linearitätsannahme generell gerechtfertigt zu sein, jedoch weichen die betrachteten Aktienwerte teilweise drastisch unterschiedliche Streuungen in den Überschreitungen auf, was sich aus später an den entsprechenden CVaR-Modellen bemerkbar machen wird. Um die Fitnessfunktion nach Gl. (2.28) zu maximieren

Aktie	GP	VC	HS	$\mathrm{VC}_{\mathrm{EWMA}}$	$\mathrm{HS}_{\mathrm{EWMA}}$
Amex	63.52	77.37	85.61	63.63	69.58
M. Stanley	112.37	151.37	173.27	54.04	64.83
S & P	6.50	13.79	19.62	4.26	5.86
Fr. Mac	472.17	704.39	815.63	770.67	916.0
Dax	14.37	12.68	11.23	12.11	16.73
Aktie	$\mathrm{VC}_{\mathrm{GARCH}}$	$\mathrm{HS}_{\mathrm{GARCH}}$	$\mathrm{VC}_{\mathrm{GARCH-t}}$	$\mathrm{HS}_{\mathrm{GARCH-t}}$	EVT
Amex	56.12	66.76	38.01	51.68	—
M. Stanley	66.20	79.43	63.35	77.46	122.69
S & P	5.84	6.91	7.14	6.95	9.58
Fr. Mac	770.67	916.00	572.51	841.81	1349.32
Dax	16.74	18.21	12.53	17.50	19.70

Tabelle 2.5.: Werte von $-f_{\mathrm{CVaR}}/m$ nach Gl. (2.28), berechnet für verschiedene Aktien und VaR-Modelle.

kann nach Einsetzen von Gl. (2.29) eine einfache OLS-Regression durchgeführt werden. Die Koeffizienten a' und b' ergeben sich hieraus dann zu

$$a' = \frac{\sum_{i=1}^{m} (x_{a_i} - \overline{x}) \cdot \left(\mathrm{VaR}_\alpha(x_{a_i}) - \overline{\mathrm{VaR}}_\alpha\right)}{\sum_{i=1}^{m} \left(\mathrm{VaR}_\alpha(x_{a_i}) - \overline{\mathrm{VaR}}_\alpha\right)} \tag{2.30}$$

$$b' = \overline{x} - a' \cdot \overline{\mathrm{VaR}}_\alpha \tag{2.31}$$

Die Mittelwerte $\overline{x}$ und $\overline{\mathrm{VaR}}_\alpha$ sind dabei nur über die Indizes a_i zu bilden. Die nach obiger Gleichung geschätzten Koeffizienten liefern einen näherungsweise erwartungstreuen Schätzer für den CVaR. Die Größe $-f_{\mathrm{CVaR}}/m$ gibt dabei die mittlere quadratische Abweichung des CVaR von den realisierten Verlustwerten an und eignet sich gut, um die Qualität verschiedener CVaR-Modelle zu vergleichen. Daher wurden für alle generierten VaR-Modelle aus dem letzten Abschnitt die linearen CVaR-Koeffizienten und hieraus die mittleren quadratischen Abweichungen berechnet. Dies erlaubt es, die Qualität der einzelnen VaR-Verfahren auch im Hinblick auf ihre Eignung zur Abschätzung des CVaR zu überprüfen und zu vergleichen.

Tab. 2.5 enthält die berechneten Werte f_{CVaR}/m und erlaubt den Vergleich der gefundenen CVaR-Modelle für alle untersuchten Aktienwerte. Wie man sieht, schneiden die bedingten Modelle wiederum besser als die unbedingten ab, jedoch lässt sich bei Verwendung der OLS-Regression kein signifikanter Qualitätsunter-

schied zwischen den einzelnen bedingten Modellen feststellen.

2.1.8.2. Berechnung des CVaR mit genetischer Programmierung

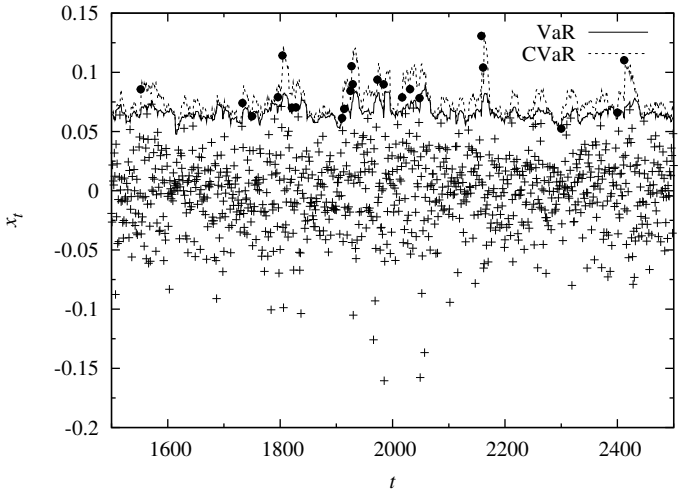


Abbildung 2.7.: Mittels genetischer Programmierung generiertes CVaR-Modell für den Aktienwert Morgan Stanley, basierend auf dem durch genetische Programmierung gefundenen VaR-Modell. Runde Punkte markieren die Überschreitungen des VaR.

Alternativ zum linearen Modell kann der CVaR auch mittels genetischer Programmierung modelliert werden. Dazu wird die Fitnessfunktion nach Gl. (2.28) unter Verwendung der gleichen Terminal- und Nichtterminalsymbole wie im Falle des VaR minimiert. In Abb. 2.7 ist ein mittels genetischer Programmierung erzeugtes CVaR-Modell für den Aktienwert Morgan Stanley gezeigt. Es hat sich im Rahmen der Untersuchungen dabei jedoch gezeigt, dass die linearen CVaR-Modelle im Vergleich zu den mittels genetischer Programmierung generierten Modellen üblicherweise einen geringeren quadratischen Fehler aufweisen, daher wird nicht weiter auf die erstellten Modelle eingegangen. U.U. kann eine CVaR-Modellierung mithilfe des GP-Algorithmus jedoch nützlich sein, wenn neben Gl. (2.28) noch zusätzliche Nebenbedingungen oder Anforderungen durch das CVaR-Modell optimiert werden sollen.

2.2. Risikomessung mit Copulas

In den letzten Abschnitten wurde das Preisrisiko isoliert für einzelne Aktienwerte betrachtet. In der Praxis werden jedoch häufig **Portfolios** aus mehreren Einzelaktien zusammengestellt, um so durch einen möglichen Diversifikationseffekt das Gesamtrisiko des Portfolios gegenüber dem Risiko der Einzelpositionen zu verringern. Der genaue Diversifikationseffekt bei der Zusammenstellung einzelner Aktien zu einem Portfolio hängt dabei jedoch in starkem Maße von der gegenseitigen statistischen Abhängigkeit der einzelnen Aktienpositionen ab. In der Praxis wird die statistische Abhängigkeit zwischen zwei Zufallsvariablen X_1 und X_2 dabei meist über den sog. Korrelationskoeffizienten erfasst, welcher definiert ist als

$$\rho(X_1, X_2) = \frac{E\left(\left[X_1 - \overline{X}_1\right] \cdot \left[X_2 - \overline{X}_2\right]\right)}{\sigma(X_1) \cdot \sigma(X_2)} \tag{2.32}$$

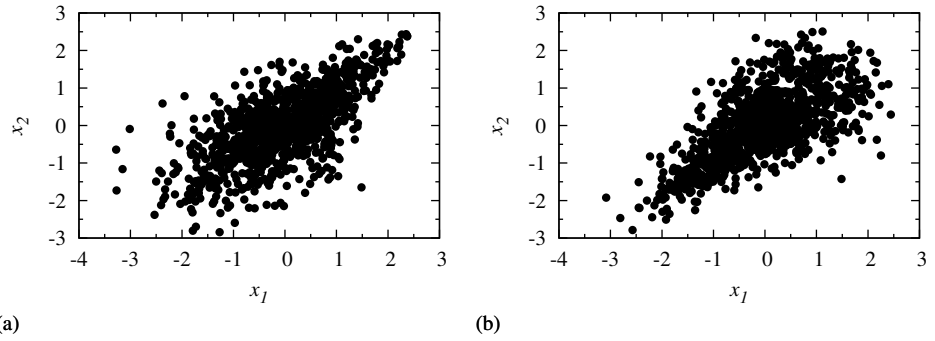


Abbildung 2.8.: (a) Eine mithilfe der Gumbel-Copula mit Parameter $\theta = 2.0$ numerisch generierte Stichprobe zweier Zufallsvariabler mit normalverteilten Rändern. (b) Gleiches Verfahren, jedoch auf Grundlage einer Clayton-Copula mit $\theta = 2.0$. In beiden Fällen ist der Korrelationskoeffizient $\rho \approx 0.68$.

