2. Von der Evolution zur Optimierung

Die Prinzipien der biologischen Evolution werden auf die Optimierung übertragen. Am Beispiel wird ein erster evolutionärer Algorithmus zur Optimierung konstruiert und ein formaler Rahmen für die verschiedenen Algorithmenvarianten entwickelt.

Lernziele in diesem Kapitel

- Optimierungsprobleme können formal definiert werden.
- Das allgemeine Ablaufschema der einfachen evolutionären Algorithmen wird verstanden und als generisches Muster aufgefasst.
- Die Unterscheidung zwischen Genotyp und Phänotyp wird verinnerlicht und kann effektiv im konkreten Beispiel umgesetzt werden.
- Die Anpassung eines evolutionären Algorithmus an ein Optimierungsproblem kann zumindest am Beispiel nachvollzogen werden.
- Evolutionäre Algorithmen werden als eine Optimierungstechnik unter vielen verstanden und auch entsprechend differenziert eingesetzt.

Gliederung

Biologen studieren die Evolution als Mechanismus, der in der Natur spezielle Lösungen für spezielle Probleme erzeugt. Sie produziert etwa Antworten auf Fragen hinsichtlich der Aufnahme von Energie aus der Umwelt, der Produktion von genügend Nachkommen, um die Art zu erhalten, der Partnerfindung bei sexueller Fortpflanzung, des optimalen Energieaufwands zur Erzeugung von vielen oder wenigen Nachkommen, der optimale Tarnung etc. Diese Lösungen sind unter anderem das Resultat von Mutation, Rekombination und natürlicher Selektion – so gesehen ist der Evolutionsprozess die »natürliche Allzweckwaffe« für die Weiterentwicklung der Natur.

Computer hingegen dienen seit ihrer Erfindung als universeller Problemlöser für verschiedenste Aufgaben. Als ein Modell für Rechenmaschinen hat Turing in den 1930er Jahren die Turing-

K. Weicker, *Evolutionäre Algorithmen,* DOI 10.1007/978-3-658-09958-9_2,

Maschine eingeführt und die Behauptung aufgestellt, dass sich jedes algorithmisch lösbare Problem auf diesem Maschinenmodell lösen lässt. Gleichzeitig hat er bewiesen, dass Probleme existieren, die in allgemeiner Form algorithmisch nicht gelöst werden können. Ein Beispiel ist das so genannte Halteproblem, bei dem für ein beliebiges Programm zu entscheiden ist, ob es für eine gegebene Eingabe anhält.

Für algorithmisch lösbare Probleme gibt es kein allgemeines Rezept, wie der Algorithmus für ein spezielles Problem auszusehen hat. Dies bleibt der Kreativität des Informatikers oder Programmierers überlassen. Darüber hinaus kann für sehr viele Probleme nicht gewährleistet werden, dass es einen Algorithmus mit effizienter Laufzeit gibt. (In diese Kategorie fallen auch die sog. NP-harten Probleme.)

Evolutionäre Algorithmen kombinieren den Computer als universelle Rechenmaschine mit dem allgemeinen Problemlösungspotential der natürlichen Evolution. So wird im Computer ein Evolutionsprozess künstlich simuliert, um für ein nahezu beliebig wählbares Optimierungsproblem möglichst gute Näherungswerte an eine exakte Lösung zu erzeugen. Dabei wird ein beliebiges abstraktes Objekt, das eine mögliche Lösung für ein Problem darstellt, wie ein Organismus behandelt. Dieses wird durch Anwendung von so genannten evolutionären Operatoren variiert, reproduziert und bewertet. Diese Operatoren nutzen in der Regel Zufallszahlen für ihre Veränderungen an den Individuen. Folglich zählen evolutionäre Algorithmen zu den stochastischen Optimierungsverfahren, die häufig keine Garantie auf das Auffinden der exakten Lösung (in einem vorgegebenen Zeitrahmen) geben können.

Insbesondere bei Problemen, die nicht in akzeptabler Zeit exakt lösbar sind, gewinnen Algorithmen, die auf solchen biologischen Vorbildern beruhen, immer mehr an Bedeutung.

2.1. Optimierungsprobleme

Optimierungsprobleme werden allgemein definiert und am Beispiel des Handlungsreisendenproblems erläutert.

Optimierungsprobleme treten in allen Bereichen von Industrie, Forschung und Wirtschaft auf. Den Anwendungsgebieten sind dabei keine Grenzen gesetzt. Beispiele reichen von der reinen Kalibrierung von Systemen, über Strategien zur besseren Ausnutzung vorhandener Ressourcen bis hin zu Prognosen von Zeitreihen oder der Verbesserung von Konstruktionen. Jedes dieser Probleme bringt andere Voraussetzungen für die Bewertung von Lösungskandidaten sowie unterschiedliche Anforderung an deren Optimalität mit. Daher werden wir Optimierungsprobleme zunächst in einer allgemeinen Grundform definieren. (Kapitel 5 behandelt verschiedene spezielle Anforderungen in Optimierungsproblemen.)

