

# **Entwicklung und Erprobung eines Evolutionären Algorithmus zur rechnergestützten Optimierung der Maschinenbelegung in einem Druckgußwerk**

## **Diplomarbeit DI**

Vorgelegt dem Fachbereich Wirtschaftswissenschaften

- Fachgebiet Wirtschaftsinformatik der Produktionsunternehmen -

der

**Universität Gesamthochschule Essen**

von

Rodemeyer, Christian

Goldregenstraße 28

45770 Marl

Matr.-Nr. 888086

Erstgutachter: Prof. Dr. Ing. Dorloff  
Zweitgutachter: Prof. Dr. Dr. Hansohm

Eingereicht im Wintersemester 1994/95, 9. Studiensemester  
Voraussichtlicher Studienabschluß DI im Wintersemester 1994/95

# Inhaltsverzeichnis

<b>1 Einleitung</b> .....	<b>4</b>
1.1 Gegenstand und Ziel der Arbeit .....	4
1.2 Aufbau der Arbeit.....	5
<b>2 Optimieren mit Evolutionären Algorithmen</b> .....	<b>7</b>
2.1 Zum Begriff der Optimierung .....	7
2.2 Konventionelle Optimierungsverfahren .....	8
2.2.1 Analytische Optimierung.....	8
2.2.2 Numerische Optimierung .....	9
2.2.3 Enumeration .....	10
2.2.4 Zufallsstrategien .....	10
2.2.5 Heuristiken .....	11
2.2.6 Beurteilung und Konsequenzen.....	11
2.3 Evolutionäre Algorithmen.....	12
2.3.1 Chromosom als Parameterrepräsentation .....	15
2.3.2 Bewertung und Fitneß .....	17
2.3.3 Rekombination .....	19
2.3.4 Mutation .....	21
2.3.5 Populationsverhalten .....	22
2.3.6 Selbstadaption.....	25
2.4 Einordnung der Evolutionären Algorithmen.....	26
<b>3 Optimieren der Druckgußmaschinenbelegung</b> .....	<b>29</b>
<b>3.1 Problemstellung</b> .....	<b>29</b>
3.1.1 Unternehmenssituation .....	29
3.1.2 Fertigungssituation in der Gießerei .....	30
3.1.3 Optimierungsziele.....	32
3.1.4 Das Planungsproblem.....	34
<b>3.2 Lösungsansatz</b> .....	<b>34</b>

---

3.2.1 Ansätze in der Literatur der Evolutionären Algorithmen .....	36
3.2.2 Realisierter Ansatz.....	38
<b>3.3 Implementierung.....</b>	<b>39</b>
3.3.1 Repräsentation der Maschinenbelegungsparameter.....	39
3.3.2 Bewerten der Maschinenbelegung.....	43
3.3.3 Implementierte Zufallsprozesse.....	47
3.3.4 Rekombinationsstrategien .....	50
3.3.5 Mutation der Maschinenbelegung .....	53
3.3.6 Generationenübergang.....	55
3.3.7 Selbstadaption.....	56
<b>3.4 Erprobung .....</b>	<b>57</b>
3.4.1 Modellierter Planungssituation .....	57
3.4.2 Durchgeführte Experimente .....	59
3.4.3 Experimentelle Ergebnisse .....	60
3.4.4 Zusammenfassung und Auswertung.....	64
<b>4 Schlußbemerkungen .....</b>	<b>66</b>
<b>Anhang A: Inhalt der Diskette.....</b>	<b>67</b>
<b>Anhang B: Literaturverzeichnis.....</b>	<b>68</b>
<b>Anhang C: Eidesstattliche Versicherung .....</b>	<b>70</b>

# 1 Einleitung

## 1.1 Gegenstand und Ziel der Arbeit

Diese Arbeit beschäftigt sich mit einem unkonventionellen Ansatz zur Optimierung der Maschinenbelegung, der ein von der Natur seit langem erfolgreich angewandtes Optimierungsverfahren aufgreift, die Evolution. Durch die Anwendung so einfacher Prinzipien wie Mutation und Selektion hat die Natur es geschafft, für fast alle Umweltbedingungen nahezu optimal angepasste Lebewesen hervorzubringen. Mit einem so erfolgreichen Verfahren sollte es möglich sein, Optimierungsprobleme zu lösen, für die bisher keine zufriedenstellenden Lösungsverfahren existieren.

Mit Hilfe des Evolutionsprinzips soll die kurz- bis mittelfristige Planung der Maschinenbelegung in einem mittelständischen Industrieunternehmen, einem Druckgußwerk, optimiert werden. Bei der Planung wird aufgrund ihrer Komplexität und Bedeutung für den Produktionsprozeß nur die Gießereistufe berücksichtigt. Damit reduziert sich das Problem zwar auf eine einstufige, losweise Produktion, eine besondere Schwierigkeit stellt jedoch die Zuordnung von Fertigungsaufträgen zu den Druckgußmaschinen dar: jedes Gußteil kann von mehreren Maschinen produziert werden, aber nicht jede Maschine kann jedes Gußteil produzieren. Die Maschinen unterscheiden sich zudem durch unterschiedliche Geschwindigkeiten und Rüstzeiten.

Bis heute geschieht die Planung manuell mit Hilfe einer auf einem Gantt-Diagramm basierenden Plantafel. Der Disponent verläßt sich bei der Planung auf seine Erfahrung und Intuition. In naher Zukunft wird dieses System durch eine rechnergestützte Plantafel ersetzt, die zwar einige Routinearbeiten (z.B. die Berechnung der Dauer eines Auftrags aufgrund der Stückzahl) und Heuristiken (Losbildung) automatisiert, aber immer noch manuell optimiert werden muß. Langfristig wird jedoch das Ziel verfolgt, eine optimierende Planung zu implementieren. Aufgrund der Komplexität des Planungsproblems scheitern exakte Verfahren. Sowohl der Aufwand für die Erfassung und Bereitstellung aller benötigten Daten, als auch die zur Optimierung benötigte

Rechenkapazität ist wirtschaftlich nicht vertretbar. Der Ansatz der Evolution hat gegenüber den in PPS-Systemen eingesetzten Verfahren den Vorteil, robust auf sich ändernde Zielvorstellungen zu reagieren: ändern sich die Umweltbedingungen (Ziele), ändern sich die Lebewesen (Lösungsvorschläge), aber die Funktionsweise der Evolution (des Algorithmus) bleibt identisch. Der Einsatz prozeduraler Methoden hätte in diesem Fall den Wechsel zu einem anderen Algorithmus, und damit eine Änderung der Implementierung zur Folge.

Das Ziel dieser Arbeit ist es, einen geeigneten Algorithmus zu finden, der mit evolutionären Methoden das konkrete Planungsproblem im Druckgußwerk lösen, und in die rechnergestützte Plantafel eingebunden werden kann. Wenn dies gelingt, kann die Praxistauglichkeit des Verfahrens untersucht werden.

## 1.2 Aufbau der Arbeit

Die Arbeit ist in einen theoretischen und einen praktischen Teil gegliedert. Kapitel 2: *Optimieren mit Evolutionären Algorithmen* schildert kurz die Optimierungsproblematik und traditionelle Lösungsverfahren. Anschließend wird erörtert, wie die Evolution mit Hilfe des Computers simuliert und zur Lösung von Optimierungsaufgaben eingesetzt werden kann. Unter dem Oberbegriff der Evolutionären Algorithmen werden die Prinzipien der beiden wichtigsten Ansätze auf diesem Gebiet, den Genetischen Algorithmen und Evolutionsstrategien, gemeinsam beschrieben.

Kapitel 3: *Optimieren der Druckgußmaschinenbelegung* behandelt die Umsetzung der theoretischen Erkenntnisse in die Praxis. Der erste Abschnitt widmet sich der Planungssituation im Unternehmens. Unter Berücksichtigung der konkreten Anforderungen und Nebenbedingungen wird ein Lösungsansatz auf der Basis Evolutionärer Algorithmen entwickelt. Seine Implementierung wird in allen wichtigen Details beschrieben. Abschließend werden die Lösungseigenschaften verschiedener Varianten untersucht. Dazu wurden Testdaten erzeugt, die einen Teil des Unternehmens modellieren. Nach der Auswertung der Ergebnisse wird das für den praktischen Einsatz am besten

geeignete Verfahren bestimmt. Auf der beiliegenden Diskette im MS-DOS Format sind alle für den Test benutzten Daten sowie die Testergebnisse enthalten. Weiterhin finden sich dort der vollständige C++ Quelltext und ausführbare Programme aller implementierten Varianten .

## 2 Optimieren mit Evolutionären Algorithmen

### 2.1 Zum Begriff der Optimierung

Optimieren bedeutet im allgemeinen die beste Lösung für ein gegebenes Problem zu finden. Der Begriff der Optimierung beinhaltet damit zwei Aspekte, 1.) die beste Lösung und 2.) den Prozeß, mit dem sie gefunden wird. Es muß zwischen dem Ziel und dem Weg dorthin unterschieden werden. Das Ziel der Optimierung ist die Suche nach einem globalen Optimum, d.h. einem Parametervektor  $x^* \in D^n$  für den die Zielfunktion  $z: D^n \mapsto Z$  einen globalen Extremwert annimmt:  $\forall x \in D^n: z(x) \leq z(x^*)$  bzw.  $z(x) \geq z(x^*)$  (globales Maximum bzw. Minimum). Es gibt auch lokale Optima, dies sind Extremwerte innerhalb eines bestimmten Gebietes des Parameterraums:  $\exists e \in D^n: |x - x^*| < e \quad z(x) \leq z(x^*)$  bzw.  $z(x) \geq z(x^*)$  (lokales Maximum bzw. Minimum). Außerhalb der Umgebung  $e$  um das lokale Optimum  $x^*$  kann es noch größere (kleinere) Extremwerte geben. Das größte (kleinste) lokale Optimum ist gleichzeitig auch das globale Optimum. Dies ist insoweit interessant, als daß einige Optimierungsverfahren bereits beim Erreichen eines lokalen Optimums abbrechen, ohne das globale Optimum zu finden.