Der Korrelationskoeffizient kann leicht berechnet werden und liegt stets im Wertebereich $\rho(X_1, X_2) \in [-1, 1]$, wobei ein Wert $\rho(X_1, X_2) = 0$ dem Fall statistisch (linear) unabhängiger Variablen entspricht und $\rho(X_1, X_2) = \pm 1$ eine perfekte (anti-)lineare Beziehung zwischen X_1 und X_2 impliziert. Problematisch am Korrelationskoeffizienten ist jedoch die Tatsache, dass er keine Aussagen über die genaue Art der Abhängigkeit zwischen X_1 und X_2 macht. So können zwei völlig unterschiedlich verteilte Paare von Zufallsvariablen problemlos den gleichen Wert des Korrelationskoeffizienten aufweisen. Abb. 2.8 verdeutlicht dies. Gezeigt sind dort zwei Paare unterschiedlich verteilter Zufallsvariablen, welche beide den glei-

chen Wert des Korrelationskoeffizienten aufweisen. Der stärkste Korrelationsbereich für die Variablenwerte aus Abb. 2.8a liegt hierbei im unteren Wertebereich der Verteilungen, wohingegen in Abb. 2.8b der obere Wertebereich die stärkste Korrelation aufweist. Entsprächen die gezeigten Verteilungen der Preisänderung eines Portfolios aus zwei Aktien, so wäre das Gesamtrisiko der beiden Portfolios trotz des identischen Korrelationskoeffizienten sehr unterschiedlich. Als bessere Alternative zum Korrelationskoeffizienten und zur exakteren Beschreibung der Abhängigkeitsstruktur zwischen mehreren statistischen Variablen kommen daher sog. **Copulas**[Joe, 1997, Trivedi and Zimmer, 2005] zum Einsatz. Eine Funktion $C : [0,1]^d \to [0,1], (u_1, \ldots, u_d) \to C(u_1, \ldots, u_d)$ wird dabei als Copula bezeichnet, wenn sie folgende Eigenschaften erfüllt[McNeil et al., 2005, S. 185]

Definition 2.2.1. Eigenschaften von Copulas

(1) Der Wert von $C(u_1, \ldots, u_d)$ nimmt mit allen Variablen u_i zu.

(2) $C(1, \ldots, 1, u_k, 1, \ldots, 1) = u_k$ für alle $k \in \{1, \ldots, d\}$ und $u_k \in [0,1]$.

(3) Für alle $(a_1, \ldots, a_d)$, $(b_1, \ldots, b_d) \in [0,1]^d$ mit $a_i \leq b_i$ gilt

$$\sum_{i_1=1}^{2} \cdots \sum_{i_d=1}^{2} (-1)^{i_1+\ldots+i_d} C(u_{1i_1}, \ldots, u_{di_d}) \geq 0 \tag{2.33}$$

wobei $u_{j1} = a_j$ und $u_{j2} = b_j$ für alle $j \in \{1, \ldots, d\}$ gilt.

Die Bedingungen (1) und (2) lassen sich leicht mit der Interpretation von $C(u_1, \ldots, u_d)$ als Verteilungsfunktion der stetigen Zufallsvariablen $u_i \in [0,1]$ rechtfertigen, Bedingung (3) stellt zusätzlich sicher, dass bei der Auswertung von $C(u_1, \ldots, u_d)$ als Wahrscheinlichkeitsintegral keine negativen Wahrscheinlichkeiten auftreten können. Die Summe in Ungl. (2.33) entspricht hierbei einem d-dimensionalen Wahrscheinlichkeitsintegral. Der linke Teil von Ungl. (2.33) ist somit proportional zu $P(a_1 \leq U_1 \leq b_1, \ldots, a_d \leq U_d \leq b_d)$.

Die Beziehung zwischen Copulas und multivariaten Wahrscheinlichkeitsverteilungen wird durch Sklar's Theorem erfasst.

Theorem 2.2.1 (Sklar 1959). Sei F eine gemeinsame Verteilungsfunktion mit Randverteilungen $F_1, \ldots, F_d$. Dann existiert eine Copula $C : [0,1]^d \to [0,1]$, so dass für

alle $x_1, \ldots, x_d$ in $\overline{\mathbb{R}} = [-\infty, \infty]$ gilt

$$F(x_1, \ldots, x_d) = C(F_1(x_1), \ldots, F_d(x_d)) \tag{2.34}$$

Wenn die Ränder stetig sind ist C eindeutig bestimmt, andernfalls ist C vollständig bestimmt auf $\mathrm{Ran}F_1 \times \ldots \times \mathrm{Ran}F_d$, wobei $\mathrm{Ran}F_i = F_i(\hat{\mathbb{R}})$ das Begrenzungsintervall der Wertemenge von F_i bezeichnet. Umgekehrt ist F aus Gl. (2.34) eine gemeinsame Verteilungsfunktion mit Randverteilungen $F_1, \ldots, F_d$, falls $F_1, \ldots, F_d$ univariate Verteilungsfunktionen sind und C eine Copula ist.

2.2.1. Beispiele

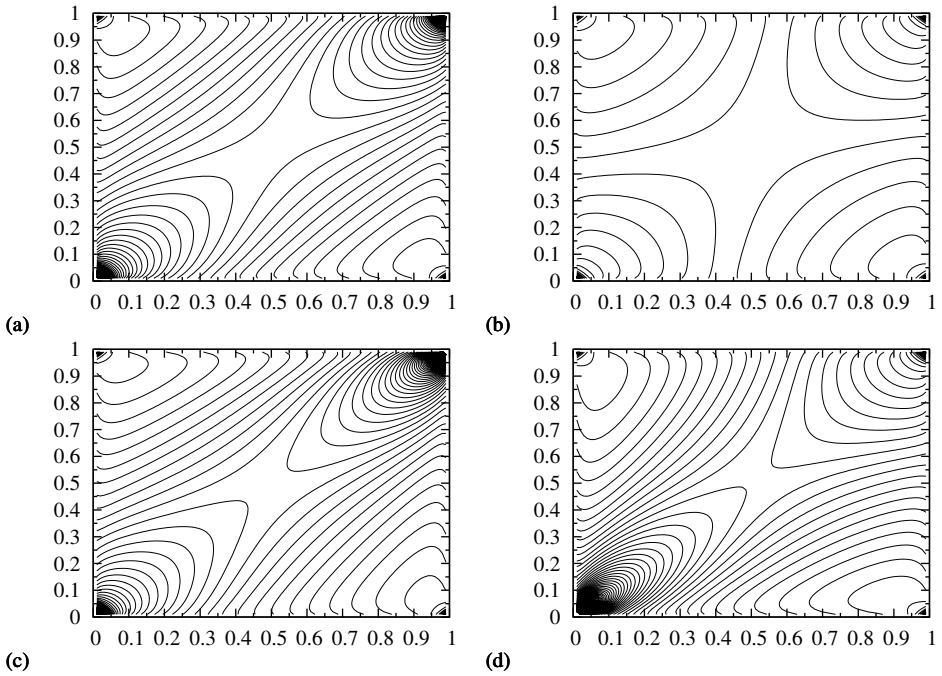


Abbildung 2.9.: (a)Normal-Copula mit $\rho = 0.7$, (b) Frank-Copula mit $\theta = 2.0$, (c) Gumbel-Copula mit $\theta = 2.0$ und (d) Clayton-Copula mit $\theta = 2.0$

Nachfolgend einige Beispiele oft verwendeter Copulas, die auch in den weiteren Untersuchungen verwendet werden.

- Die **Unabhängigkeitscopula** entspricht dem Fall unabhängig verteilter Zu-

fallsvariablen und ist definiert als

$$C_{\Pi}(u_1, \ldots, u_d) = \prod_{i=1}^{d} u_i \tag{2.35}$$

- Die **Gaußcopula** (auch als **Normalcopula** bezeichnet) modelliert multivariat-normalverteilte Zufallsvariablen und hat für $d = 2$ die Integraldarstellung

$$C_{\rho}^{Ga}(u_1, u_2) = \int_{-\infty}^{\Phi^{-1}(u_1)} \int_{-\infty}^{\Phi^{-1}(u_2)} \frac{1}{2\pi(1-\rho^2)^{1/2}} \exp\left\{-\frac{s_1^2 - 2\rho s_1 s_2 + s_2^2}{2(1-\rho^2)}\right\} \mathrm{d}s_1 \mathrm{d}s_2 \tag{2.36}$$

- Eine **t-Copula** besitzt die Darstellung

$$C_{\nu,P}^{t}(\mathbf{u}) = \mathbf{t}_{\nu,\mathbf{P}}\left(t_{\nu}^{-1}(u_1), \ldots, t_{\nu}^{-1}(u_d)\right) \tag{2.37}$$

wobei t_ν der Verteilungsfunktion einer univariaten t-Verteilung und $\mathbf{t}_{\nu,\mathbf{P}}$ der gemeinsamen Verteilungsfunktion eines Vektors $\mathbf{X} \sim t_d(\nu, \mathbf{0}, \mathbf{P})$ mit der Korrelationsmatrix $\mathbf{P}$ entspricht.

- Eine **Archimedische Copula** kann in der Form

$$C_H(u_1, \ldots, u_d) = \Psi^{-1}\left(\sum_{i=1}^{d} \Psi(u_i)\right) \tag{2.38}$$

geschrieben werden, wobei $\Psi(u)$ als Generatorfunktion bezeichnet wird. Wichtige Vertreter der archimedischen Copulaklasse sind die **Clayton-Copula**, welche als

$$C_C(u_1, u_2) = \left(u_1^{-\theta} + u_2^{-\theta} - 1\right)^{-1/\theta} \tag{2.39}$$

gegeben ist, die **Gumbel-Copula** mit der Funktionsdarstellung

$$C_G(u_1, u_2) = \exp\left[-\left((-\ln u_1)^{\theta} + (-\ln u_2)^{\theta}\right)^{1/\theta}\right] \tag{2.40}$$

und die **Frank-Copula** mit der Darstellung

$$C_F(u_1, u_2) = -\frac{1}{\theta} \ln \left(1 + \frac{(\exp(-\theta u_1) - 1)(\exp(-\theta u_2) - 1)}{\exp(-\theta) - 1} \right) \tag{2.41}$$

Die obige Liste stellt lediglich eine kleine Auswahl der gebräuchlichsten Copula-Klassen dar. Eine Vielzahl weiterer Copulamodelle findet sich beispielsweise in [Joe, 1997, S. 139 ff.]. Abb. 2.9 zeigt zum Vergleich die Wahrscheinlichkeitsdichte der Normal-, Frank-, Clayton- und Gumbel-Copula.