Für eine formale Definition werden die folgenden Forderungen an ein Problem gestellt: Die Menge aller möglichen Lösungskandidaten hat klar definiert zu sein und für jeden Lösungskandidaten muss auf irgendeine Art und Weise seine Güte oder Qualität als mögliche Lösung eindeutig berechenbar sein. Damit werden die verschiedenen Lösungskandidaten vergleichbar.

Definition 2.1 (Optimierungsproblem):

Ein *Optimierungsproblem* $(\Omega, f, \succ)$ ist gegeben durch einen Suchraum Ω, eine *Bewertungsfunktion* $f: \Omega \to \mathbb{R}$, die jedem Lösungskandidaten einen Gütewert zuweist,

sowie eine Vergleichsrelation $\succ \in \{<, >\}$. Dann ist die *Menge der globalen Optima* $\mathscr{X} \subseteq \Omega$ definiert als

$$\mathscr{X} = \left\{ x \in \Omega \;\middle|\; \forall\, x' \in \Omega :\; f(x) \succeq f(x') \right\}.$$

Ein Beispiel dafür ist das Handlungsreisendenproblem (TSP, engl. *travelling salesman problem*), bei dem eine kostenminimale Rundreise durch eine gegebene Menge von Städten gesucht wird, wobei jede Stadt nur einmal besucht werden darf.

Definition 2.2 (Handlungsreisendenproblem):

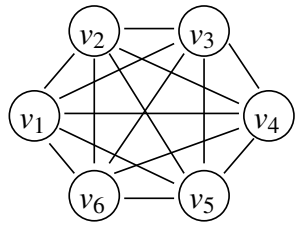
Die Grundlage für die Definition des Handlungsreisendenproblems ist ein Graph $G = (V, E, \gamma)$ zur Berechnung der Kosten. Die Knotenmenge $V = \{v_1, \ldots, v_n\}$ repräsentiert n verschiedene Städte, die paarweise durch Straßen in der Kantenmenge $E \subseteq V \times V$ verbunden sind. Jeder dieser Straßen ist eine Fahrtzeit $\gamma : E \to \mathbb{R}$ zugeordnet. Das *Handlungsreisendenproblem* ist dann definiert als Tupel $(\mathscr{S}_n, f_{\text{TSP}}, <)$, wobei der Raum aller Permutationen $\mathscr{S}_n$ die unterschiedlichen Besuchsreihenfolgen repräsentiert. Die zu minimierende Bewertungsfunktion f_{TSP} ist definiert für $(\pi_1, \ldots, \pi_n) \in \mathscr{S}_n$ als

$$f_{\text{TSP}}\left((\pi_1, \ldots, \pi_n)\right) = \gamma\left((v_{\pi_n}, v_{\pi_1})\right) + \sum_{j=2}^{n} \gamma((v_{\pi_{j-1}}, v_{\pi_j})).$$

Ein Handlungsreisendenproblem heißt ferner genau dann *symmetrisch*, wenn für alle $(v_i, v_j) \in E$ sowohl $(v_j, v_i) \in E$ als auch $\gamma((v_i, v_j)) = \gamma((v_j, v_i))$ erfüllt sind.

Beispiel 2.1:

Bild 2.1 zeigt ein kleines Handlungsreisendenproblem mit sechs Städten. Der dazugehörige Suchraum mit allen möglichen Rundreisen ist in Bild 2.2 dargestellt. Jede der dargestellten Rundreisen steht dabei für zwölf verschiedene Rundreisen, die an jeder der 6 Städten mit zwei unterschiedlichen Fahrtrichtungen beginnen kann. Wenn man die Rundtouren weglässt, die sich nur durch die Fahrtrichtung oder die Startstadt unterscheiden, gibt es im vorliegenden Beispiel genau 60 verschiedene Lösungen. Bei 101 Städten sind es bereits $4{,}663\,1 \cdot 10^{157}$, allgemein $\frac{1}{2} \cdot (n-1)!$ für n Städte.