Jedes Verfahren, das versucht das Optimum zu finden, wird hier den Optimierungsverfahren zugeordnet. Die Qualität eines Verfahrens darf nun aber nicht nur aufgrund der Abweichung zwischen der gefundenen und der optimalen Lösung beurteilt werden, sondern auch die Prozeßeigenschaften wie Robustheit und Effizienz müssen berücksichtigt werden ([Goldberg]). Ein Methode, mit der effizient gute Näherungen ermittelt werden können, ist besser als ein Verfahren, welches das Optimum zwar theoretisch exakt bestimmt, in der Praxis aber ineffizient ist.

Robustheit ist ein Maß dafür, wie gut ein Verfahren in verschiedenen Situationen arbeitet. Ein Algorithmus, der an die Zielfunktion sehr restriktive Anforderungen stellt, wie z.B. Stetigkeit und Differenzierbarkeit, ist nicht robust, denn er versagt, sobald diese Annahmen nicht mehr zutreffen. Ein

robusteres Verfahren könnte dagegen sowohl stetige, als auch unstetige Optimierungsprobleme lösen, ohne dabei auf Ableitungen angewiesen zu sein.

Effizienz und Performance beeinflussen die Anwendbarkeit eines Verfahrens: selbst wenn in allen Situationen immer das globale Optimum gefunden wird, kann es doch nicht zum Einsatz gelangen, wenn der Aufwand außerhalb eines akzeptablen Bereichs liegt (z.B. eine zu hohe Rechenzeit).

Leider ist noch kein Optimierungsverfahren bekannt, das sowohl robust als auch effizient das globale Optimum exakt bestimmt. Eine gute Optimierungsmethode zeichnet sich daher dadurch aus, daß die obigen Kriterien in einem ausgewogenen Verhältnis zueinander stehen, denn „The most important goal of optimization is improvement“ ([Goldberg] 7). Das heißt nicht, daß spezialisierte Methoden in ihrem Anwendungsgebiet nicht sehr gute Ergebnisse liefern, aber außerhalb ihres Spezialgebietes sind sie nicht mehr effizient anwendbar.

Im folgenden werden die konventionellen Optimierungsverfahren grob klassifiziert und hinsichtlich der obigen Kriterien eingeschätzt. In diesen Kontext kann dann die Optimierung mit Evolutionären Algorithmen eingeordnet werden.

## **2.2 Konventionelle Optimierungsverfahren**

### **2.2.1 Analytische Optimierung**

Bei der *analytischen* oder *indirekten Optimierung* macht man sich die Eigenschaft stetig partiell differenzierbarer Funktionen zunutze, daß ihre ersten partiellen Ableitungen im Optimum alle gleich Null sind. Dies trifft allerdings auch für lokale Optima und Wendepunkte zu, das Verschwinden der ersten partiellen Ableitungen ist also nur eine notwendige, aber keine hinreichende Bedingung. Das tatsächliche globale Optimum muß noch unter den so gefundenen Punkten gesucht werden (z.B. durch Vergleich der Funktionswerte). Um die Nullstellen der partiellen Ableitungen zu bestimmen, wird die ursprüngliche Optimierungsaufgabe auf das Lösen eines

Gleichungssysteme zurückgeführt. Dieses ist nur dann linear, wenn die Ordnung der Zielfunktion kleiner als zwei ist. Aufgrund der notwendigen Differentiationsprozesse und der Lösung möglicherweise nichtlinearer Gleichungssysteme ist die Programmierung einer analytischen Optimierung eine schwere Aufgabe. Will man auch Unstetigkeiten und Nebenbedingungen berücksichtigen, wird dieses Verfahren noch komplexer. Analytische Methoden haben allerdings den Vorteil, daß sie bei Erfüllung aller Voraussetzungen das Optimum exakt bestimmen können.

### 2.2.2 Numerische Optimierung

Mit *numerischer* oder *direkter Optimierung* bezeichnet man Verfahren, die sich dem Optimum schrittweise nähern. Die *Hill-Climbing Strategien* sind die bekannteste Gruppe dieser Verfahren. Ihr Name beschreibt anschaulich die Vorgehensweise: die Zielfunktion stellt man sich als Gebirge vor, in dem ein (blinder) Bergsteiger versucht den höchsten Gipfel (oder das tiefste Tal) durch Betasten seiner Umgebung zu finden. Für die Steuerung des Bergsteigers gibt es verschiedene Möglichkeiten, die Gradienten-Strategie z.B. geht von jedem Punkt des Gebirges aus einen Schritt in die Richtung des steilsten Anstiegs (partielle Ableitung der Zielfunktion), wobei die Schrittweite proportional zur Steigung ist, d.h. in der Nähe eines Optimums wird die Schrittweite immer kleiner. Diese iterativen Verfahren sind sehr gut programmierbar, da der zugrundeliegende Algorithmus direkt in ein Programm umgesetzt werden kann. Außerdem kann statt mit partiellen Ableitungen meist mit Differenzenquotienten gearbeitet werden, die sich ja einfach durch Berechnen von Funktionswerten bestimmen lassen. Da diese Verfahren lokal arbeiten besteht die Gefahr, das globale Optimum zu verfehlen, und nur ein lokales Optimum zu finden. Durch mehrere Durchläufe mit unterschiedlichen (stochastisch oder deterministisch gewählten) Startpunkten kann dieses Risiko zu Lasten der Effizienz minimiert werden.

### 2.2.3 Enumeration

Bei diskreten oder gemischt-ganzzahligen Optimierungsproblemen versagen analytische und numerische Methoden. Für diesen Fall kommen *enumerative Verfahren* in Betracht. Im allgemeinsten Fall, der *vollständigen Enumeration*, werden alle möglichen Parameterkombinationen systematisch überprüft. Auf den ersten Blick scheint dies eine sehr natürliche und einfach zu programmierende Lösung zu sein, die garantiert das globale Optimum findet. Bei praxisrelevanten Problemen scheitert sie jedoch häufig an der zu hohen Zahl möglicher Parameterkombinationen. Dadurch wird eine effiziente Lösung verhindert. Als Reaktion darauf wurden Verfahren wie die *dynamische Programmierung* oder *Branch-and-Bound* entwickelt, die die Zahl der zu untersuchenden Möglichkeiten drastisch reduzieren können, allerdings nur dann, wenn wieder gewisse Annahmen zutreffen. Trotzdem versagen auch diese Verfahren häufig an der hohen Komplexität realer Probleme.

### 2.2.4 Zufallsstrategien

Bei den *Zufallsstrategien* unterscheidet man zwischen den reinen Verfahren (pure random search) und solchen, die im Prinzip deterministisch vorgehen, aber Zufallsprozesse systematisch oder in ausweglosen Situationen nutzen (randomized techniques) ([Schwefel] 101ff). Wenn z.B. eine Hill-Climbing-Methode an einen Punkt gelangt, von dem aus die Zielfunktion in jeder Richtung den gleichen Anstieg hat, kann der Zufall entscheiden, in welcher Richtung weitergesucht wird. Bei der reinen Zufallssuche wird der Funktionswert für zufällig gewählte Punkte des Parameterraums berechnet. Aus dieser Stichprobe wird auf das Optimum geschlossen. Die *Monte-Carlo-Methode* ist ein Beispiel für eine reine Zufallssuche. Reine Zufallsverfahren sind robust, da sie nur eine einzige Bedingung an die Zielfunktion stellen, nämlich daß sich ein Funktionswert für eine gegebene Parameterkombination berechnen läßt. Der Nachteil dieser Verfahren ist klar: es werden nur Werte in der Nähe des globalen Optimums gefunden, das exakte Optimum zu treffen wäre „Zufall“. Aber je mehr Versuche durchgeführt werden, desto höher ist die

Wahrscheinlichkeit eines Treffers in der Nähe des Optimum. Gutes Konvergenzverhalten und Effizienz stehen konträr zueinander.

### 2.2.5 Heuristiken

Wenn alle deterministischen oder stochastischen Verfahren scheitern, werden oft *Heuristiken* eingesetzt. Dies sind „Rezepte“ von denen man hofft, daß sie zu einer guten Lösung führen. Heuristiken sind daher zwar leicht anzuwenden und zu programmieren, aber aufgrund der willkürlichen festgelegten Regeln ist es selten möglich, eine theoretisch fundierte Aussage darüber zu machen, wie nahe die gefundene Lösung am tatsächlichen Optimum liegt. Heuristiken sind häufig sehr spezialisiert. Sie versagen bei Problemen, für die sie nicht konzipiert wurden, und schon wenn die getroffenen Annahmen nicht exakt zutreffen, kann sich ihre Qualität und Effizienz stark verschlechtern.

### 2.2.6 Beurteilung und Konsequenzen

Alle konventionellen deterministischen Optimierungsmethoden leiden entweder an mangelnder Robustheit („...: conventional search methods are not robust.“ ([Goldberg] 5)) oder Effizienz. Der Grund dafür ist das feste innere Modell der Zielfunktion, das diese Methoden benutzen, um die im Laufe der Iterationen gesammelten Informationen zu interpretieren ([Schwefel] 104-105). Sobald die Annahmen über dieses Modell nicht mehr zutreffen, kann das Verfahren das Optimum nicht mehr sicher finden oder wird ineffizient. Die Gefahr besteht nun darin, daß das Optimum, ohne daß man es weiß, gar nicht gefunden werden kann: „Wenn man nicht weiß, wo die Optima im Suchraum liegen und wie sie aussehen, so weiß man in der Regel auch nicht, ob man die richtige Suchstrategie gewählt und die richtigen Annahmen getroffen hat.“ ([Schöneburg2] 105). Bei den Zufallsstrategien, die keine Annahmen über die Struktur der Zielfunktion treffen, läßt sich bei Kenntnis der zugrundeliegenden Zufallsprozesse zumindest die Wahrscheinlichkeit abschätzen, mit der man sich dem Optimum nähert.