2.2.2. Risikomaße & Copulas

Copulas eignen sich sehr gut zur Risikomessung bei statistisch abhängigen Variablen. Interessant für das Risikomanagement ist dabei beispielsweise die Messung der gegenseitigen Abhängigkeit einzelner Aktienwerte innerhalb eines Portfolio und dies insbesondere in den Randbereichen der Verteilungen, wo $u_i \to 1$ bzw. $u_i \to 0$. Gebräuchliche Maße zur Erfassung der Abhängigkeit in diesen Bereichen sind die sog. **Upper Tail Dependence**- sowie **Lower Tail Dependence**-Koeffizienten. Diese sind gegeben als

$$\begin{aligned} \lambda_l &= \lim_{q \to 0^+} \frac{P(X_2 \leq F_2^{\leftarrow}(q), X_1 \leq F_1^{\leftarrow}(q))}{P(X_1 \leq F_1^{\leftarrow}(q))} \\ &= \lim_{q \to 0^+} \frac{C(q,q)}{q} \qquad (2.42) \\ \lambda_u &= \lim_{q \to 1^-} \frac{\hat{C}(1-q, 1-q)}{1-q} \\ &= \lim_{q \to 0^+} \frac{\hat{C}(q,q)}{q} \qquad (2.43) \end{aligned}$$

und können direkt aus einer gegebenen Copula berechnet werden, weisen jedoch nicht zwangsläufig einen finiten Wert auf. In Gl. (2.43) tritt die sog. **Survival Copula** auf, welche gegeben ist als $\hat{C}(1-q_1, 1-q_2) = 1 - q_1 - q_2 + C(q_1, q_2)$. Wie sich aus Gl. (2.42) ersehen lässt, gibt der Coefficient of Lower Tail Dependence (CLTD) die Wahrscheinlichkeit an, dass beide Variablen X_1 und X_2 einen gegebenen Wert $F_1^{\leftarrow}(q)$ bzw. $F_2^{\leftarrow}(q)$ unterschreiten, wobei $q \to 0$ gilt. Der Koeffizient ist dabei normiert auf die Wahrscheinlichkeit $P(X_1 \leq F_1^{\leftarrow}(q))$. Der CLTD misst also die Stärke

der gegenseitigen Abhängigkeit der beiden Zufallsvariablen im Falle extremer Ausprägungen. Der Coefficient of Upper Tail Dependence (CUTD) lässt sich analog interpretieren. Weitere gebräuchliche Abhängigkeitsmaße, welche sich direkt aus einer Copula berechnen lassen, sind z.B. **Kendall's tau** sowie **Spearman's rho**, gegeben als

$$\tau = 4 \int C \, dC - 1 \tag{2.44}$$

$$\rho_S = 12 \int \int \hat{C}(u, v) \, du \, dv - 3 \tag{2.45}$$

Wegen der direkten Berechenbarkeit der oben angesprochenen Risikomaße bei Vorliegen eines Copulamodells ist es im Kontext der Risikoanalyse äußerst interessant, ein passendes Copulamodell mithilfe von gegebenen Zeitreihendaten zu generieren. Dies kann, wie in den folgenden Abschnitten gezeigt wird, sehr effizient mittels genetischer Programmierung erfolgen.

2.2.3. Copula-Modellierung mit genetischer Programmierung

Im folgenden soll gezeigt werden, wie mittels genetischer Programmierung Copulamodelle generiert werden können. Dazu ist es zunächst nötig, ein passendes Qualitätskriterium für die Anpassung eines Copulamodells an einen gegebenen Datensatz zu bestimmen. Dieses Qualitätskriterium kann dann als Fitnessfunktion im GP-Algorithmus verwendet werden. Ein sehr robustes und leicht zu implementierendes Gütemaß ist dabei die sog. Log-Likelihood Funktion. Diese gibt den Logarithmus der Wahrscheinlichkeit an, mit welcher die untersuchte empirische Stichprobe vom einem gewählten Copulamodell generiert wurde. Seien im folgenden $\hat{\mathbf{U}}_i = (u_{i1}, \ldots, u_{id})$ mit $i = 1 \ldots T$ die beobachteten Werte der untersuchten empirischen Verteilungsfunktionen $u_{ij} = \hat{F}_j^{\leftarrow}(x_{ij})$. Die Likelihood-Funktion für ein Copula-Modell C_θ mit Parametervektor θ ist dann gegeben als

$$\ln L(\theta; \hat{\mathbf{U}}_\mathbf{1}, \ldots, \hat{\mathbf{U}}_\mathbf{t}) = \sum_{i=1}^{T} \ln C_\theta(\hat{\mathbf{U}}_\mathbf{i}) \tag{2.46}$$

Diese Gleichung kann direkt als Fitness-Funktion für den GP-Algorithmus herangezogen werden. Um entsprechende Copula-Modelle mittels genetischer Program-

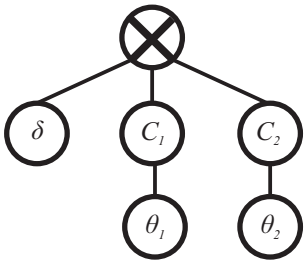


Abbildung 2.10: Der genetische Mischoperator.

mierung zu generieren werden dabei sogenannte **Misch-Copulas** eingesetzt. Diese werden folgendermaßen definiert: Seien $C_1(u_1, \ldots, u_d)$ sowie $C_2(u_1, \ldots, u_d)$ zwei Copulas. Die Mischcopula

$$C_m(u_1, \ldots, u_d) = \delta \cdot C_1(u_1, \ldots, u_d) + (1 - \delta) \cdot C_2(u_1, \ldots, u_d) \tag{2.47}$$

mit $\delta \in [0, 1]$ ist dann ebenfalls eine Copula und erfüllt alle Eigenschaften nach Def. 2.2.1. Der Mischprozess kann beliebig oft wiederholt werden, um immer komplexere Copulamodelle zu erzeugen. Um den Mischoperator nach Gl. (2.47) innerhalb des GP-Algorithmus umzusetzen, muss dieser zunächst als Nichtterminal modelliert werden. Dieses zu modellierende Nichtterminal enthält dabei einen Mischparameter δ sowie die beiden zu mischenden Copulas C_1 und C_2 als Parameter. Abb. 2.10 zeigt die Baumstruktur des Mischoperators. Der Operator besitzt eine sogenannte syntaktisch eingeschränkte Struktur, d.h. seine Unterknoten können nicht beliebig mit Terminal und Nicht-Terminalsymbolen besetzt werden. Vielmehr muss der erste Unterknoten des Operators stets mit einer reellen Zahl $\delta \in [0, 1]$ besetzt werden, wohingegen die beiden verbleibenden Knoten zwingendermaßen mit Nicht-Terminalsymbolen (d.h. Copulamodellen oder weiteren Mischoperatoren) besetzt werden müssen. Auch Copulamodelle selbst sind dabei syntaktisch beschränkt, da als Parameter nur reelle Zahlen und nicht wiederum andere Copulas auftreten dürfen. Um all diese syntaktischen Beschränkungen umzusetzen ist es nötig, spezielle Crossover-, Mutations- und Shrink-Operatoren für den GP-Algorithmus zu implementieren. Als für die Mischung einzusetzenden Copulamodelle wurden die Normalcopula nach Gl. (2.36) sowie die Gumbel-, Clayton- und Frank-Copulas nach Gln. (2.39. . . 2.41) gewählt. Zusammenfassend wurde also eine Nichtterminalmenge $\mathcal{N} = \{\texttt{mix}, \texttt{normal}, \texttt{clayton}, \texttt{gumbel}, \texttt{frank}\}$ definiert, Als

Terminalsymbole wurden lediglich reelle Zahlen im Intervall $[-10, 10]$ definiert.

2.2.4. Test des GP-Algorithmus

Copula/Modell	Normal	Gumbel	Clayton	Frank	GP
Normal	**71.92**	—	160.62	97.07	72.62
Gumbel	143.37	**99.74**	277.62	164.51	101.46
Clayton	267.13	311.16	**97.09**	195.20	100.26
Frank	118.94	139.72	125.91	**114.26**	113.57
Mischung	307.31	292.25	292.58	302.19	**105.51**

Tabelle 2.6.: Werte des χ^2-Tests für verschiedene Copula-Testdatensätze (linke Spalte) bei Anpassung unterschiedlicher Copulamodelle (obere Zeile).

Um die Funktionsfähigkeit des GP-Algorithmus zur Copulamodellierung zu testen, wurden erneut Datensätze mit bekannter Verteilung generiert. Die Erzeugung der bivariat verteilten Zufallszahlen mit gegebenen Randverteilungen $F_1(x)$ und $F_2(x)$ sowie gegebener Copula $C(u_1, u_2)$ erfolgte dabei nach folgendem Algorithmus.

Algorithmus 2.2.1 (Erzeugung bivariater Zufallszahlen mit der Copulamethode).

1. Wähle einen Intervallparameter du.

2. Für $i = 1 \dots n$ wiederhole folgenden Ablauf:

   Erzeuge Pseudozufallszahlen u_i, v_i sowie h_i mit gleichmäßiger Verteilung auf dem $[0, 1]$-Intervall.

   Berechne $p_i = C(u_i+du, v_i+du)+C(u_i, v_i)-C(u_i+du, v_i)-C(u_i, v_i+du)$.

   Ist $h_i < p_i$, so berechne die Werte $x_i = F_1(u_i)$ und $y_i = F_2(v_i)$ und speichere diese. Ansonsten gehe zu 2.