Kante e	$\gamma(e)$	Kante e	$\gamma(e)$	Kante e	$\gamma(e)$
(v_1, v_2)	5	(v_2, v_3)	10	(v_3, v_5)	17
(v_1, v_3)	8	(v_2, v_4)	4	(v_3, v_6)	8
(v_1, v_4)	11	(v_2, v_5)	9	(v_4, v_5)	6
(v_1, v_5)	3	(v_2, v_6)	12	(v_4, v_6)	5
(v_1, v_6)	7	(v_3, v_4)	6	(v_5, v_6)	11

Bild 2.1. Ungerichteter Graph eines beispielhaften symmetrischen Handlungsreisendenproblems, bei dem es zwischen allen Paaren von Städten eine Straße gibt. Die Tabelle gibt die Kosten bzw. Fahrtzeiten der einzelnen Straßen wieder.

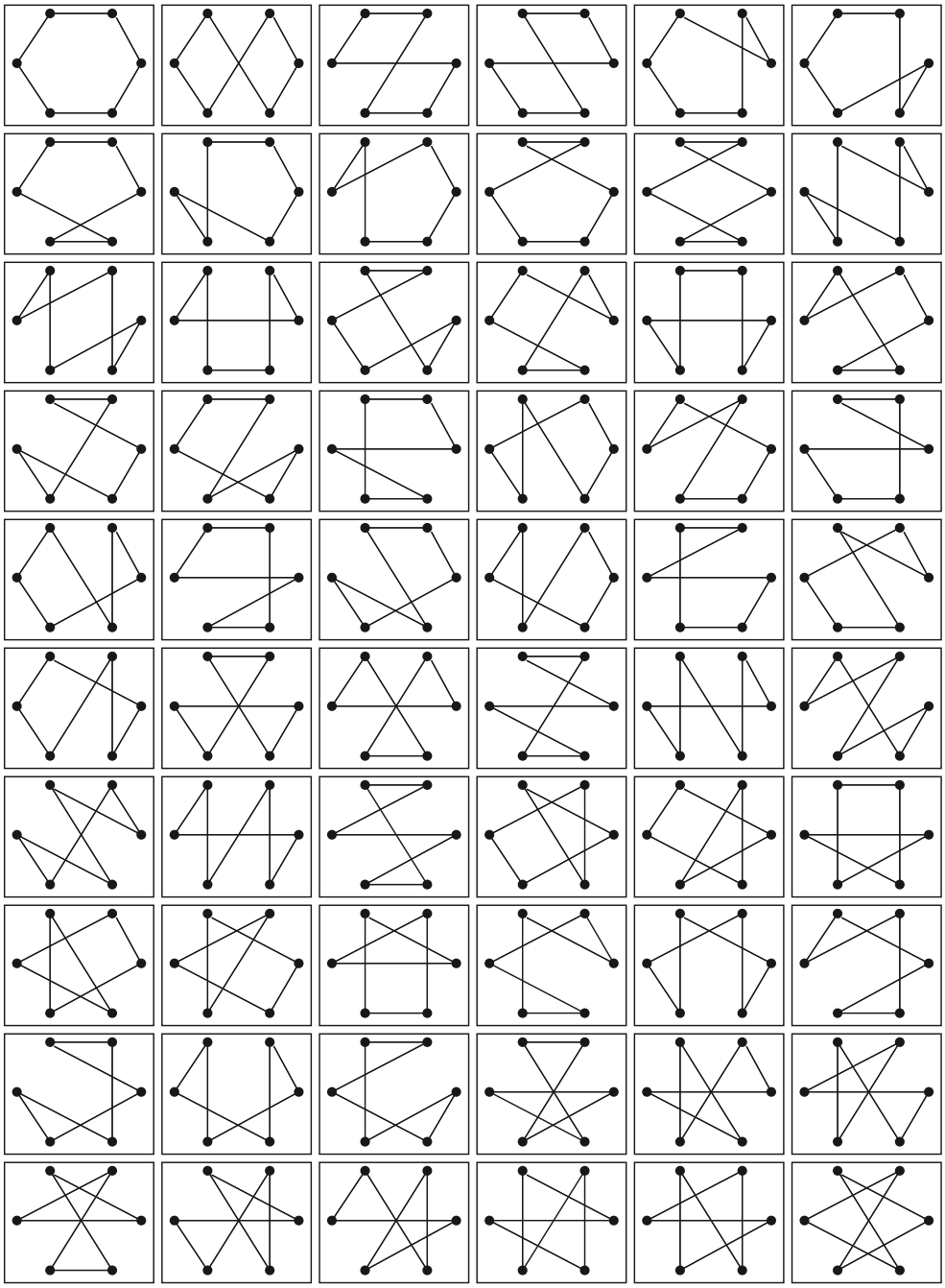
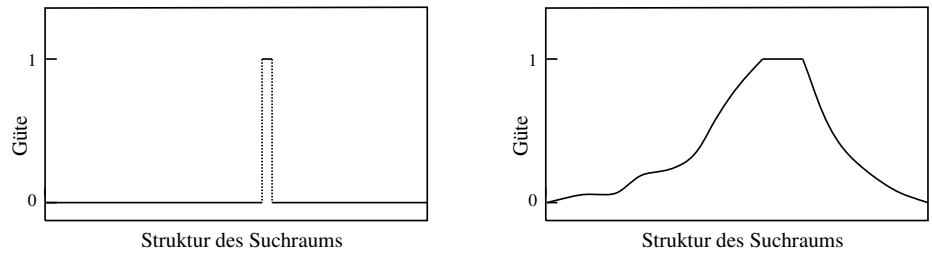