Das hat für eine rechnergestützte Optimierung die Folge, daß der implementierte Algorithmus bei jeder Änderung der Optimierungssituation (neue oder

geänderte Annahmen) auf seine Anwendbarkeit überprüft werden muß. Wenn nicht, wird eine Anpassung nötig, die im Extrem eine Neuprogrammierung des Verfahrens bedeutet. Jede Änderung des Programmcodes oder der Datenstruktur läßt sich aber nur unter hohem Aufwand realisieren, insbesondere wenn sich das System bereits im Einsatz befindet. Nicht robuste Verfahren erfordern einen höheren Wartungsaufwand. Da die Umweltbedingungen in der Realität einem ständigen Wandel unterliegen, sollten robuste Verfahren, die keine besondere Anpassung benötigen, und somit einen geringeren Wartungsaufwand verursachen, bevorzugt werden.

In der Praxis werden für die Maschinenbelegungsplanung hauptsächlich Heuristiken eingesetzt, z.B. Prioritätsregeln. Exakte Verfahren, wie z.B. das auf gemischt-ganzzahliger Programmierung beruhende Modell von Manne (vgl. [Kistner] 122ff), scheitern an der Komplexität des Problems. Aber allein um ein Modell aufzustellen, das theoretisch exakt lösbar ist, müssen bereits so viele Abstraktionen von der Wirklichkeit vorgenommen werden, daß sich die Frage stellt ob die gefundene Lösung, wenn sie denn in endlicher Zeit ermittelt werden könnte, überhaupt noch mit dem realen Optimum übereinstimmt.

In diesem Fall erscheint es sinnvoller, ein robust und effizient arbeitendes Verfahren zu benutzen, das zwar das Optimum nicht mit Sicherheit exakt bestimmen kann, in das sich dafür aber alle wesentlichen Nebenbedingungen und Einflußfaktoren leicht integrieren lassen, und das sich somit dem realen Optimum nähert und nicht dem theoretischen, das auf möglicherweise wirklichkeitsverfremdenen Annahmen beruht. Ein solches Verhalten erhofft man sich von den Evolutionären Algorithmen.

### **2.3 Evolutionäre Algorithmen**

Für die Entwicklung des Lebens, wie wir es kennen, spielte die Evolution eine bedeutende Rolle. Sie sorgte für eine optimale Anpassung der Lebewesen an ihre Umwelt. Durch die Vielzahl der unterschiedlichsten Umweltverhältnisse entstanden die Arten. Jede ist ihrem Lebensraum nahezu optimal angepaßt. Aber auch auf Umweltveränderungen im Zeitablauf reagierte die Evolution

flexibel, die Lebewesen wurden ständig den sich verändernden Bedingungen angepaßt. Die Evolution kann somit als ein natürlicher, dynamischer Optimierungsprozeß interpretiert werden. Der „Parameterraum“, in dem das „Optimum“ gesucht wird, ist extrem groß. Seine Komplexität überschreitet jedes vom Menschen angegangene Problem bei weitem. Interessant ist, daß die Evolution zur Optimierung keine globalen, übergeordneten Strukturen einsetzt und mit wenigen einfachen Grundprinzipien effizient sehr gute Lösungen findet. Es bietet sich an, diesen natürlichen Optimierungsprozeß auch auf künstliche Systeme zu übertragen. Verfahren, die die Evolution simulieren, werden *Evolutionäre Algorithmen* (EA) genannt.

Jeder Organismus ist Träger von Erbinformationen, die in Form von Genen auf seinen Chromosomen abgelegt sind. Die Erbinformation kann als Vorschrift zum Aufbau des Lebewesens interpretiert werden. Sie bestimmt somit indirekt, wie gut der Organismus an seine Umwelt angepaßt ist. Je besser die Anpassung, desto höher ist die Wahrscheinlichkeit, sich fortpflanzen zu können. Bei der sexuellen Fortpflanzung werden die Gene zweier Organismen neu miteinander kombiniert, so daß die Nachkommen Eigenschaften beider Elternteile erben. Durch Mutationen treten zufällige Veränderungen im Erbgut auf. So werden neue Varianten eingeführt. Die Evolution übt also keinen direkten Einfluß auf die Individuen aus, sondern die Veränderungen finden auf der Ebene der Chromosomen und Gene statt. In der Natur existieren normalerweise Populationen von Organismen. Damit kann die Evolution parallel, d.h. zeitgleich, die Tauglichkeit vieler Individuen prüfen.

EA simulieren die Grundprinzipien der natürlichen Evolution um optimale Lösungen zu finden. Sie wollen aber nicht jedes Detail der Natur kopieren, sondern versuchen vielmehr eine Art „idealisiertes Grundschema“ zu finden, welches die nützlichsten Elemente des Vorbildes enthält. Die Simulation der Evolution erfolgt hauptsächlich mit Hilfe von Computern, obwohl auch experimentelle Verfahren möglich und angewandt worden sind ([Rechenberg]). Die meisten EA-Anwendungen reduzieren die Erbinformationen eines Individuums auf ein einziges Chromosom. Die Begriffe Individuum und Chromosom werden in diesem Fall synonym benutzt. Das

Chromosom wird häufig als ein Vektor fester Länge von Genen abgebildet:  $\langle g_1, g_2, \dots, g_n \rangle$ . In ihm sind die Parameter des zu lösenden Optimierungsproblems kodiert. Die EA betrachten nicht mehr die ursprünglichen Parameter, sondern arbeiten ausschließlich auf der Basis der Chromosomen. Daraus resultiert ein großer Teil der Robustheit dieses Verfahrens, denn derselbe Algorithmus kann auf eine Vielzahl vollkommen unterschiedlicher Probleme angewandt werden. Die Individuen werden in diskreten Zeitpunkten reproduziert, die so entstehenden aufeinanderfolgenden Populationen werden Generationen genannt. Beim Generationenübergang kommt das Prinzip der Selektion zur Anwendung: gute Individuen überleben, schlechte sterben aus. Die Güte der Individuen steigt im Laufe der Generationen immer weiter an.

Nur bei der Ermittlung der Güte eines einzelnen Individuums wird explizit problemspezifisches Wissen benötigt: das Chromosom wird in die ursprünglichen Parameter dekodiert, diese werden in die Zielfunktion eingesetzt. Das Ergebnis ist eine Bewertung des Individuums, seine Tauglichkeit. Selbst wenn die Zielfunktion nicht exakt bekannt (oder effizient berechenbar) ist, genügt es, eine Methode zu kennen, mit der man unterschiedliche Parameterkonstellationen relativ zueinander bewerten kann. Ändert sich die Zielfunktion, muß nur die Bewertungsfunktion angepaßt werden, der Grundalgorithmus bleibt unverändert!

Die Möglichkeit, mit Hilfe der Evolution Optimierungsprobleme zu lösen, haben Ende der 60-ziger Jahre Ingo Rechenberg und John H. Holland unabhängig voneinander erkannt. Rechenberg versuchte mit seiner *Evolutionstrategie* (ES) technische Probleme zunächst experimentell zu lösen, während sich Holland dafür interessierte, wie sich der Anpassungsprozeß natürlicher Systeme (z.B. Lebewesen) erklären läßt, und wie dieser von natürlichen Systemen auf Softwaresysteme übertragen werden kann. Dabei richtete sich sein Augenmerk auf den Aufbau der Erbinformation, die Chromosomen und Gene, und wie durch ihre Veränderung Anpassungsprozesse realisiert werden. Auf ihn gehen die *genetische Algorithmen* (GA) zurück. Obwohl sich beide Ansätze vom Prinzip her stark

ähneln, gibt es in den Details teilweise gravierende Unterschiede. Unter dem Begriff „Evolutionärer Algorithmus“ wird eine Obermenge beider Ansätze verstanden. Dadurch hat man die Möglichkeit, die für ein gegebenes Problem besten Ideen zu benutzen, ohne sich auf eine Schule (GA oder ES) festlegen zu müssen. Bei der genaueren Betrachtung der einzelnen Elemente in den folgenden Kapiteln, bleiben aber bedeutende Unterschiede zwischen den beiden Ansätzen nicht unerwähnt.

### 2.3.1 Chromosom als Parameterrepräsentation

Evolutionäre Algorithmen arbeiten mit einer Repräsentation der Parameter des Optimierungsproblems. Diese wird in Analogie zur Genetik als Chromosom bezeichnet. Wie sieht dieses „künstliche“ Chromosom nun im Detail aus? Angenommen, das Optimum der Funktion  $f(x_1, x_2, \dots, x_n)$ ,  $x \in \mathbf{R}$  soll gesucht werden. Der natürlichste Aufbau des Chromosoms ist dann ein Vektor reeller Zahlen  $\langle g_1, g_2, \dots, g_n \rangle$ , wobei jedes Gen einem Parameter der Zielfunktion entspricht:  $g_n = x_n$ . Dies ist die Standardvorgehensweise der Evolutionsstrategien.