Die vorgestellte Methode beinhaltet einen Fehler der Größenordnung $\sqrt{du}$ in der Verteilung der erzeugten Zufallszahlen, du sollte daher möglichst klein gewählt werden.

Für die so generierten Testdatensätze wurden anschließend mittels genetischer Programmierung Copulamodelle erzeugt. Ebenfalls wurden Standardcopulas (Normal-, Clayton-, Gumbel- und Frank-Copula) an die Testdaten angepasst. Zur

Bewertung der relativen Güte der generierten Modelle wurden die Testdaten anschließend in Intervalle $\mathbf{I}_{ij} = [0.1 \cdot i, 0.1 \cdot (i+1)] \times [0.1 \cdot j, 0.1 \cdot (j+1)]$ mit $i, j = 0 \ldots 9$ unterteilt und mittels χ^2-Test mit den aus den geschätzten Modellen zu erwartenden Häufigkeiten verglichen. Der kritische Wert der χ^2-Teststatistik war dabei stets gegeben als $\chi^2(0.95, 99) \approx 123.23$.

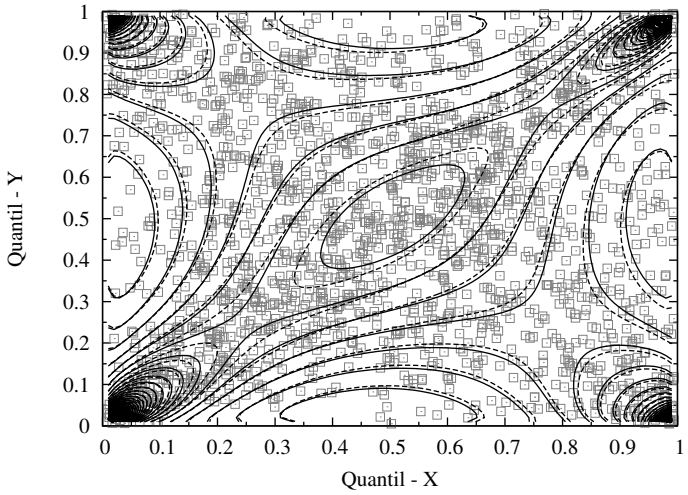


Abbildung 2.11.: Verteilung der Quantilwerte der generierten Testdaten (Quadrate). Die durchgezogene Linie zeigt die vorgegebene Wahrscheinlichkeitsverteilung der Testcopula, die gestrichelte Linie die Wahrscheinlichkeitsverteilung der vom GP-Algorithmus gefundenen Copula.

Tab. 2.6 zeigt die Ergebnisse dieses Tests, für den jeweils $n = 2000$ Variablenwerte gemäß dem in der linken Spalte angegebenen Copula-Modell erzeugt wurden. Wie man sieht, liegen die Teststatistiken der Schätzungen der jeweils passenden Modelle (siehe Zeilenköpfe) zu den Testdaten stets innerhalb des akzeptablen Bereichs der χ^2-Statistik, was natürlich zu erwarten war. Darüber hinaus schafft es der GP-Algorithmus (rechte Spalte), für alle gegebenen Testdatensätze Modelle mit unterkritischem χ^2-Wert zu generieren. Neben elementaren Copulas wurde dabei auch eine Mischcopula der Form `mix(0.5,normal(u,v,-0.7), mix(0.4,gumbel(u,v,4.0),clayton(u,v,2.0)))` als Testmodell verwendet. Diese kann mit den elementaren Copula-Modellen aus der linken Spalte nicht korrekt beschrieben werden. Der genetische Algorithmus hingegen schafft es, wiederum ein gültiges Modell für diesen Testdatensatz zu erzeugen. Der Test zeigt somit, dass genetische Programmierung für eine große Bandbreite an Daten gültige Copulamodelle erzeugen kann.

2.2.5. Evolution der Copulamodelle

Abb. 2.12 zeigt beispielhaft die Evolution eines Copulamodells im Verlauf des GP-Algorithmus, dargestellt für die oben als Testmodell verwendete Mischcopula. Dabei wird vom Algorithmus zunächst ein recht einfaches Copulamodell erzeugt, welches jedoch bereits Teile der positiven und negativen Korrelationen der Testdaten berücksichtigt (Abb. 2.12a). Dieses Ausgangsmodell wird anschließend durch Mutation und Crossover weiter verbessert. Dabei werden nach und nach die in den Testdaten vorhandenen Abhängigkeiten in das Modell eingeführt, bis schließlich ein fast optimales Modell entstanden ist (Abb. 2.12d). Abb. 2.11 zeigt das endgültige mittels GP generierte Copulamodell sowie das vorgegebene Testmodell zum Vergleich.

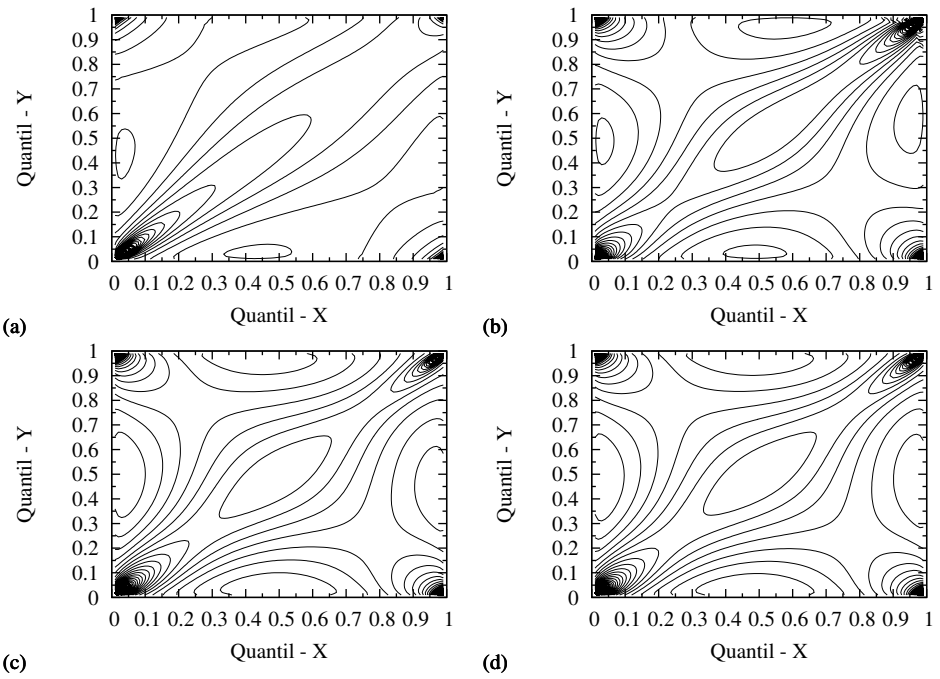


Abbildung 2.12.: Die Evolution des Copulamodells beim Ablauf des GP-Algorithmus. Gezeigt ist die Wahrscheinlichkeitsverteilung der besten Copula der (a)1., (b)10., (c) 20. und (d) 30. Generation.

2.2.6. Copula-Modelle für Aktienportfolios

Im folgenden werden mittels genetischer Programmierung Copulamodelle für Portfolios bestehend aus zwei Einzelaktien/Aktienindizes generiert

und auf ihre Qualität hin untersucht. Dazu wurden verschiedene Aktienwerte (`Freddie Mac`, `Fannie Mae`, `Dax`, `Dow Jones`, `S & P`, `Microsoft`, `IBM`, `American Express`, `Citigroup`, `BEARX`) ausgewählt und aus diesen wurden anschließend Portfolios aus jeweils zwei Aktien erstellt. Wiederum wurden dabei die entsprechenden Preisdaten von [Yahoo! Deutschland GmbH, 2008] bezogen. Zur Aufbereitung der Daten mussten zunächst jeweils korrespondierende Preisänderungen x_t und y_t der beiden ausgewählten Aktien anhand der Datumsangaben in den einzelnen Datensätzen zusammengeführt werden. Mithilfe dieser zusammengeführten Datensätze wurden anschließend die empirischen Verteilungsfunktionen $\hat{F}_1^{\leftarrow}(x)$ sowie $\hat{F}_2^{\leftarrow}(y)$ ermittelt. Ausgehend hiervon wurde schließlich ein entsprechender Copuladatensatz durch die Transformation $(x_i, y_i) \rightarrow (\hat{F}_1^{\leftarrow}(x_i), \hat{F}_2^{\leftarrow}(y_i))$ erzeugt, welcher dann als Grundlage für die zu schätzenden Copulamodelle diente. Es sei angemerkt, dass durch die Verwendung der empirischen Verteilungsfunktionen u.U. Verzerrungen bei der Erzeugung des Copuladatensatzes auftreten können, diese werden im folgenden jedoch als klein angenommen und können bei großen Datensätzen auch oft vernachlässigt werden.

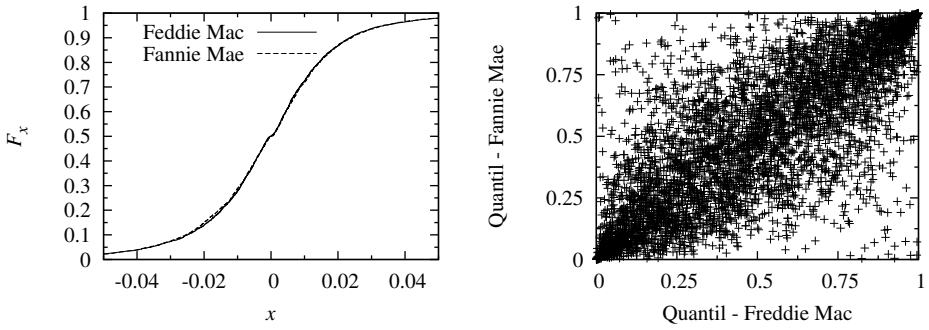


Abbildung 2.13.: (a) Kumulative empirische Wahrscheinlichkeitsfunktion der relativen Preisänderungen der Aktienwerte `Freddie Mac` und `Fannie Mae`. (b) Verteilung von $(\hat{F}_1^{\leftarrow}(x_i), \hat{F}_2^{\leftarrow}(y_i))$, berechnet auf Grundlage der empirischen Verteilungsfunktionen aus (a).