Bild 2.2. Problemraum des Handlungsreisendenproblems: alle möglichen Rundreisen



(a) Nadel im Heuhaufen (b) leitende Bewertungsfunktion

Bild 2.3. Vergleich der (a) Bewertungsfunktion eines Entscheidungsproblems und einer (b) leitenden graduellen Bewertung

Das Handlungsreisendenproblem zeichnet sich durch eine strikt vorgegebene und damit offensichtliche Struktur der Lösungskandidaten aus: eine Permutation der Indizes der Städte. Dies ist beispielsweise nicht der Fall, wenn eine Brückenkonstruktion gewichtsminimal so optimiert werden soll, dass sie dennoch eine vorgegebene maximale Last tragen kann. Hier sind verschiedene Ansätze denkbar, wie ein Lösungskandidat die Struktur eines solchen Tragwerks beschreibt. Auch solche Probleme lassen sich mit Definition 2.1 erfassen, indem der Suchraum Ω entsprechend definiert wird.

Vorsicht wiederum ist bei vielen »Optimierungsproblemen« aus der Wirtschaft geboten: Ohne die Möglichkeit oder die Bereitschaft, das Problem mathematisch zu modellieren, sind einmalige Managemententscheidungen oder Verbesserungen im Workflow nicht optimierbar. Eine klare Definition des Bewertungskriteriums ist die Voraussetzung für alle in diesem Buch vorgestellten Verfahren.

Das Optimierungsproblem muss nicht nur präzise definiert werden – eine gute Bewertungsfunktion zeichnet sich zusätzlich durch die folgenden Eigenschaften aus.

- Eine graduelle Bewertung ist besser als eine absolute. So könnte etwa in einem Handlungsreisendenproblem eine Rundtour gesucht werden, die eine vorgegebene Kostenschranke unterschreitet. Dies ließe sich leicht als Erfüllbarkeitsproblem formulieren, indem je nach Länge der Rundtour auf die Werte »1« (Erwartungen werden erfüllt) und »0« (Tour ist zu lang) abgebildet wird. Aus Anwendersicht spiegelt eine solche Definition zwar die Anforderungen genau wider – eine Optimierung wird jedoch zur Suche nach der Nadel im Heuhaufen, da wir keinen Anhaltspunkt dafür haben, welche von zwei zu langen Touren eventuell näher an der Kostenschranke liegt (Bild 2.3 (a)). Folglich sollten Erfüllbarkeitsprobleme wenn möglich als Optimierungsprobleme formuliert werden, welche eine Optimierung auch lenken und leiten kann (Bild 2.3 (b)).
- Die Anforderungen an eine Lösung des Problems spiegeln sich möglichst genau in der Bewertungsfunktion wider. Ist dies nicht der Fall, kann es einerseits passieren, dass bestimmte Aspekte gar nicht berücksichtigt werden und damit jede vom Optimierungsverfahren präsentierte Lösung beliebig weit von den Erwartungen entfernt ist. Andererseits können Lösungskandidaten aus einem breiten Qualitätsspektrum (aus Sicht des Anwenders) auf ähnliche Gütewerte abgebildet werden, sodass nur gelegentlich eine sinnvolle Lösung gefunden wird.

Die Bewertungsfunktion ist der wichtigste Bestandteil eines Problems, aus dem ein Optimierungsalgorithmus die Richtung der Optimierung ableitet. Daher muss in vielen Anwendungen diesem Aspekt hinreichend viel Aufmerksamkeit gewidmet werden. Und in einigen Fällen ist mit der Suche nach geeigneten Kriterien für die Erfassung der Güte eines Lösungskandidaten der größte Teil der Arbeit erledigt.

2.2. Der simulierte evolutionäre Zyklus

Die einzelnen Faktoren der natürlichen Evolution werden auf die Optimierung übertragen und in einen klar definierten Ablauf eingeordnet.