Der Ansatz der Genetischen Algorithmen orientiert sich näher an der Informationsspeicherung der Natur. Die Erbinformation wird in natürlichen Organismen nämlich durch eine lineare Folge von nur vier chemischen Bausteinen<sup>1</sup> gespeichert. Jede Position des Chromosoms kann nur einen der als Alphabet des Lebens bezeichneten vier Buchstaben aufnehmen. GA's arbeiten daher im Gegensatz zu Evolutionsstrategien mit Chromosomen, die die Parameter des zu lösenden Optimierungsproblems diskret repräsentieren. Wegen dem „Prinzip des minimalen Alphabets“ („The user should select the smallest alphabet that permits a natural expression of the problem.“ ([Goldberg] 80)) halten sie in den meisten Fällen eine binäre Codierung für optimal, d.h. das Alphabet enthält nur die zwei Werte 0 und 1. Ein Chromosom besteht aus einer Kette von booleschen Werten, die einen „Bitstring“ bilden. Ein Gen ist ein einzelnes Bit oder eine Folge davon. Die

---

<sup>1</sup>den Basen Adenin (A), Cytosin (C), Guanin (G) und Thymin (T), vgl. /xxx/

Gene eines Chromosoms können unterschiedliche Längen besitzen. Für die Implementierung auf Computern hat die binäre Kodierung zunächst nur Vorteile, bei reellen Parametern, wie im obigen Beispiel, können sich jedoch diffizile Schwierigkeiten bei der Mutation und Rekombination ergeben (vgl. [Schöneburg2] 189 ff.).

In der Natur ist die Bedeutung eines Gens losgelöst von seiner Position im Chromosom, d.h. das Gen für die Haarfarbe erfüllt seine Funktion unabhängig davon, ob es sich am Anfang oder Ende des Chromosoms befindet. Bei einem Gen muß man also zwischen seiner Funktion (Bestimmung der Haarfarbe), seinem Wert (schwarz, blond, ...) und seiner Position im Chromosom unterscheiden ([Goldberg] 166ff). Dies erreicht man durch einfaches Durchnumerieren der Gene, wie Abbildung 1 zeigt. Aber auch die Reihenfolge, in der die Gene angeordnet sind, kann eine Bedeutung für das Individuum und seine Tauglichkeit haben. Durch die Numerierung wird diese für die Bewertungsfunktion sichtbar und kann ebenfalls berücksichtigt werden.

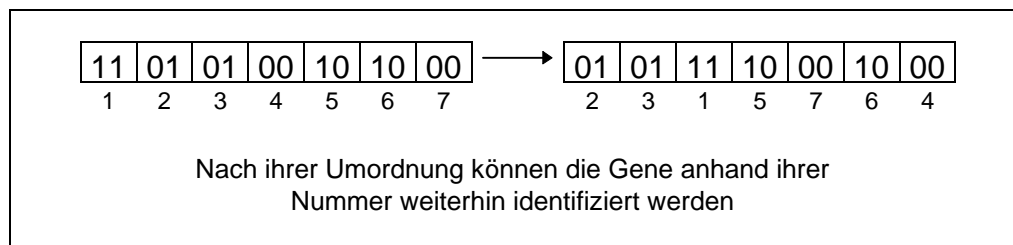


Abbildung 1: Positionsunabhängigkeit der Gene

Die Kodierung der Parameter trennt den Optimierungsalgorithmus vom zu lösenden Problem. Insbesondere GA besitzen durch die reine Binärcodierung (zwischen Gen und Parameter ist kein direkter Zusammenhang mehr erkennbar) keine Kenntnis vom Aufbau der Zielfunktion, sie sind blind. ES kennen zumindest die Dimension des Parameterraums (Anzahl der Parameter). Diese Entkopplung ist ein wesentlicher Grund für die Robustheit des Verfahrens.

Das erste Problem bei der Realisierung eines EA ist die Wahl einer geeigneten Parameterrepräsentation. Allgemein läßt sich festhalten, daß bei reellen Parametern die Methode der Evolutionsstrategien, bei diskreten Problemen jedoch eine Kodierung nach dem GA-Ansatz günstiger ist ([Schöneburg2]

190). Der Aufbau des Chromosoms und seiner Gene muß natürlich mit den noch zu erörternden genetischen Operatoren und der Bewertungsfunktion abgestimmt sein, damit das Verfahren korrekt funktioniert.

### 2.3.2 Bewertung und Fitneß

Die Bewertungs- oder Qualitätsfunktion bestimmt das eigentliche Optimierungsziel. Sie ermittelt für ein Chromosom einen Wert, der ein Maß für die Güte der Lösung ist. Evolutionäre Algorithmen versuchen nun Chromosomen zu finden, die diesen Wert maximieren bzw. minimieren. In der Qualitätsfunktion ist das für die Bewertung benötigte problemspezifische Wissen unabhängig vom eigentlichen Algorithmus gekapselt. Dies hat den Vorteil, daß sich Änderungen in der Zielfunktion, wie z.B. die Hinzunahme eines neuen Optimierungskriteriums, nur auf das Bewertungsmodul des Algorithmus auswirken.

Der erste Schritt bei der Bewertung ist die Dekodierung des Chromosoms in die ursprünglichen Parameter. Aufgrund dieser Parameter wird dann die Qualität der Lösung bestimmt. Dies kann durch einfaches Einsetzen in die Zielfunktion geschehen. Wenn die Zielfunktion nicht in geschlossener Form vorliegt, kann die Bewertung aber auch durch einen beliebig komplexen Algorithmus erfolgen. Soll z.B. ein Kommissionierungssystem optimiert werden, könnte man zunächst ein Simulationsmodell erstellen. Parameter des Modells wären Lagerkapazität, Anzahl Transporter, etc. Durch Simulation aufgrund der dekodierten Parameter können dann die eigentlichen Bewertungskriterien wie Auslastung und Reaktionsgeschwindigkeit, oder auch Kosten ermittelt werden. Durch eine Verknüpfung dieser Kriterien, z.B. über eine Nutzwertanalyse, erhält man schließlich eine Maßzahl für die Bewertung des Chromosoms.

Das Einhalten von Nebenbedingungen ist eine weitere wichtige Aufgabe: ist das Chromosom so aufgebaut ist, daß auch ungültige Lösungen dargestellt werden können (vgl. Kapitel 2.3.3: *Rekombination*), muß die Bewertungsfunktion aktiv werden. Ein Verstoß gegen Nebenbedingungen kann mit einer unendlich schlechten Qualität bewertet werden, so daß der nachfolgende

Selektionsprozeß die ungültigen Individuen eliminiert. Dies ist aber nicht immer eine gute Idee: die ungültige Lösung könnte ja ein Zwischenschritt zum Optimum sein, oder der Parameterraum ist so beschaffen, daß zwischen sehr vielen ungültigen Lösungen nur wenige gültige liegen. Ebenso können Nebenbedingungen formuliert sein, die nicht in jeder Situation erfüllbar sind! Angenommen für einen Produktionsplan gelte die Nebenbedingung, daß keine Verspätungen auftreten dürfen. Sobald der Auftragseingang die vorhandene Produktionskapazität übersteigt, ist dies aber nicht mehr möglich. Es können keine Pläne mehr generiert werden, da jede Lösung ungültig ist. Daher ist es besser, für Nebenbedingungen sogenannte Straffunktionen einzuführen: sie verschlechtern die Bewertung in Abhängigkeit vom Ausmaß, in dem gegen die Nebenbedingung verstoßen wird.

Die Bewertung ist die Grundlage der in EA stattfindenden Selektionsprozesse. Solange die Auswahl deterministisch getroffen wird, z.B. wähle die  $n$  besten Individuen, reicht eine Bewertung im obigen Sinne aus. Die natürliche Evolution arbeitet jedoch mit einer stochastischen Auswahl. Wenn man dieses Prinzip simulieren möchte, benötigt man für jedes Individuum einer Population eine Wahrscheinlichkeit, mit der es ausgewählt wird. Diese Wahrscheinlichkeit wird als Fitneß bezeichnet. Da die Bewertung eine beliebige reelle Zahl sein kann, die Fitneß jedoch aus dem Intervall  $[0;1]$  stammen und ihre Summe über alle Individuen gleich Eins sein muß, ist eine Funktion notwendig, die die Fitneß in Abhängigkeit von der Bewertung ermittelt ([Goldberg] 76ff).

Ein häufig angewandtes Verfahren ist die proportionale Fitneß, die die Auswahlwahrscheinlichkeit je Individuum wie folgt bestimmt:

$$Fitneß_i = \frac{Bewertung_i}{\sum Bewertung}$$

Wenn mit kleinen Populationen gearbeitet wird,

können bei dieser Variante einige besonders gute Individuen schnell die gesamte Population dominieren, d.h. sie besitzen eine so hohe Fitneß, daß andere Individuen kaum noch die Chance besitzen, ausgewählt zu werden. Dies reduziert über kurz oder lang die genetische Vielfalt, wodurch die Gefahr besteht, daß der Algorithmus zu früh konvergiert und in einem lokalen

Optimum stecken bleibt. Ein ähnliches Problem tritt auf, wenn die Bewertungen innerhalb der Population sehr ähnlich sind. Dann ist die Wahrscheinlichkeit, selektiert zu werden, annähernd gleich und das Optimum wird erst nach sehr vielen Schritten gefunden. Zur Vermeidung dieser Probleme wird die Fitness skaliert. Die Skalierung soll nahe zusammenliegende Werte auseinander ziehen und Ausreißer zurückholen. Die relative Fitness der Chromosomen untereinander darf sich aber nicht ändern.

Abschließend ergibt sich die folgende Vorgehensweise bei der Bewertung: das Chromosom wird dekodiert und die resultierende Parameterkombination mit Hilfe der Zielfunktion bewertet. Das Ergebnis wird skaliert, um extreme Unterschiede oder Ähnlichkeiten innerhalb der Population auszugleichen.