Abb. 2.13 illustriert die beschriebene Vorgehensweise. Abb. 2.13a zeigt dabei die empirischen Verteilungsfunktionen der Preisänderungen der beiden beispielhaft untersuchten Aktienwerte `Freddie Mac` und `Fannie Mae`, Abb. 2.13b zeigt die hieraus gewonnene Verteilung der entsprechenden Quantilwerte der jeweiligen Preisänderungen. Wie man leicht erkennen kann, weist das untersuchte Portfolio eine sehr starke Korrelation für extreme positive bzw. negative Preisänderungen auf. Solch eine erhöhte Korrelation im Randbereich der Verteilungen ist für die Risikoanalyse

von großer Bedeutung und kann mithilfe eines passenden Copulamodells und den vorher besprochenen Risikomaßen abgeschätzt werden.

Copula/Modell	Normal	Gumbel	Clayton	Frank	GP
Freddie Mac / Fannie Mae	739.03	350.61	922.20	426.71	129.159
Dax / BEARX	192.95	—	934.16	942.12	98.36
Dax / Dow Jones	1160.37	180.31	358.67	295.51	135.78
S & P / Dow Jones	—	—	1137.81	617.64	167.17
Microsoft / IBM	289.10	222.72	467.99	220.79	101.62
Amex / Citigroup	474.92	369.14	778.51	477.03	208.06
Dow / BEARX	3040.43	1004.84	3033.63	3028.25	136.27

Tabelle 2.7.: Werte der χ^2-Teststatistik der generierten Copulamodelle für die untersuchten Aktienportfolios. Die rechte Spalte enthält die Ergebnisse des GP-Algorithmus, die anderen Spalten die Testwerte für die Standardmodelle.

Tab. 2.7 zeigt die Ergebnisse der durchgeführten Untersuchung. Für alle sechs Portfolios wurden mehrere klassische, einparametrige Copulamodelle angepasst, ebenso wurde ein Copulamodell mithilfe der genetischen Programmierung generiert. Für jedes dieser Modelle wurde der resultierende Wert der χ^2-Statistik angegeben. Wie sich zeigt, weisen alle klassischen Copulamodelle überkritische χ^2-Werte auf. Die mittels genetischer Programmierung erzeugten Modelle hingegen weisen durchweg geringere (jedoch trotzdem teilweise überkritische) χ^2-Werte auf und modellieren die Abhängigkeitsstruktur der untersuchten Aktienportfolios damit weitaus besser als die klassischen Modelle. Vom GP-Algorithmus werden dabei vorwiegend Lösungen generiert, die einer Mischung zwischen einer oder mehreren Normalcopulas mit einer oder mehreren archimedischen Copulas entsprechen. Für das `Dax / BEARX` Portfolio, welches eine starke negative Korrelation zwischen den beiden Aktienwerten aufweist, enthält die gefundene Lösung `mix(0.5364, normal(-0.686), mix(0.5364, normal(-0.686)), mix(0.0871, gumbel(4.868, mix(0.8135, mix(0.9846, frank(0.714), normal(0.0)),normal(0.0)))))` mehrere Normalcopulas mit negativer Korrelation sowie eine sehr geringe Beimischung von Copulas mit positiver Korrelation. Der Aktienwert `BEARX` entspricht dabei einem Fonds, der vorwiegend Short-Positionen enthält und somit eine Antikorrelation zu den meisten Aktienindizes aufweist. Diese Antikorrelation kann im Rahmen der ausgewählten einparametrigen Copulamodelle lediglich durch die Normalcopula modelliert werden, nichts desto trotz können einzelne Eigenschaften

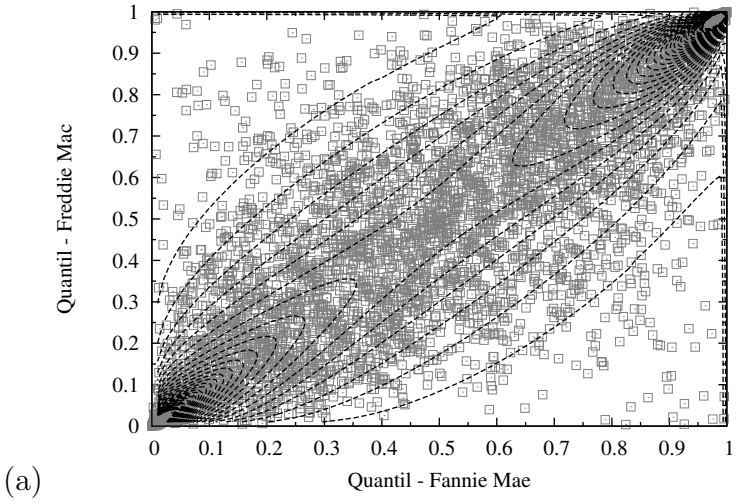
der empirischen Verteilung durch die Beimischung von Copulas mit positiver Korrelation besser modelliert werden. Das gefundene Copulamodell für das Portfolio `Freddie Mac / Fannie Mae`, `mix(0.8468, normal(0.814), mix(0.3545, mix(0.0109,clayton(6.862),gumbel(7.376)),normal(0.0)))` enthält hingegen lediglich Copulas mit positiver Korrelation und beinhaltet neben Normalcopulas auch Gumbel- bzw. Claytoncopulas als Beimischung. Abbn. 2.14 und 2.15 zeigen für die beiden Portfolios `Freddie Mac / Fannie Mae` sowie `Dow Jones / BEARX` detailliert die Verteilung der Quantilwerte, die Wahrscheinlichkeitsdichte der ermittelten Copula sowie die Verteilung der relativen Preisänderungen der Portfolios. Man beachte bei beiden Werte die große Anzahl der extremen Preisänderungen, welche bspw. durch die konventionelle Modellierung im Rahmen der Portfoliotheorie[Elton et al., 2006] unter gaußscher Verteilungsannahme nicht erfasst werden könnte und für die eine Risikoanalyse unter Zuhilfenahme eines Copulamodells damit sehr sinnvoll ist.

Eine weitere Stärke des genetischen Verfahrens ist dabei, dass in der Praxis neben den in dieser Arbeit für die Modellierung verwendeten recht einfachen Copulamodellen viele weitere, (gegenenfalls hunderte) ein- und mehrparametrige Copulamodelle berücksichtigt werden können, was die Mächtigkeit des Verfahrens vervielfacht. Daher ist davon auszugehen, dass ein auf Praxisanforderungen zugeschnittenes und entsprechend optimiertes Verfahren der genetischen Programmierung noch weitaus bessere Ergebnisse als die hier vorgestellten liefern kann.

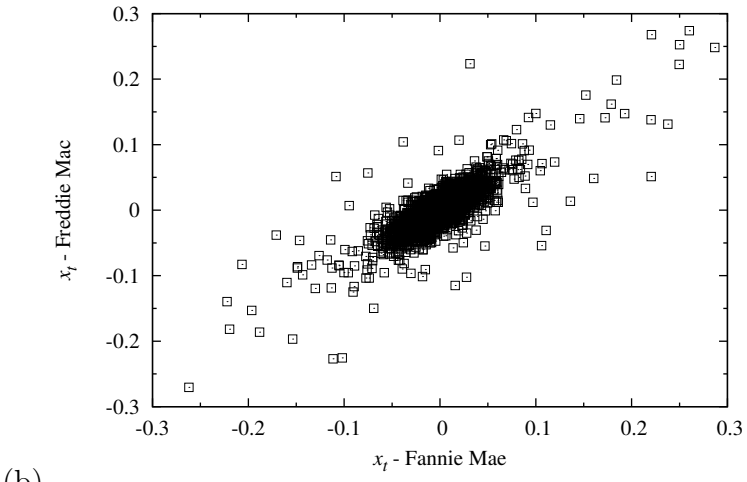
2.2.7. Abschätzung von Risikomaßen

Aus den generierten Copulamodellen können Risikomaße wie z.B. die Tail-Koeffizienten nach Gln. (2.42. . .2.43) abgeschätzt werden. Da sämtliche generierten Modelle einfache Mischungen einparametriger Copulas sind, können diese Koeffizienten analytisch, aber auch numerisch abgeschätzt werden. Auch hierbei ist es von großem Vorteil, dass die mittels genetischer Programmierung erzeugten Copulamodelle leicht interpretierbar sind. So ist das generierte Copulamodell für das Portfolio `Microsoft / IBM` beispielsweise gegeben als `mix(0.3627,normal(0.0), mix(0.5676,frank(6.374), mix(0.0288,gumbel(2.36),gumbel(2.148))))`. Der Upper-Tail-Koeffizient wurde für dieses Modell numerisch abgeschätzt zu $\lambda_u \approx 0.163$. Analytisch kann der Koeffizient auch recht leicht berechnet werden, hierzu ist zu beachten, dass für die

Gumbel- und Normalcopula $\lambda_u = 0$ sowie für die Clayton-Copula $\lambda_u = 2 - 2^{1/\theta}$ gilt. Damit ergibt sich unter Berücksichtigung der Mischfaktoren ein Wert von $\lambda_u = (1-0.3627)\cdot(1-0.567)\cdot(0.0288\cdot(2-2^{1/2.36})+(1-0.0288)\cdot(2-2^{1/2.148})) = 0.17$, was recht nahe am numerischen Ergebnis liegt.