Aus dem Blickwinkel einer natürlichen Population ist der Erhalt der eigenen Art das höchste Ziel der Evolution. Dabei stellt die Umwelt die Organismen vor vielfältige Herausforderungen, für die es keine Ideallösung gibt. Die natürliche Evolution hat kein übergeordnetes, klar überprüfbares Ziel. Daher wird der Nutzen eines Allels auch nicht direkt gemessen, sondern indirekt im Vergleich mit anderen Allelen durch die Anzahl der Nachkommen als Fitness angenähert.

Im Gegensatz dazu ist bei klassischen Optimierungsproblemen meist ein klares Bewertungskriterium für die Qualität eines Lösungskandidaten vorhanden, das insbesondere keinen Zufallseinflüssen unterworfen ist. Die Qualität eines Lösungskandidaten kann durch eine so genannte Ziel- oder Bewertungsfunktion berechnet werden und wird im Weiteren als »Wert« oder »*Güte*« bezeichnet. Wir ersetzen also unsere schwer fassbare Umwelt durch eine klar definierte Bewertungsfunktion. Synonym werden in der Literatur auch die Begriffe Objektfunktion und Fitnessfunktion benutzt. Ebenso wird die Güte eines Lösungskandidaten auch als Kosten oder Fitness bezeichnet – letzteres hat in diesem Buch allerdings eine andere Bedeutung.

Um die natürliche Evolution auf die Lösung von Optimierungsproblemen zu übertragen, orientieren wir uns an den Evolutionsfaktoren Mutation, Rekombination und Selektion. Das Wechselspiel von Variation und Selektion gemäß Darwin wird als sequentielles Abarbeiten einzelner Phasen auf einer Population interpretiert. Fügt man noch einen definierten Startpunkt zur Initialisierung der Population und einen Endpunkt in Form einer Terminierungsbedingung hinzu, resultiert der in Bild 2.4 dargestellte evolutionäre Zyklus.

In Anlehnung an die biologische Terminologie spricht man bei einem Lösungskandidaten vom *Individuum* und bei den aktuell betrachteten Individuen von der *Population*. Im Weiteren werden Individuen meist mit A, B, ... bezeichnet und Populationen mit $P = \langle A^{(i)} \rangle_{1 \leq i \leq s}$. Obwohl die Individuen einer Population grundsätzlich nicht sortiert sind, werden sie als Tupel repräsentiert, wodurch sich die Algorithmen einfacher formulieren lassen. Zudem können einzelne Individuen in der Population mehrfach vorkommen, sodass bei einer Darstellung als Menge die Notation von Multimengen mit der Angabe der Häufigkeit für jedes Individuum notwendig wäre.

Die Initialisierung legt eine Population mit ersten Lösungskandidaten an. Meist werden diese zufällig gewählt, allerdings können auch Startkandidaten wie beste bekannte Lösungskandidaten oder Ergebnisse anderer Optimierungsverfahren genutzt werden.

Die anschließende erste Bewertung ermittelt für jeden Lösungskandidaten der Population die Güte, indem die Bewertungsfunktion berechnet wird.

Die Paarungs- oder Elternselektion nutzt die Güte der Individuen, um für jedes Individuum festzulegen, wie viele Kindindividuen daraus erzeugt werden sollen. Die Generierung der neuen

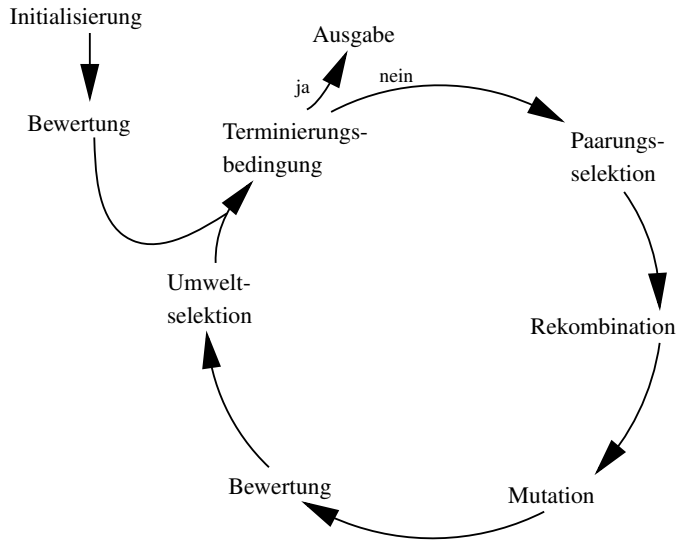


Bild 2.4. Schematische Darstellung des Zyklus bei evolutionären Algorithmen

Individuen geschieht allgemein durch eine Rekombination der Merkmale mehrerer Elternindividuen und eine anschließende Mutation der Kinder. Analog zur Biologie dient die Rekombination der Durchmischung in der Population und die Mutation nimmt in der Regel nur eine sehr kleine Veränderung am Individuum vor, um die Vererbung der elterlichen Eigenschaft auf das Kind nicht zu stark zu stören. Einzelne evolutionäre Algorithmen verzichten auf die Rekombination.