### **2.3.3 Rekombination**

Reproduziert sich ein Organismus, erben seine Nachkommen seine Erbinformation. Geschieht dies ungeschlechtlich, z.B. durch Zellteilung, können Veränderungen nur durch Fehler beim Kopieren der Erbinformation entstehen, neues genetisches Material wird relativ selten in eine Population eingeführt. Bei der geschlechtlichen Fortpflanzung dagegen wird das Erbmaterial von zwei Organismen neu miteinander kombiniert. Dadurch entsteht bei fast jeder Reproduktion neues Material. In der Natur dominiert die zwei-geschlechtliche Fortpflanzung, weshalb man sich von einer Nachahmung dieses Prinzips Vorteile verspricht.

Bei der natürlichen Rekombination brechen die Chromosom an zufälligen Positionen auseinander (stark vereinfacht ausgedrückt), wodurch Teilstücke entstehen. Die Teilstücke der Elternchromosomen verbinden sich „überkreuz“ wieder miteinander und bilden so ein neues vollständiges Chromosom. Dieses als „Crossing Over“ bezeichnete Rekombinationsverfahren ist Grundbestandteil jedes Genetischen Algorithmus. Hauptsächlich durch die Neuordnung existierender Parameterkombinationen wird neues Erbmaterial (Punkte des Parameterraums) in den Evolutionsprozeß eingebracht. Abbildung 2 zeigt das realisierte Prinzip: Beim einfachen Ein-Punkt-Crossover wird zufällig eine Position gewählt, an der das Chromosom bricht. Jede mögliche

Position hat die gleiche Auswahlwahrscheinlichkeit. Im allgemeineren Fall des  $n$ -Point-Crossover werden  $n$  Bruchstellen gewählt.

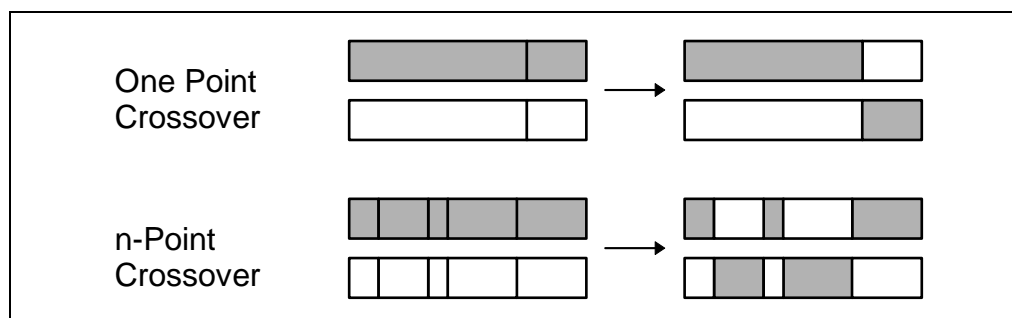


Abbildung 2: Crossover Prinzip

Dieses einfache Verfahren kann aber zu Problemen führen. Angenommen, ein Chromosom benutzt Gene, deren Bedeutung von ihrer Position unabhängig ist (vgl. Kapitel 2.3.1). Das oben beschriebene Verfahren kann dazu führen, daß in den Chromosomen einige Gene doppelt auftauchen, während andere ganz fehlen. In einigen Anwendungen kann dies zu „ungültigen“ Parameterkombinationen führen. Dies soll am Beispiel des Travelling-Sales-Person- Problems, einem klassischen Reihenfolgeproblem, erläutert werden.

Es geht darum, eine Menge vorgegebener Städte genau einmal zu besuchen, und dabei den kürzesten Weg zurückzulegen. Jede Stadt wird durch ein Gen kodiert (Funktion des Gens), die Reihenfolge, in der die Gene im Chromosom angeordnet sind, ist die Reihenfolge, in der die Städte besucht werden. Jedes Gen darf nur einmal, aber an beliebiger Position im Chromosom auftreten. Der konkrete Wert eines Gens hat keine Bedeutung. Abbildung 3 zeigt einen 2-Punkt-Crossover, der ungültige Individuen produziert. Nach der Reproduktion treten im unteren Chromosom die Städte F und G zweimal auf, die Städte E und H überhaupt nicht.

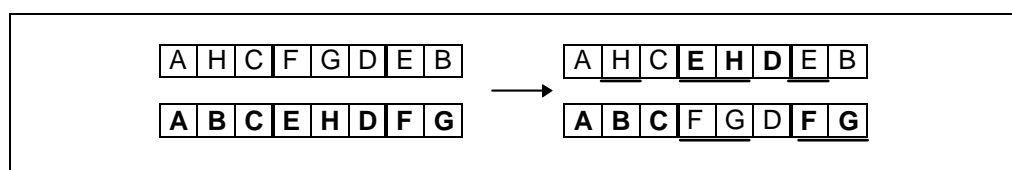


Abbildung 3: Crossover erzeugt ungültige Individuen

Für solche Situationen sind besondere Crossover-Verfahren entwickelt worden (vgl. [Goldberg]). Dies beschränkt den Algorithmus zwar auf bestimmte

Anwendungsgebiete, in diesem Fall auf Reihenfolgeprobleme, aber auch die Konvergenzgeschwindigkeit und Konvergenzsicherheit wird verbessert.

Die Evolutionsstrategien könnten die Rekombination nach dem gleichen Prinzip verwirklichen, aufgrund der Darstellung des Chromosoms als Vektor reeller Zahlen sind aber auch andere Varianten entwickelt worden: jedes Gen eines neuen Individuums wird von einem anderen, zufällig gewählten Elter genommen. Dies ist der Übergang von der Bisexualität zur Multisexualität ([Schwefel] 171). Eine weitere Möglichkeit ist die arithmetische Kombination von Genen, wenn z.B. einem neuzubildenden Gen der Mittelwert der gewählten Elterngene zugewiesen wird. Die Evolutionsstrategien messen der Rekombination jedoch eine geringere Bedeutung zu als die Genetischen Algorithmen und benutzen primär die Mutation um neue Punkte des Parameterraums zu erforschen.

#### **2.3.4 Mutation**

Die Mutation ist eine zufällige Veränderung der Erbinformation. Dabei sind kleine Änderungen häufiger als große. Man geht davon aus, daß die Verteilung der Änderungen durch eine Normalverteilung mit dem Erwartungswert Null (keine Änderung) approximiert werden kann. Mutationen erzeugen neues genetisches Material, also Parameterkombinationen, die durch Rekombination allein nicht entstehen können. Der Einsatz von Mutationen soll verhindern, daß der Algorithmus in lokalen Optima „steckenbleibt“ und das globale Optimum verfehlt.

Für die Evolutionsstrategien bedeutet der Mutations-Operator noch wesentlich mehr, er ist der primäre Antrieb bei der Erkundung neuer Gebiete im Parameterraum. Zur Mutation wird ein Vektor mit normalverteilten Zufallszahlen erzeugt, der zu dem Chromosom (ein Vektor reeller Zahlen) addiert wird. Der Erwartungswert  $\mu$  ist gleich Null, die Standardabweichung  $\sigma$  bestimmt die „Bandbreite“ der Mutation. Dieser Wert kann als Schrittweite verstanden werden ([Schwefel]). Ein Problem stellt die Wahl der richtigen Schrittweite dar: ist sie zu klein, werden viele Generation gebraucht um das Optimum zu finden, ist sie zu groß wird das Optimum nur grob angenähert. ES

können daher für jedes Gen (Parameter) eine eigene Standardabweichung festlegen, welche während der Optimumsuche automatisch angepaßt wird. Diese Selbstadaption oder Schrittweitensteuerung geschieht ebenfalls nach der Methode der Evolution. Das Chromosom wird einfach um den Vektor  $\langle \sigma_1, \sigma_2, \dots, \sigma_n \rangle$  erweitert, wodurch die Schrittweiten mit optimiert werden.

GA messen der Mutation nur eine zweitrangige Bedeutung zu, nämlich zu verhindern daß bestimmte Genkombinationen aussterben ([Goldberg] 14). In einem binär kodierten Chromosom wird eine Mutation durch die Invertierung eines zufällig gewählten Bits realisiert. Mutationen treten im Gegensatz zu ES nicht bei jeder Reproduktion, sondern nur sehr selten auf. Goldberg empfiehlt eine Mutation je tausend Bit ([Goldberg] 14).

An dem Unterschied zwischen GA und ES wird klar, daß die Art, in der die Mutation durchgeführt wird, von der Art der Kodierung abhängt. In dem oben geschilderten TSP-Beispiel kann eine Mutation nicht einfach durch Invertieren von Bits oder der Addition einer Zufallszahl erfolgen, weil dadurch die formale Gültigkeit des Individuums zerstört würde. Ein angemessenerer Mutations-Operator wäre in diesem Fall das Vertauschen zweier Gene, wobei die Distanz zwischen den Genen ein Maß für die Schrittweite ist.

### 2.3.5 Populationsverhalten

Das Populationsverhalten beschäftigt sich mit der Frage, welche Schritte zur Erzeugung der Generation  $n+1$  aus der Generation  $n$  notwendig sind. Für diese Aufgabe wurden von beiden Ansätzen eine Fülle von Varianten entwickelt. Um diese gemeinsam erläutern zu können, wurde versucht, in Abbildung 4 eine Obermenge des Verhaltens beider Ansätze darzustellen. Aus diesem allgemeinen Ansatz kann jedes konkrete Populationsverhaltens eines Evolutionären Algorithmus abgeleitet werden.