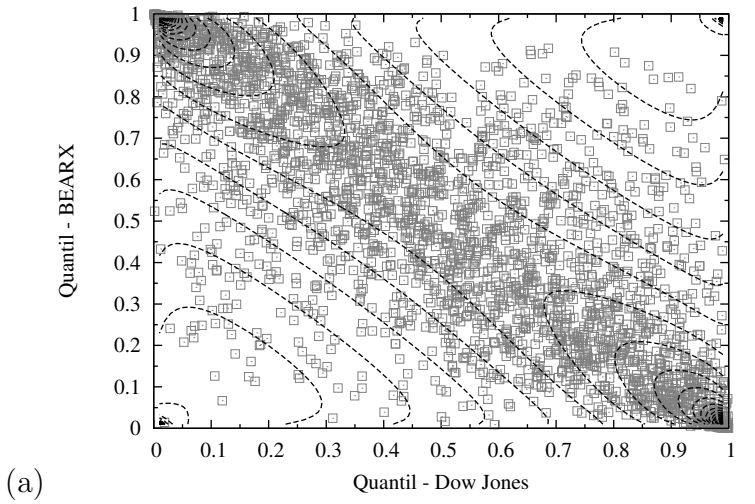


(a)

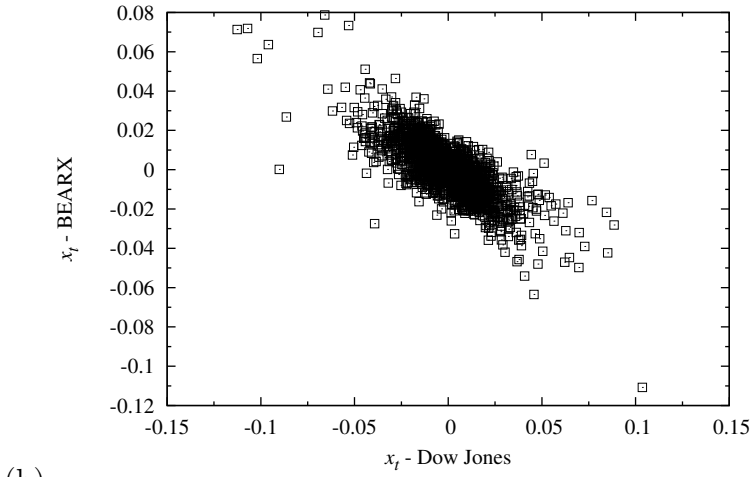


(b)

Abbildung 2.14.: (a) Verteilung der Quantilwerte des Aktienportfolios `Freddie Mac / Fannie Mae` sowie Wahrscheinlichkeitsverteilung der durch genetische Programmierung gefundenen Copula. (b) Verteilung der relativen Preisänderungen des Portfolios. Man beachte die extremen Ausprägungen.



(a)



(b)

Abbildung 2.15.: (a) Verteilung der Quantilwerte des Aktienportfolios `Dow Jones / BEARX` sowie Wahrscheinlichkeitsverteilung der durch genetische Programmierung gefundenen Copula. `BEARX` (Federated Prudent Bear Fund) enthält Short-Positionen und weist daher eine Anti-Korrelation mit dem Dow Jones auf. (b) Verteilung der relativen Preisänderungen des Portfolios.

3. Schlussfolgerungen

Im letzten Kapitel wurde anhand zweier Fallbeispiele gezeigt, wie genetische Programmierung in der Risikoanalyse zum Einsatz kommen kann. Die folgenden Abschnitte sollen das Potential und mögliche Probleme der hierbei verwendeten Verfahren kurz erläutern.

3.1. Anwendbarkeit von genetischer Programmierung im Risikomanagement

Wie sich in dieser Arbeit gezeigt hat, kann genetische Programmierung in mehreren Bereichen der Risikoanalyse erfolgreich eingesetzt werden. Zum einen können die mittels genetischer Programmierung erstellen Modelle des Value at Risk (VaR) sowie des Conditional Value at Risk (CVaR) prinzipiell den von der Praxis verlangten fachlichen Anforderungen genügen und zeigen durchweg eine mit den heute gebräuchlichen Standardmodellen vergleichbare, hohe Qualität.

Der Hauptvorteil eines mittels genetischer Programmierung modellierten VaR/CVaR ist dabei sicherlich die Fähigkeit des GP-Algorithmus, sich auf eventuell auftretende Strukturbrüche und veränderte Eigenschaften der untersuchten Zeitreihe einzustellen und hierauf "intelligent" zu reagieren. Im Gegensatz zu vielen parametrischen VaR-Modellen werden dabei fast keine modelltheoretischen Annahmen über die Struktur der zugrunde liegenden Zeitreihe vorausgesetzt. Weiterhin ist es mittels des genetischen Verfahrens möglich, neben den üblichen Zielkriterien für den VaR (Erwartungswert der Anzahl an Überschreitungen, stochastische Unabhängigkeit zwischen einzelnen Überschreitungen) weitere Zielkriterien zu definieren und in die Fitnessfunktion des Algorithmus zu integrieren. Hierdurch können die generierten VaR-Modelle im Hinblick auf fast beliebige Kriterien optimiert werden, was bei konventionellen Verfahren nicht oder nur sehr schwer möglich ist.

Schließlich ist es im Rahmen des GP-Algorithmus überaus einfach, zusätzliche Terminal- und Nichtterminalsymbole zur Modellgenerierung heranzuziehen. Zusätzliche Terminalsymbole könnten dabei, wie vorher bereits ausgeführt z.B. aus Expertenschätzungen gewonnene Maßzahlen bzw. sonstige quantitative Werte jeglicher Natur sein. Eine Einbeziehung solcher zusätzlicher Informationen ist im Rahmen der klassischen VaR-Modellierung nicht oder nur unter großen Schwierigkeiten möglich, im Rahmen der genetischen Programmierung stellt sie jedoch kein Problem dar.

Bei der Modellierung von Copulas hat sich ebenfalls gezeigt, dass eine genetische Programmierung der Standardmodellierung u.U. überlegen sein kann. Auch hier können ohne menschliches Zutun qualitativ hochwertige Modelle erzeugt werden, welche vorher festgelegten Kriterien entsprechen. Dabei können im Prinzip beliebig viele ein- oder mehrparametrige Copulamodelle vom genetischen Algorithmus zu einem Gesamtmodell zusammengeführt werden, was im Rahmen der klassischen Modellierung nur schwer möglich ist. Weiterhin sind die mittels des genetischen Verfahrens erstellen Copulamodelle unmittelbar interpretierbar und erlauben die einfache Berechnung wesentlicher Risikomaße. Durch die Einführung weiterer Fitnesskriterien können die modellierten Copulas weiterhin auf beliebige zusätzliche Eigenschaften hin optimiert werden.

In beiden betrachteten Anwendungsfällen ist davon auszugehen, dass ein für die Praxis ausgelegter und vollständig optimierter GP-Algorithmus unter sorgfältiger Wahl der Terminalsymbole und der Fitnessfunktion noch weitaus bessere Resultate hervorbringen kann. Da weiterhin die für diese Arbeit zur Verfügung stehenden Computerresourcen beschränkt waren, konnten die untersuchten GP-Algorithmen nicht mit den heute in der Praxis gebräuchlichen Populationsgrößen ($\gg 10^6$!) ausgeführt werden. Eine solche Erhöhung der Populationsgrößen würde jedoch mit fast absoluter Sicherheit eine weitere Qualitäts- und Effizienzsteigerung der Verfahren zur Folge haben, weswegen davon ausgegangen werden kann, dass die betrachteten Algorithmen noch ein hohes, unausgeschöpftes Potential besitzen.

3.2. Probleme & Verbesserungsmöglichkeiten

Problematisch an den hier betrachteten GP-Verfahren ist vor allem die Sensitivität der Ergebnisse im Hinblick auf die gewählte Fitnessfunktion. So schafft es

ein sorgfältig entworfener GP-Algorithmus zwar fast mit Sicherheit, eine gegebene Fitnessfunktion über den zur Verfügung stehenden Suchraum zu maximieren, dies garantiert jedoch nicht zwangsläufig die damit einhergehende optimale Lösung des untersuchten Problems. Wäre z.B. im Falle des VaR das verwendete Fitnesskriterium lediglich anhand der Überschreitungen des VaR formuliert, so würde der GP-Algorithmus u.U. Lösungen hervorbringen, die zwar eine korrekte Anzahl an Überschreitungen, jedoch einen viel zu hohen durchschnittlichen Wert des VaR aufweisen könnten. Daher ist die für ein Optimierungsproblem zu nutzende Fitnessfunktion stets mit äußerster Sorgfalt zu wählen.

Je nach der Komplexität des zu lösenden Problems kann es weiterhin zu einer schlechten Effizienz des genetischen Verfahrens kommen, falls die Evolutionsparamter oder der zur Verfügung stehende Satz an Terminal- und Nichtterminalsymbolen nicht optimal gewählt wurde. Generell ist die Komplexität der in dieser Arbeit betrachteten Probleme jedoch überschaubar und stellt für einen gut optimierten GP-Algorithmus kein Problem dar. Trotzdem kann sich die passende Wahl von Terminal- und Nichtterminalsymbolen bei anderen Problemen in der Praxis als schwierig herausstellen.