Nach einer Bewertung der neuen Individuen werden die Kinder durch die Umweltselektion in die Population der Eltern integriert. Da die Populationsgröße meist begrenzt ist, werden entweder einzelne Individuen aus der Elternpopulation oder die gesamte Elternpopulation durch die Kinder ersetzt.

Anschließend beginnt der Zyklus wieder mit der Paarungsselektion. Die Population jeder Iteration wird als neue *Generation* bezeichnet. Am Ende jeder Iteration prüft die Terminierungsbedingung, ob das Ziel bereits erreicht wurde. Dies kann mittels eines vorgegebenen Schwellwerts für die erwünschte Güte, eine Anzahl der Generationen ohne Verbesserung und/oder eine maximale Anzahl an Iterationen geschehen.

Um einen solchen Algorithmus anwenden zu können, wird lediglich eine im Rechner speicherbare Darstellung des Suchraums und eine Bewertungsfunktion benötigt. Beide Aspekte wurden im Rahmen der Definition der Optimierungsfunktion gefordert. Die Tatsache, dass keine weiteren Voraussetzungen für die Anwendbarkeit des Algorithmus erfüllt sein müssen, ist eine der attraktivsten Eigenschaften von evolutionären Algorithmen.

Bisher wurden die evolutionären Algorithmen streng aus der Biologie heraus entwickelt – wie dies auch historisch geschehen ist. Interessanterweise gelangen wir jedoch auch intuitiv in einem Black-Box-Szenario zu nahezu identischen Konzepten. So lasse ich meine Studierenden in der Vorlesung an der Tafel ein zweidimensionales Problem durch Platzieren von Stichproben lösen, woraus dieselben Grundoperationen abgeleitet werden. Ein Beispiel ist in Bild 2.5 gezeigt. Was dennoch vom natürlichen Vorbild bleibt, ist der einzigartige Reichtum der Natur als Inspirationsquelle für neue Techniken.

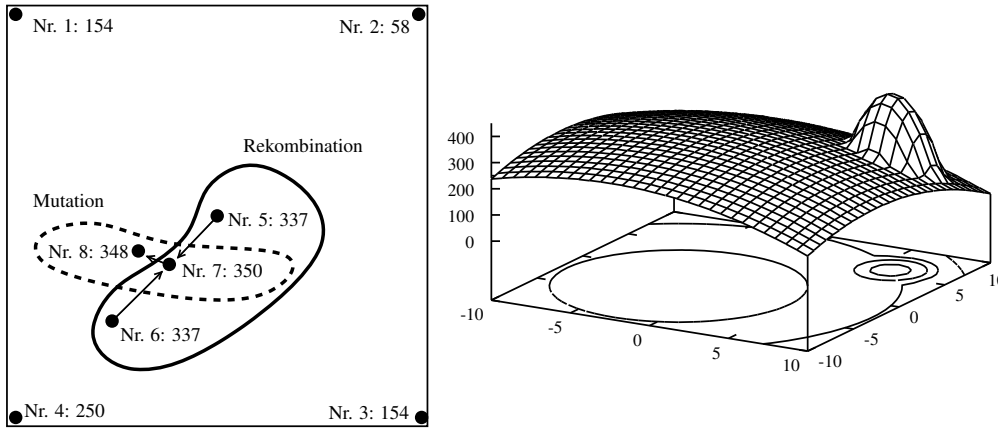


Bild 2.5. Ein Beispiel dafür, wie in einem Black-Box-Szenario das Maximum der rechten Funktion gesucht wird. Durch Stichproben muss der Problemraum erkundet werden. Ohne Strukturinformation werden beispielsweise zunächst die Eckpunkte und die Mitte betrachtet (Initialisierung). In Schritt 6 und 7 wird jeweils eine Stichprobe zwischen den bestbewerteten Punkten betrachtet (Rekombination). Und um den besten Punkt wird durch leichte Variation (Mutation) geprüft, ob Verbesserungen möglich sind. Das globale Optimum wurde hier im Beispiel (noch) nicht gefunden.

2.3. Ein beispielhafter evolutionärer Algorithmus

Dieser Abschnitt entwickelt einen einfachen evolutionären Algorithmus am Beispiel des Handlungsreisendenproblems.