Die Rechtecke in Abbildung 4 symbolisieren alle Individuen einer Generation, die Kreise stehen für (Sub-)Populationen und über die Kanten werden Individuen zwischen den Populationen transportiert. Hinter jeder Kante stehen Selektionsprozesse. Diese können entweder stochastisch oder deterministisch

sein. Die deterministische Selektion wählt entweder die  $n$  besten, alle oder keine Individuen. Die letzten beiden Möglichkeiten sind „entartet“, sie werden aber benötigt, falls nicht hinter jeder Kante eine Selektion steht. Die simulierte Evolution beginnt mit einer zufällig erzeugten Startpopulation. Diese bildet gleichzeitig die erste Elternpopulation  $E$ , aus der die Kinder  $K$  erzeugt werden. Sowohl Kinder als auch Eltern können in die Übergangspopulation  $\ddot{U}$  eingehen, aus der dann die Elternpopulation  $E'$  der nächsten Generation entsteht. Interessant ist nun die Realisierung der Verbindungen zwischen den Subpopulationen.

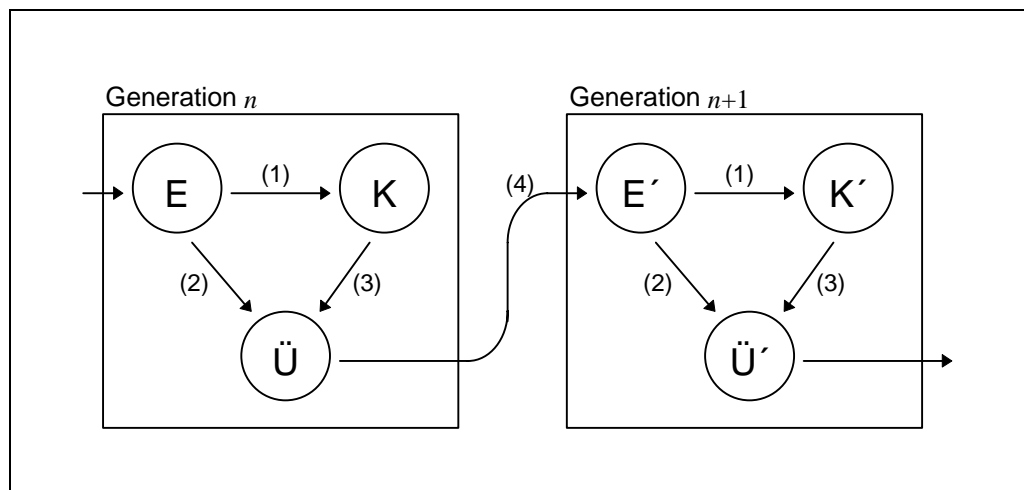


Abbildung 4: Populationsverhalten

Die Kante (1) repräsentiert die Reproduktion von Individuen. Nur an diesem Übergang werden neue Individuen erzeugt. Dies geschieht bei sexueller Fortpflanzung in vier Schritten:

- 1.) Selektion eines Elternpaars aus der Elternpopulation
- 2.) Rekombination der Elternchromosomen zu zwei neuen Chromosomen
- 3.) Mutation der Kindchromosomen
- 4.) Einfügen der beiden neuen Individuen in die Population der Kinder

Bei einer ungeschlechtlichen Fortpflanzungsstrategie wird nur ein Elter selektiert und Schritt 2.) entfällt. Die Auswahl der Eltern beruht bei den genetischen Algorithmen auf der Fitneß. Je „tüchtiger“ ein Individuum, desto höher ist seine Chance sich fortzupflanzen bzw. die Anzahl seiner Nachkommen. Dies ist der zentrale Punkt, an dem das Prinzip der Selektion

greift. In einer typischen Evolutionsstrategie dagegen besteht für jedes Individuum die gleiche Wahrscheinlichkeit, als Elter ausgewählt zu werden, unabhängig von seiner tatsächlichen Tauglichkeit. Das Prinzip der Selektion wird erst durch die Kante (4) verwirklicht. Beide Ansätze sind aber stochastisch. Ebenfalls stochastisch sind die Entscheidungen, ob eine Rekombination oder Mutation stattfindet. Dafür ist eine Rekombinations- und eine Mutationswahrscheinlichkeit definiert. Erstere bezieht sich auf das ganze Chromosom, letztere gibt die Wahrscheinlichkeit je Gen an.

Die Übergangspopulation  $\ddot{U}$  wird durch die Kanten (2) und (3) gefüllt. Durch (2) gelangen Eltern unverändert in die Übergangspopulation. Hier gibt es drei Varianten, welche Individuen übernommen werden: keine, alle, oder die  $n$  Besten. Die letzte Variante, der sogenannte Elitismus soll sicherstellen, daß das beste bisher gefundene Individuum nicht verlorenggeht, sondern in die nächste Generation hinübergerettet wird. Damit erreicht man eine monotone Konvergenz, d.h. im Laufe der Generationen kann sich die Lösung nicht verschlechtern. Die anderen beide Varianten treten auf, wenn reine GA bzw. reine ES in dieses Schema abgebildet werden. Durch (3) werden normalerweise alle zuvor erzeugten Kinder in die Übergangspopulation übernommen. Es wäre aber auch denkbar hier bereits eine stochastische oder deterministische Selektion anzuwenden.

Verbindung (4) wählt aus der Übergangspopulation die Individuen für die Elternpopulation der nächsten Generation. Die Auswahl kann sowohl deterministisch, d.h. wähle die  $n$  besten Individuen, als auch stochastisch (Fitneß = Auswahlwahrscheinlichkeit) geschehen. In einem Standard GA hat die Selektion schon in (1) stattgefunden, hier würde einfach die Übergangspopulation kopiert ( $E' := \ddot{U}$ ). ES fassen in der Übergangspopulation zunächst Eltern und Kinder zusammen, und wenden das Prinzip der Selektion erst bei der Erzeugung der neuen Elterngeneration an. Dabei wählen sie deterministisch die  $n$  besten. Da in  $\ddot{U}$  sowohl Eltern als auch Kinder enthalten sein können, wird der Selektionsdruck stärker, und die Wahrscheinlichkeit, daß schlechte Mutation schnell verschwinden, steigt.

Die Populationsgröße ist ein wichtiger Parameter für jeden Evolutionären Algorithmus, der sowohl die Konvergenzgeschwindigkeit als auch die Konvergenzsicherheit beeinflusst. Für eine zu kleine Population besteht die Gefahr, frühzeitig gegen ein lokales Optimum zu konvergieren, weil die genetische Vielfalt zu klein ist. Mit der Größe der Population steigt die erforderliche Rechenzeit. Die Anzahl der Individuen wird im Gegensatz zu natürlichen Populationen konstant gehalten, obwohl es auch Vorschläge für dynamisches Verhalten gibt (vgl. [Schöneburg2] 156ff). Genetische Algorithmen beziehen sich bei Größenangaben auf die Elternpopulation. Als Richtwert gilt eine Größe von  $40 \pm 20$ . Evolutionsstrategien geben die Größe der Eltern- und der Kinderpopulation getrennt an. Dies beruht auf der von Schwefel ([Schwefel] 139ff) eingeführten  $(\mu\#\lambda)$ -Notation, bei der  $\mu$  die Anzahl der Eltern und  $\lambda$  die der Kinder bezeichnet. Das # steht für einen Operator der angibt, ob, wie zu Kante (2) beschrieben, Eltern mit in die Übergangspopulation aufgenommen werden sollen (+) oder nicht (.). Hierbei wurden gute Ergebnisse mit  $\mu < \lambda$  und  $\mu + \lambda = 60 \pm 20$  erzielt.

### 2.3.6 Selbstadaption

In den vorangegangenen Kapiteln wurde klar, daß Evolutionäre Algorithmen durch verschiedene Strategieparameter kontrolliert werden, z.B. Mutationswahrscheinlichkeiten und Populationsgrößen. Es stellt sich nun die Frage, welche Einstellungen zu guter Konvergenz und Effizienz führen. Die Antwort ist nicht einfach, da die optimalen Parameterwerte von Problem zu Problem unterschiedlich sein können. Gute Parametereinstellungen werden daher häufig in Vorabversuchen experimentell ermittelt. Aber auch während der Optimumsuche, also zur Laufzeit des Programms, können sich die optimalen Einstellungen ändern. Steigt die Zielfunktion in einer Region des Parameterraums glatt an, sind andere Parameterwerte besser geeignet, als in einer stark zerklüfteten Region mit vielen Gipfeln und Tälern. Es ist in solchen Situationen nicht optimal, die Strategieparameter während des Suchprozesses konstant zu halten.

Ein Ausweg aus diesem Dilemma bietet die Erweiterung um Methoden, die die Strategieparameter selbständig an die jeweiligen Randbedingungen anpassen. Dies kann z.B. durch eine externe Heuristik geschehen, die für jede neue Generation diese Parameter neu einstellt. [Rechenberg] war der erste, der so vorgegangen ist. Seine Regel zur Anpassung der Mutationsschrittweite lautete: *Ist der Quotient der erfolgreichen Mutationen durch die nicht erfolgreichen größer als 1/5 wird die Streuung der Mutation (Schrittweite) erhöht, ansonsten verringert.* Der nächste Schritt war die Entwicklung eines Meta-Algorithmus, der die Parameter optimiert. Dies verlagert aber nur das Problem auf die Meta-Ebene, für die ja ebenfalls Parameter festgelegt werden müssen. Am interessantesten erscheint der Versuch, die Strategieparameter zusammen mit den Problemparametern im Individuum zu repräsentieren, und den Prinzipien der Evolution zu unterwerfen. Die Evolution optimiert sich dann quasi selbst. Dieses Verfahren wird auch von der Natur angewandt. Durch Reparatur- und Mutationsmechanismen (die ebenfalls in den Erbinformationen gespeichert sind) wird die Mutationsrate beeinflusst. Empirisch sind unterschiedliche Mutationsraten zwischen Lebewesen belegt, die nicht extern, sondern durch ihre Gene bedingt sind ([Schöneburg2] 80ff).