Verbesserungsmöglichkeiten für den GP-Algorithmus liegen vor allem in der Realisierung der genetischen Operationen (Mutation, Crossover, Reproduktion). So sollte es für den GP-Algorithmus generell möglich sein, von jedem Element des Suchraumes durch die Anwendungen der genetischen Operationen zu jedem anderen Element zu gelangen. Ist dies nicht möglich, so verhält sich der Algorithmus nicht optimal und sucht nur in einem beschränkten Unterraum des gesamten Lösungsraums nach guten Lösungen. Gerade bei syntaktisch eingeschränkten Programmen wie z.B. bei der Copulamodellierung im letzten Kapitel ist es dabei nicht trivial, korrekte genetische Operatoren nach den obigen Erfordernissen zu implementieren. Hierin liegt daher zusätzliches Verbesserungspotential für die Algorithmen.

A. Programme

Zur Durchführung der numerischen Untersuchungen und der GP-Algorithmen im Rahmen dieser Diplomarbeit wurde ein Programmpaket für die genetische Programmierung in der Programmiersprache C++ [Stroutrup, 2000] erstellt. Die folgenden Abschnitte erläutern kurz die einzelnen Elemente dieses Programmpakets und beispielhaft auch dessen Benutzung. Das gesamte Programmpaket umfaßt dabei mehr als 6.000 Zeilen Quelltext und ist in vollständiger Form auf der beiliegenden CD enthalten.

Alle Programmteile wurden in ISO-C++ geschrieben und können auf jeder konformen Plattform ausgeführt werden (dies schließt u.A. Linux, Unix sowie alle gängigen Windows-Varianten ein). Entsprechende Konfigurationsdateien/Makefiles liegen für Linux/Unix sowie Windows (Microsoft Visual Studio 2008) bei. Bei der Entwicklung des Programms wurde durchgängig auf einen objektorientierten Ansatz zurückgegriffen um die maximale Anpassungsfähigkeit der einzelnen Programmbestandteile zu garantieren. Die Kontrolle der einzelnen Programmteile geschieht über eine eingebettete Skriptsprache[Lua, 2009], die einen hohen Wiederverwendungswert der geschriebenen Routinen ermöglichen soll.

A.1. Klassendiagramm

Folgende, für die genetische Programmierung relevante Klassen wurden implementiert.

- **Node**: Basisklasse für `Branch`- und `Leaf`-Klassen. Repräsentiert einen Knoten im Programmbaum des genetischen Programms. Enthält u.a. Prototypfunktionen zur textuellen Ausgabe der Knoteneigenschaften und zur Evaluierung des Funktionswertes des Knotens. Enthält einen Vektor zur Modellierung von Unterknoten.

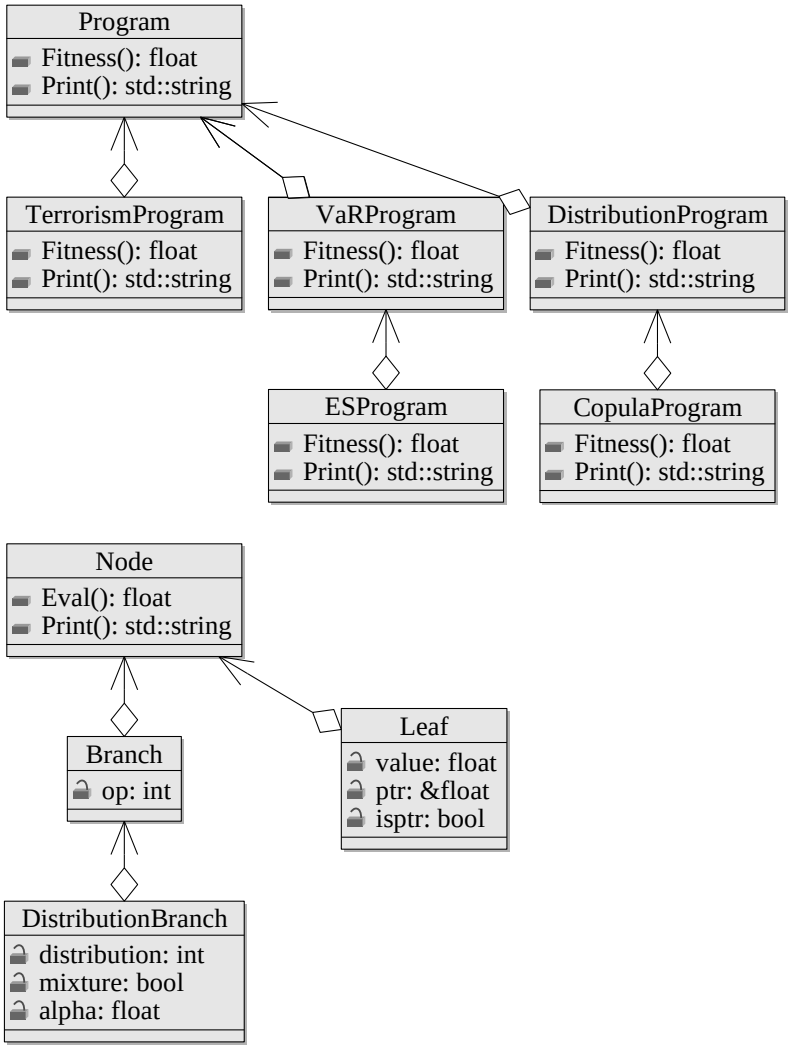


Abbildung A.1.: Vereinfachtes UML-Klassendiagramm der für die genetische Programmierung erstellten Programme.

- **Leaf**: Unterklasse von `Node`. Modelliert ein Terminal-Symbol bzw. "Blatt" des genetischen Programms. Überschreibt die Evaluations- und Ausgabefunktion von `Node` und enthält als Funktionswert entweder eine numerische Konstante oder einen externen `float`-Wert, der als Zeiger angegeben werden kann.

- **Branch**: Unterklasse von `Node`. Modelliert ein Nichtterminal-Symbol bzw. einen "Ast" des genetischen Programms. Überschreibt die Evaluations- und Ausgabefunktion von `Node` und enthält mindestens einen Unterknoten. Auf den Unterknoten wird eine bestimmte Operation ausgeführt, deren Ergebnis

wird bei der Evaluierung zurückgegeben.

- **Program**: Basisklasse für ein genetisches Programm. Enthält Prototypfunktionen für die Evaluation der Programmfitness (`float Fitness()`), für die Ausgabe des Programms in eine Datei (`void Write(std::string directory,int generation`) sowie für die Durchführung von Crossover-, Mutations- und Shrink-Operationen. Spezifische genetische Programme erben von dieser Klasse und implementieren eigene `Fitness()` sowie `Write()`-Funktionen.

- **Population**: Verwaltet eine Menge von genetischen Programmen und führt die globale Fitnessberechnung und die Selektion einzelner Programme für die Crossover-, Mutations- und Shrink-Operationen durch. Gibt weiterhin Statistiken über relevante Populationsparameter aus und steuert den Ablauf des GP-Algorithmus.

A.2. Genetische Programme

Für die im Rahmen der Diplomarbeit durchgeführten Untersuchungen wurden verschiedene Unterklassen von `Program` erstellt. Diese Unterklassen werden im folgenden kurz erläutert.

- **VaRProgram**: Programm zur Berechnung des Value at Risk für einen Aktienwert oder einen Testdatensatz. Werden bei der Erzeugung keine Parameter angegeben, so wird ein Testdatensatz generiert. Es kann bei der Erzeugung ein Dateiname, ein Spaltenindex für die Datenspalte sowie ein Trennzeichen als Parameter angegeben werden. Beispiel: `pop.create(p,500,10, program.var, ''data/stocks/amex.csv'',6,'','');` erzeugt eine Population von 500 genetischen VaR-Programmen mit einer durchschnittlichen Länge von 10 und dem zugrunde liegenden Aktienwert von `American Express`, der sich in einer kommaseparierten Datei befindet deren sechste Spalte den Aktienpreis enthält.

- **ESProgram**: Programm zur Berechnung des Conditional Value at Risk (CVaR) bzw. Expected Shortfalls (ES). Eingabeparameter ist eine tabulatorseparierte Ausgabedatei einer

VaRProgram-Klasse. Beispiel: `pop.create(p,10000,10,program.es, ''data/var_dax_good/best_17.tsv'');` lädt das Ergebnis des besten Programms der 17. Generation einer VaRProgram-Population für den Aktienindex `Dax` als Eingabevektor.

- **CopulaProgram**: Programm zur Modellierung von Copulas. Enthält als Eingabeparameter entweder zwei Aktienwerte in der gleichen Syntax wie bei `VaRProgram`, einen Dateinamen der bereits eine vorher gespeicherte Ausgabedatei des Programms enthält oder überhaupt keinen Parameter, wodurch ein Testdatensatz erzeugt wird. Beispiel: `pop.create(p,10000,10, program.copula, ''data/stocks/amex.csv'', ''data/stocks/citigroup.csv'',6,'','');`

- **DistributionProgram**: Ähnlich zu `CopulaProgram`, jedoch zur Modellierung univariater Verteilungen.

- **SymbolicProgram**: Führt eine symbolische Regression auf einem Testdatensatz durch und dient lediglich zum Testen der genetischen Operatoren der Basisklassen `Program` und `Population`.