Für das Handlungsreisendenproblem (Definition 2.2 in Abschnitt 2.1) wird ein evolutionärer Algorithmus konstruiert, indem Schritt für Schritt die notwendigen Bestandteile zusammengestellt werden. Das Ziel ist es, ein Handlungsreisendenproblem mit 101 Städten schnell und mit ausreichender Qualität zu lösen. Bild 2.6 zeigt die Koordinaten eines Beispielproblems. Wie wir bereits in Abschnitt 2.2 erläutert haben, müssten wir für eine vollständige Suche $4{,}663\,1 \cdot 10^{157}$ Rundreisen untersuchen. In Anbetracht physikalischer Schätzungen, dass das Universum etwa 10^{78} Atome enthält bzw. seit dem Urknall etwa 10^{19} Sekunden verstrichen sind, liegt diese Zahl jenseits der menschlichen Vorstellungskraft. Jeglicher Versuch, das Problem durch systematisches Aufzählen aller Rundreisen zu lösen, ist unabhängig von der Schnelligkeit und der Anzahl an Prozessoren zum Scheitern verurteilt.

Es gibt sehr viele Möglichkeiten, einen evolutionären Algorithmus für das Handlungsreisendenproblem zu formulieren. Der hier beschriebene Ansatz ist nur ein einfaches einführendes Beispiel und weit vom derzeit besten bekannten Algorithmus entfernt.

Am Anfang ist zu entscheiden, wie der konkrete Raum der Individuen aussehen soll, auf dem der Algorithmus arbeitet. Da das Problem mit seiner Gütefunktion bereits auf Permutationen definiert wurde, liegt es nahe, diese direkt als Darstellung für die Individuen zu wählen: Also ist der Raum aller Lösungskandidaten $\Omega = \mathscr{S}_n$. Auf diesem Raum können nun geeignete Operatoren

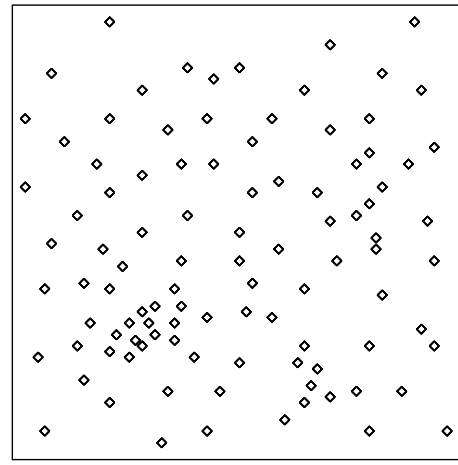


Bild 2.6.
Das Bild zeigt die Positionen der Städte für eine Beispielinstanz des Handlungsreisendenproblems mit 101 Städten.

zur Variation der Lösungskandidaten definiert werden. Dabei muss jeder Operator aus Permutationen wieder gültige Permutationen erzeugen (d. h. keine Zahl darf mehrfach in der Permutation vorkommen).

Zunächst entwerfen wir einen Mutationsoperator. Eine Möglichkeit für eine geringfügige Veränderung ist die VERTAUSCHENDE-MUTATION (Algorithmus 2.1), die zwei Zahlen in der Permutation miteinander vertauscht. Da sich lediglich die Position der Zahlen ändert, erzeugt der Operator für alle Permutationen und Zufallszahlen wieder eine gültige Permutation und stellt einen gültigen Mutationsoperator dar. So wird z. B. aus dem Individuum $(1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8)$ durch Anwendung des Operators mit den Zufallszahlen $u_1 = 2$ und $u_2 = 6$ das Individuum $(1,\ \underline{6},\ 3,\ 4,\ 5,\ \underline{2},\ 7,\ 8)$. Wie in Bild 2.7 links deutlich wird, werden bei der Anwendung des Operators aus der bestehenden Rundtour vier Kanten gestrichen und vier neue Kanten eingefügt. Das bedeutet, dass bei der Bewertung des neuen Individuums vier Kantengewichte abgezogen und vier Kantengewichte zur Güte des Ausgangsindividuums hinzuaddiert werden.

Ausgehend von der natürlichen Evolution hatten wir im letzten Abschnitt die Mutation als eine kleine Veränderung charakterisiert. Daher kann man sich an dieser Stelle fragen, ob die Mutation durch Tausch zweier Zahlen die kleinstmögliche Veränderung hinsichtlich des Handlungsreisendenproblems ist. Durch Ausprobieren an einem kleinen Beispiel findet man bald die INVERTIERENDE-MUTATION (Algorithmus 2.2), die ein Teilstück der Permutation invertiert (umkehrt). Auch hierbei werden mit der selben Begründung wie oben nur gültige Individuen erzeugt. Bei diesem Operator wird mit den Zufallszahlen $u_1 = 2$ und $u_2 = 6$ aus dem Individuum