Die Selbstadaption ist Bestandteil jeder neueren Evolutionsstrategie, für Genetische Algorithmen jedoch nur wenig erforscht. Ermutigt durch die guten Ergebnisse, die mit ersten Experimenten erzielt wurden, empfiehlt [Hoffmeister] das Verfahren der Selbstadaption in zukünftigen Anwendungen Genetischer Algorithmen stärker zu berücksichtigen. Die selbständige Anpassung an problemspezifische Besonderheiten (Struktur der Zielfunktion) erhöht die Robustheit des Verfahrens, da nicht mehr manuell nach der richtigen Parametereinstellung gesucht werden muß.

## 2.4 Einordnung der Evolutionären Algorithmen

Bei der Evolution handelt es sich nicht um ein exaktes Verfahren, durch die Verwendung von Zufallsprozessen ist der Beweis der Optimalität einer gefundenen Lösung nicht möglich. Ebenso ist die genaue Funktionsweise

verschiedener Prinzipien, z.B. Kodierung der Gene oder Rekombinationsstrategien, noch Gegenstand der Forschung. Der Einsatz Evolutionärer Algorithmen ist somit eine heuristische Vorgehensweise. Das Besondere ist der deskriptive Ansatz: ein Optimierungsziel wird beschrieben, der Algorithmus generiert Lösungen, die diesem Ziel möglichst nahe kommen. Anstatt für jedes spezielle Problem eine Menge von Regeln vorzugeben, mit der eine gute Lösung erzielt wird, kann ein und derselbe Algorithmus für grundverschiedene Optimierungsziele eingesetzt werden.

Von der reinen Zufallssuche unterscheiden sich die Evolutionären Algorithmen dadurch, daß Informationen über bereits untersuchte Punkte des Parameterraums verwertet werden. Die Individuen sind nicht das Ergebnis unabhängiger Zufallsexperimente, die „guten“ Eigenschaften werden von einer Generation in die nächste vererbt, und nur partiellen geändert.

Die Optimierung mit Hilfe der Evolution hat gegenüber den beschriebenen konventionellen Verfahren folgende Vorteile:

- *Robustheit*: Alle Optimierungsprobleme können prinzipiell mit demselben Algorithmus gelöst werden. Die Anpassungen für die Problemdarstellung und Bewertung sind relativ leicht durchzuführen.
- *Skalierbarkeit des Aufwands*: Die Güte der besten gefundenen Lösung steigt von Generation zu Generation. Der Aufwand wird an der Anzahl der untersuchten Generationen gemessen. Die Evolution ist ein endloser Prozeß, es ist kein Abbruchkriterium definiert. Die Simulation kann daher nach einer beliebigen Anzahl von Generationen unterbrochen werden. Je mehr Generationen simuliert wurden, also je höher der investierte Aufwand war, desto besser wird das Ergebnis ausfallen. Der Anwender muß nur solange Rechenzeit einsetzen, bis er ein Ergebnis erhält, das seinen Ansprüchen genügt.
- *Parallelität*: Die Erzeugung und Bewertung der Individuen ist unabhängig voneinander, und somit parallelisierbar. Damit sind Evolutionäre Algorithmen für eine Implementierung auf zukünftigen, parallelen Rechnerarchitekturen besonders gut geeignet. Für die konventionellen

sequentiell arbeitenden Verfahren ist eine Parallelisierung mit großen Schwierigkeiten verbunden.

Nachdem nun der Einsatz Evolutionären Algorithmen zur Optimierung auf allgemeiner Basis erläutert wurde, setzt das nächste Kapitel die gewonnenen Erkenntnisse zur Lösung eines konkreten, praxisnahen Problems ein: der Optimierung der Maschinenbelegung in einem Druckgußwerk.

## 3 Optimieren der Druckgußmaschinenbelegung

### 3.1 Problemstellung

Diese Arbeit befaßt sich schwerpunktmäßig mit der Entwicklung eines auf evolutionären Algorithmen basierenden Verfahrens zur Optimierung der Maschinenbelegungsplanung in einem Druckgußwerk. Dazu ist das Verständnis der konkreten Planungssituation im Unternehmen notwendig. Diese ergibt sich aus der Unternehmenssituation, der Produktionsstruktur und den mit der Optimierung verfolgten Zielen. Die folgenden Kapitel erläutern diese Aspekte.

#### 3.1.1 Unternehmenssituation

Der Situation des Unternehmens wird anhand von sechs Merkmalen geschildert, die einem bei [Glaser] beschriebenen Typisierungsschema entnommen wurden. Das von Glaser verfolgte Ziel der Typisierung bestand darin, die maßgebenden Kriterien herauszufinden, die die Auswahl von PPS-System für mittelständische Produktionsbetriebe bestimmen. Da der hier vorgestellte Algorithmus einen kleinen Teilaspekt eines Produktionsplanungs- und Steuerungssystems abdeckt, erscheine eine Beschreibung nach diesem Schema sinnvoll. Die sechs Merkmale haben im betrachteten Unternehmen folgende Ausprägungen:

- **Produktstruktur:** Einteilige Fertigerzeugnisse. Ein Gußteil erfährt nach dem Gießen nur noch eine geringfügige Nachbearbeitung, z.B. Entgraten und Reinigen. Montageprozesse finden nicht statt. Die Produktpalette umfaßt einige hundert Gußteile.
- **Produkttypisierungsgrad:** Kundenindividuelle Produkte. Die Gußteile sind in der Regel kundenspezifisch. Sie sind so speziell, daß die zur Produktion benötigten Werkzeuge teilweise Eigentum des Kunden sind.
- **Betriebsauftragsauslöseart:** Der Primärbedarf entsteht durch Kundenaufträge auf der Basis von Rahmenverträgen. Ein Rahmenvertrag

wird über einen längeren Zeitraum abgeschlossen, z.B. 1 Jahr. Wöchentlich oder monatlich übermittelt der Kunde seinen Bedarf für die nächsten Perioden in Form von Abrufen. Diese Bedarfe müssen gedeckt werden, eine Ablehnung von Abrufen ist nicht möglich.

- **Fertigungsauftragsgröße:** Serienfertigung. Die Dauer eines Auftrags liegt zwischen einem Tag und einigen Wochen. Aufgrund der langfristigen Verträge wird eine Serie wiederholt aufgelegt.
- **Organisationsform der Fertigung:** Werkstattfertigung. Die Maschinen für die unterschiedlichen Arbeitsgänge sind räumlich zusammengefaßt. In einigen Maschinen können auch mehrere Arbeitsgänge integriert sein, z.B. Gießen und Stanzen.
- **Produktionstiefe:** Mehrstufig. Hier wird jedoch nur eine Stufe betrachtet, die Gießerei. Der Fertigungsprozeß erfordert mehrere Arbeitsgänge, diese sind jedoch alle dem Gießen untergeordnet. Die Gießerei ist die erste und wichtigste Stufe im Produktionsprozeß, die alle nachfolgenden Stufen dominiert. Nur für die Gießerei wird ein detaillierter Plan aufgestellt.

### 3.1.2 Fertigungssituation in der Gießerei

Der Produktionsprozeß in der Gießerei des betrachteten Unternehmens läßt sich vereinfacht wie folgt charakterisieren: um Gußteile zu produzieren, wird in eine Druckgußmaschine eine Form für das gewünschte Teil eingebaut. Eine Form besteht aus zwei Hälften, der Ofen- und der Auswerferseite. Die Maschine ist mit einem Vorratsbehälter für flüssiges Metall verbunden (Ofen). Sie preßt das flüssige Metall mit hohem Druck durch Öffnungen an der Ofenseite in die Form. Nachdem das Metall erstarrt ist, wird die Form geöffnet und die Gußteile entnommen. Anschließend wird die Form automatisch gereinigt und für den nächsten Produktionsvorgang vorbereitet. Ein kompletter Zyklus wird Schuß genannt, ein Schuß ist daher die grundlegende Maßeinheit der Planung.

Eine Form kombiniert einen Rahmen mit einem oder mehreren Einsätzen, sie wird deshalb auch als Druckgußwerkzeugkombination bezeichnet. In den

Einsätzen sind Aussparungen vorgesehen, aus denen die Gußteile entstehen. Sie werden Nester genannt. Ein Einsatz kann mehrere Nester enthalten, die normalerweise alle das gleiche Teil produzieren. Ausnahmen bestätigen die Regel. Für den Einbau in eine Maschine ist ein Rahmen notwendig. Jeder Einsatz kann nur in genau einen Rahmen eingebaut werden, in einen Rahmen passen mehrere, unterschiedliche Einsätze. Ein Gußteil kann durch die Nummer des Rahmens und des verwendeten Einsatzes eindeutig identifiziert werden. Im betrachteten Unternehmen gibt es demzufolge keine Unterscheidung zwischen der Bezeichnung der Werkzeugkombination und des damit produzierten Gußteils, ein Indiz für die Dominanz der Gießerei-Stufe im Produktionsablauf.

In der Gießerei sind Druckgußmaschinen unterschiedlichen Typs vorhanden. Sie unterscheiden sich durch ihre technischen Eigenschaften. Dazu gehören insbesondere die Produktionsgeschwindigkeit (Zeit je Schuß), die verarbeitbaren Werkstoffe (Legierungen) und die maximale Kraft, mit der das flüssige Metall in die Form gedrückt werden kann. Die Eigenschaften der Maschine bestimmen, welche Werkzeugkombination in welche Maschine eingebaut werden kann. Eine Werkzeugkombinationen läßt sich üblicherweise in mehrere Maschinen einbauen, aber nicht unbedingt in alle.