Sämtliche Programme erzeugen in einem vorgegebenem Ausgabeverzeichnis eine Reihe von Dateien. U.a. werden verschiedene Logbücher angelegt (`event.log`, `script.log`, `error.log`), in denen vom Programm wichtige Informationen wie z.B. aktuelle Populationsparameter, Regressionswerte etc. abgelegt werden können. Weiterhin werden in jedem Schritt des GP-Algorithmus die Dateien `programs_xxx.tsv`, `histo_xxx.tsv` und `best_xxx.tsv` erzeugt (`xxx` wird durch Nummer der jeweiligen Generation ersetzt). Erstere enthält die textuelle Darstellung aller Programme in der entsprechenden Population, die zweite enthält die aufbereitete Verteilung der Fitnesswerte innerhalb der Population und letztere enthält die programmspezifische Ausgabe des besten Individuums der entsprechenden Population. Weiterhin wird die Datei `stat.tsv` erzeugt, welche statistische Informationen über die Entwicklung der Programmlänge und der Fitness in Abhängigkeit der Generation enthält. Schließlich enthält die Datei `script.lua` eine Kopie des Skriptes, welches dem Programm zur Ausführung vorgegeben wurde.

A.3. Beispielskript

Nachfolgend ein Beispielskript, welches eine Population von genetischen Programmen zur Berechnung eines Copulamodells zweier Aktienwerte erstellt und den GP-Algorithmus über 100 Generationen ausführt. Sämtliche Ausgabedateien werden dabei wie oben beschrieben in einem Zielordner gespeichert. Das aufgeführte Skript kann als Muster für die Ausführung eigener Skripte herangezogen werden.

```
p=pop.new();

dir="data/copula_dow_bearx";

pop.output_dir(p,dir);

logger.reopen(dir);

print("Mutation rate:",pop.mutation_rate(p,0.05),"\n");
print("Shrink rate:",pop.shrink_rate(p,0.025),"\n");
print("Crossover size:",pop.crossover_size(p,0.2),"\n");
print("Tournament size:",pop.tournament_size(p,6),"\n");

pop.create(p,1000,10,program.copula,                              ->
"data/stocks/dow.csv", "data/stocks/bearx.csv",6,",");
pop.run(p,100);
exit(0);
```

Literaturverzeichnis

[Applegate et al., 2007] Applegate, D., Bixby, R., Chvatal, V., and Cook, W. (2007). *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 1st edition.

[Artzner et al., 2001] Artzner, P., Delbaen, F., Eber, J., and Heath, D. (2001). Coherent measures of risk. *Mathematical Finance*, 9:203–228.

[Banzhaf et al., 2002] Banzhaf, W., Nordin, P., and Keller, R. (2002). *Genetic Programming, an Introduction*. Dpunkt. Verlag GmbH.

[Basler Ausschuss für Bankenaufsicht, 2004] Basler Ausschuss für Bankenaufsicht (2004). Internationale Konvergenz der Eigenkapitalmessung und der Eigenkapitalanforderungen. Technical report, Bank für Internationalen Zahlungsausgleich.

[Bentley, 1999] Bentley, P. (1999). *Evolutionary Design by Computers*. Morgan Kaufmann.

[Bentley and Corne, 2001] Bentley, P. and Corne, D. (2001). *Creative Evolutionary Systems*. Academic Press.

[Chen, 2008] Chen, S. (2008). *Genetic Algorithms and Genetic Programming in Computational Finance*. Kluwer Academic Publishers.

[Darwin, 1859] Darwin, C. (1859). *The Origin of Species*. John Murray.

[Eiben and Smith, 1998] Eiben, A. E. and Smith, J. E. (1998). *Introduction to Evolutionary Computing*. Springer, New York.

[Elton et al., 2006] Elton, E., Gruber, M., Brown, S., and Goetzmann, W. (2006). *Modern Portfolio Theory and Investment Analysis*. Wiley, 7th edition.

[Feigenbaum and Feldman, 1995] Feigenbaum, E. and Feldman, J., editors (1995). *Computers and Thought*. MIT Press.

[Grefenstette et al., 1985] Grefenstette, J. J., Gopal, R., Rosmaita, B. J., and Gucht, D. V. (1985). Genetic algorithms for the traveling salesman problem. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 160–168, Hillsdale, NJ, USA. L. Erlbaum Associates Inc.

[Hamilton, 1994] Hamilton, J. (1994). *Time Series Analysis*. Princeton University Press, New Jersey.

[Heigl, 2008] Heigl, A. (2008). Option pricing by means of genetic programming. Master's thesis, Technische Universität Wien.

[Holland, 1962] Holland, J. H. (1962). Outline for a logical theory of adaptive systems. *J. ACM*, 9(3):297–314.

[Holland, 1992a] Holland, J. H. (1992a). *Adaptation in Natural and Artificial Systems*. MIT Press.

[Holland, 1992b] Holland, J. H. (1992b). Genetic algorithms. *Scientific American*.

[Joe, 1997] Joe, H. (1997). *Multivariate Models and Dependence Concepts*. Chapman & Hall /CRC, Boca Raton.

[J.P. Morgan, 1996] J.P. Morgan (1996). Riskmetrics - technical document. http://www.riskmetrics.com/system/files/private/td4e.pdf, Abruf:13.01.09.

[Kleinau, 2003] Kleinau, P. (2003). *Application of Genetic Programming to Finance and Operations Management*. PhD thesis, Universität Münster.

[Koza, 1992] Koza, J. R. (1992). *Genetic Programming I: On the Programming of Computers by Means of Natural Selection*. Bradford Books.

[Koza, 1994] Koza, J. R. (1994). *Genetic Programming II: Automatic Discovery of Reusable Programs*. Bradford Books.

[Koza, 1999] Koza, J. R. (1999). *Genetic Programming III: Darwinian Invention and Problem Solving*. Morgan Kaufmann Publishers.

[Koza, 2003] Koza, J. R. (2003). *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*. Springer, New York.

[Koza et al., 1996] Koza, J. R., Andre, D., Iii, F. H. B., and Keane, M. A. (1996). Use of automatically defined functions and architecture-altering operations in automated circuit synthesis using genetic programming. In *Stanford University*, pages 28–31. MIT Press.

[Langdon and Poli, 1998] Langdon, W. B. and Poli, R. (1998). *Foundations of Genetic Programming*. Springer, New York.

[Lua, 2009] Lua (2009). The Programming Language Lua. http://www.lua.org/, Abruf: 28.01.09.

[Mandelbrot and Franklin, 2008] Mandelbrot, B. and Franklin, A. (2008). *The (Mis) Behaviour of Markets: A Fractal View of Risk, Ruin and Reward*. Profile Books.

[Markowitz, 1952] Markowitz, H. (1952). Portfolio selection. *The Journal of Finance*, 7(1):77–91.

[McNeil et al., 2005] McNeil, A. J., Frey, R., and Embrechts, P. (2005). *Quantitative Risk Management*. Princeton University Press, Princeton and Oxford.

[Newell and Simon, 1995] Newell, A. and Simon, H. (1995). *GPS, a program that simulates human thought*, pages 156–165. In [Feigenbaum and Feldman, 1995].

[Philippe, 2002] Philippe, J. (2002). *Value at Risk*. McGraw Hill, 2nd edition.

[Press et al., 2002] Press, W. H., S.A.Teukolsky, Vetterling, W. T., and Flannery, B. P. (2002). *Numerical Recipes in C++*. Cambrige University Press, 2nd edition.

[Reich, 2004] Reich, C. (2004). *Applications of Extreme Value Theory in Financial Risk Modelling*. PhD thesis, Universität St. Gallen.

[Reiss and Thomas, 2007] Reiss, R. and Thomas, M. (2007). *Statistical Analysis of Extreme Values*. Birkhäuser, Basel, 3rd edition.

[Stroutrup, 2000] Stroutrup, B. (2000). *The C++ Programming Language*. Addison Wesley, Reading, MA, USA.

[Subramanian et al., 2006] Subramanian, H., Ramamoorthy, S., Stone, P., and Kuipers, B. J. (2006). Designing safe, profitable automated stock trading agents using evolutionary algorithms. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1777–1784, New York, NY, USA. ACM.

[Trivedi and Zimmer, 2005] Trivedi, P. and Zimmer, D. (2005). *Copula Modeling: An Introduction for Practitioners*. World Scientific Publishing.

[Turing, 1936] Turing, A. (1936). On computable numbers, with an application to the Entscheidungsproblem. In *Proceedings of the London Mathematical Society*.

[Wolke, 2008] Wolke, T. (2008). *Risikomanagement*. Oldenbourg, München.

[Wolpert and Macready, 1997] Wolpert, D. and Macready, W. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1:67–82.

[Yahoo! Deutschland GmbH, 2008] Yahoo! Deutschland GmbH (2008). Yahoo! Finanzen. t://de.iance.yaoo.com/, Abruf: 16.10.08.

Index

Danksagung

Ich danke zunächst Herrn Prof. **Michael Merz** für die Betreuung der Diplomarbeit und für die interessanten Diskussionen mit ihm. Weiterhin danke ich allen Mitarbeitern am Lehrstuhl für Statistik und empirische Wirtschaftsforschung für die gute Betreuung während meines Studiums.

Der **Stiftung der Deutschen Wirtschaft (SDW)** danke ich für die großzügige Förderung meines Physik- und BWL-Studiums und für die zahlreichen interessanten Seminare und Veranstaltungen.

Meinen Eltern **Bernadette** und **Richard** danke ich für die Unterstützung meines Studiums sowie für ihren Beistand und das Verständnis, das sie mir in allen Lebenslagen entgegen brachten.

Erklärung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbständig ohne unerlaubte fremde Hilfe und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen oder aus anderweitigen fremden Äußerungen entnommen worden sind, habe ich als solche einzeln kenntlich gemacht.

Tübingen, 29. Januar 2009

(Andreas Dewes)