Algorithmus 2.1

VERTAUSCHENDE-MUTATION(Permutation $A = (A_1, \ldots, A_n)$)
1 $B \leftarrow A$
2 $u_1 \leftarrow$ wähle Zufallszahl gemäß $U(\{1, \ldots, n\})$
3 $u_2 \leftarrow$ wähle Zufallszahl gemäß $U(\{1, \ldots, n\})$
4 $B_{u_1} \leftarrow A_{u_2}$
5 $B_{u_2} \leftarrow A_{u_1}$
6 **return** B

Algorithmus 2.2

INVERTIERENDE-MUTATION(Permutation $A = (A_1, \ldots, A_n)$)
1 $B \leftarrow A$
2 $u_1 \leftarrow$ wähle Zufallszahl gemäß $U(\{1, \ldots, n\})$
3 $u_2 \leftarrow$ wähle Zufallszahl gemäß $U(\{1, \ldots, n\})$
4 **if** $u_1 > u_2$
5 **then** ⊏ vertausche u_1 und u_2
6 **for each** $j \in \{u_1, \ldots, u_2\}$
7 **do** ⊏ $B_{u_2+u_1-j} \leftarrow A_j$
8 **return** B

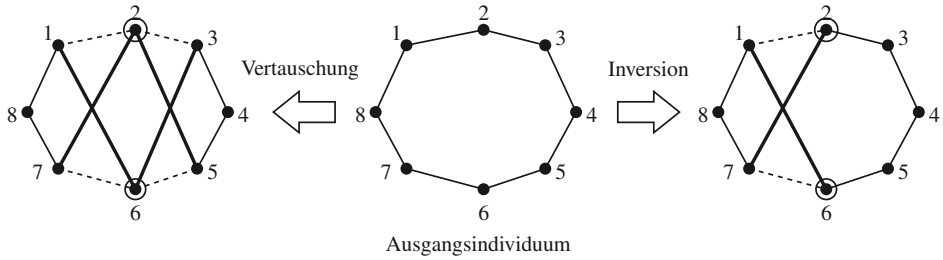


Bild 2.7. Veränderung bei der Anwendung von Mutationsoperatoren auf das in der Mitte dargestellte Individuum (1, 2, 3, 4, 5, 6, 7, 8). Die VERTAUSCHENDE-MUTATION resultiert in dem Individuum (1, 6, 3, 4, 5, 2, 7, 8), die INVERTIERENDE-MUTATION in dem Individuum (1, 6, 5, 4, 3, 2, 7, 8).

(1, 2, 3, 4, 5, 6, 7, 8) das Individuum (1, 6, 5, 4, 3, 2, 7, 8) erzeugt. Bild 2.7 zeigt rechts das Resultat dieser Mutation: Es werden lediglich zwei Kanten durch zwei neue Kanten ersetzt. Bezüglich der Bewertungsfunktion nimmt dieser Operator offensichtlich eine kleinere Veränderung an einer Rundreise vor.

Auf der Basis dieser Überlegung werden wir in unserem evolutionären Algorithmus für das Handlungsreisendenproblem dem Operator INVERTIERENDE-MUTATION den Vorzug geben.

Der zweite Operator ist die Rekombination, welche die Eigenheiten der Eltern mischen und auf das Kindindividuum übertragen soll. Diese Aufgabe erweist sich als nicht ganz so einfach, wie man zunächst annehmen könnte. Die Frage ist: Wie kann man möglichst große Teile der in den Elternindividuen vorliegenden Rundreisen in ein neues Individuum vererben, so dass keine gänzlich neue Rundtour entsteht.

In einem ersten Versuch, der ORDNUNGSREKOMBINATION (Algorithmus 2.3), übernehmen wir ein beliebig langes Präfix der einen Rundtour und fügen die restlichen Städte gemäß ihrer Reihenfolge in der anderen elterlichen Rundreise an. Durch die Abfrage in der zweiten for-Schleife wird auch hier die ausschließliche Erzeugung von gültigen Permutationen garantiert. Ein Beispiel für eine solche Berechnung ist in Bild 2.8 dargestellt. Wie man an diesem Beispiel sieht, kann es durchaus vorkommen, dass das Ergebnis des Operators stark von den Eltern abweicht – hier wurden zwei Kanten eingefügt, die in keinem der beiden Elternindividuen vorkamen. Der Name ORDNUNGSREKOMBINATION rührt daher, dass die Ordnung bzw. Reihenfolge der Städte erhalten bleibt.

http://www.springer.com/978-3-658-09957-2

Evolutionäre Algorithmen
Weicker, K.
2015, X, 331 S. 176 Abb., Softcover
ISBN: 978-3-658-09957-2