Für jede Form läßt sich somit eine Menge zulässiger Maschinen ( $ZM$ ) bestimmen. Zwischen diese Mengen können Überlappungen auftreten, d.h. sie sind nicht disjunkt. Ein Beispiel:  $a, b, c$  seien Werkzeugkombinationen und  $m_1, m_2, m_3$  die zur Verfügung stehenden Maschinen. Dann sind folgende Aussagen über die Mengen zulässiger Maschinen möglich:

$$ZM(a) = \{m_1, m_2\}, \quad ZM(b) = \{m_2\}, \quad ZM(c) = \{m_2, m_3\}$$

$$ZM(a) \cap ZM(b) \cap ZM(c) \neq \emptyset$$

Aufgrund der Differenziertheit der Maschinen, und der komplexen Zuordnungsbeziehung zwischen Werkzeugkombination und zulässiger Maschine, lassen sich die Produktionskapazitäten der einzelnen Maschinen nicht aggregieren. Die Betrachtung der gesamten Gießerei als eine einzelne Maschine ist daher nicht sinnvoll, ein Mehr-Maschinen-Problem muß gelöst werden.

Bei einem Wechsel des von einer Maschine zu produzierenden Gußteils muß der Fertigungsprozeß unterbrochen werden, um die verwendete Form auszu-tauschen und eventuell auf eine andere Legierung umzustellen. In den dadurch entstehenden Rüstzeiten steht die Maschine still. Die Rüstzeiten sind prinzi-piell von der Reihenfolge der produzierten Gußteile abhängig. Wie bereits erwähnt, können in einen Rahmen unterschiedliche Einsätze eingebaut werden. Werden zwei Gußteile nacheinander mit demselben Rahmen produziert, muß die Werkzeugkombination nicht komplett ausgetauscht werden. Der Rahmen bleibt auf der Maschine, nur die Einsätze werden ausgetauscht. Würde sich auch der Rahmen der benötigten Formen unterscheiden, müßte die gesamte Form aus- und eine neue eingebaut werden. Dieser Vorgang benötigt ungefähr die doppelte Zeit wie der ausschließliche Tausch der Einsätze.

Ein Wechsel der verwendeten Legierung zieht Reinigungsvorgänge der Maschine nach sich, um Verunreinigungen zu vermeiden. Diese Rüstzeit bezeichnet man als Werkstoffwechselzeit. Sie tritt nur dann auf, wenn zeitlich aufeinanderfolgend Teile gegossen werden, die aus unterschiedlichen Legierungen bestehen.

### **3.1.3 Optimierungsziele**

Der Optimierung der Maschinenbelegung in der Gießerei dient dem Ziel der Kostenminimierung. Weitere, aus der Unternehmensphilosophie ableitbare Ziele, wie z.B. Qualitätsoptimierung werden nicht berücksichtigt. Eine Kostenfunktionen, die für eine gegebene Maschinenbelegung die relevanten Kosten ermittelt, könnte unmittelbar in den zu entwickelnden Evolutionären Algorithmus umgesetzt werden. Sie entspricht direkt der Bewertungsfunktion. Leider ist die Ermittlung der Kostenfunktionen eine schwierige Aufgabe, denn nicht alle Auswirkungen der Planung lassen sich mit Kosten bewerten. Welche Kosten verursachen beispielsweise verspätet ausgelieferte Kundenaufträge? Daher orientieren sich Optimierungsverfahren für Maschinenbelegungspläne häufig an Zeit- statt an Kostenkriterien. Diese Vorgehensweise wird auch hier angewandt. Die folgenden Zeitkriterien sollen minimiert werden:

- **Verspätungen:** Damit ist die positive Differenz zwischen der Fälligkeit eines Bedarfs, und der tatsächlichen Bereitstellung der geforderten Menge gemeint, d.h.  $Verspätung = Bereitstellungsstermin - Fälligkeit$ . Eine Verspätung kann erhebliche Konventionalstrafen nach sich ziehen. Allgemein verursachen Verspätungen Fehlmengenkosten, da der in einem Zeitpunkt vorhandene Bedarf nicht gedeckt werden kann.
- **Verfrühungen:** Das Gegenstück zu den Verspätungen, ein Auftrag wird früher als zur Einhaltung der Fälligkeit nötig begonnen:  $Verfrühung = Fälligkeit - Fertigstellungstermin$ . Beginnt die Produktion früher als nötig, muß das benötigte Material früher bereitgestellt werden, die entstehenden Gußteile müssen gelagert werden. Dadurch entstehen Kapitalbindungskosten (für die verfrühte Bereitstellung des Ausgangsmaterials) und Lagerhaltungskosten (für die Gußteile).
- **Rüstzeiten:** Dies sind alle Zeiten, in denen die Maschine aufgrund von Rüstvorgängen (Form- bzw. Werkstoffwechsel) nicht produzieren kann. Das Rüsten an sich verursacht Rüstkosten, außerdem liegt die Kapazität der Maschine während des Rüstens brach, so daß keine Deckungsbeiträge erwirtschaftet werden können. Dies macht sich vor allem in Engpaßsituationen negativ bemerkbar.
- **Arbeitszeiten:** Die je Schuß durchschnittlich benötigte Zeit hängt von der benutzten Werkzeugkombination und der eingesetzten Maschine ab. Die Werkzeugkombination ist durch das zu produzierende Gußteil zwingend vorgeschrieben. Somit bleibt nur die Wahl zwischen unterschiedlich schnellen Maschinen. Die Minimierung der Arbeitszeit verringert einerseits Kosten (Personalkosten der Bediener), andererseits wird zusätzliche Maschinenzeit frei. Die **Durchlaufzeit** wird hier aufgrund der einstufigen Fertigung nicht explizit minimiert. Sie ist in der Gießereistufe gleichbedeutend mit der Summe der Rüst- und Bearbeitungszeiten, über Wartezeiten zwischen den Stufen sind keine Informationen bekannt

Die Minimierung jedes der obigen Kriterien trägt natürlich unterschiedlich stark zur Kostenminimierung bei. Zusätzlich können die sich die Teilziele sowohl ergänzen als auch miteinander konkurrieren. Beispielsweise kann die

Einplanung eines Auftrags auf einer langsamen Maschine notwendig werden, weil alle schnellen bereits belegt sind, und er nur so pünktlich fertiggestellt werden kann. Die Minimierung der Verspätung hat dann Vorrang vor der Minimierung der Arbeitszeit. Andererseits kann durch die Einplanung auf einer schnellen Maschine die zusätzliche Kapazität frei werden, die benötigt wird, um einen anderen Auftrag rechtzeitig abzuschließen. In diesem Fall sind die Ziele komplementär.

Das Gesamtziel läßt daher nur bestimmten, wenn alle Teilziele simultan optimiert werden. Dies geschieht durch die Bewertungsfunktion des Algorithmus, in der die Teilziele miteinander verknüpft werden. Dazu wird in dieser Arbeit das Prinzip einer einfachen Nutzwertanalyse angewandt ([Lehner] 246ff). Dieses geht davon aus, daß der Erfüllungsgrad jedes Teilziels quantitativ (Punktzahl) ermittelt werden kann. Die Summe der gewichteten Erfüllungsgrade ergibt die Gesamtbewertung. Die für diesen Fall notwendigen Modifikationen sind in Kapitel 3.3.2: *Bewerten der Maschinenbelegung* erläutert.

#### **3.1.4 Das Planungsproblem**

Das Planungsproblem besteht nun darin, festzulegen, welche Druckgußmaschinen zu welchem Zeitpunkt welche Gußteile fertigen sollen. Dabei sollen die oben beschriebenen Zeitziele minimiert werden. Die an die Gießerei übermittelten Bedarfe müssen erfüllt und die technischen Restriktionen beachtet werden. Zur Vereinfachung sei im folgenden angenommen, daß es sich bei den Bedarfen um Nettobedarfe für Gußteile handelt, d.h. der Lagerbestand und eventueller Schwund in Folgestufen wurde bereits einkalkuliert.

### **3.2 Lösungsansatz**

In der Praxis wird die Maschinenbelegung häufig sequentiell in zwei Schritten geplant: zuerst werden gleichartige Bedarfe zu Fertigungslosen zusammengefaßt. Anschließend werden diese Lose Maschinen zugeteilt, auf denen sie produziert werden sollen. Die Losbildung hat das Ziel, einen Ausgleich

zwischen den je Los konstanten Rüstkosten und den mengenabhängigen Lagerhaltungskosten zu schaffen. Diese entstehen, wenn Bedarfe vor ihrem Fälligkeitstermin produziert werden, und dadurch fertige Artikel zwischengelagert werden müssen. Von der Minimierung der Lagerhaltungskosten geht die Tendenz zu kleinen Losen, die Minimierung der Rüstkosten verlangt nach großen Losen. Lose, die Maschinen zugeordnet sind, werden häufig auch als Fertigungsaufträge bezeichnet, die Maschinen belegen. Im zweiten Schritt wird die Bearbeitungsreihenfolge der Lose auf den belegten Maschinen festgelegt. Damit wird das Ziel verfolgt, reihenfolgeabhängige Kosten (Umrüsten) unter Einhaltung der terminlichen Nebenbedingungen zu minimieren. Da hier eine einstufige Fertigung betrachtet wird (oder sich alle Folgestufen an der ersten orientieren), brauchen keine Interdependenzen zwischen Fertigungsaufträgen auf unterschiedlichen Stufen berücksichtigt werden (z.B. daß ein Auftrag  $x_n$  auf Stufe  $n$  erst dann beginnen kann, wenn der Auftrag  $x_{n-1}$  auf der vorhergehenden Stufe beendet ist).

Das Dilemma dieser zweistufigen Planung besteht darin, daß zum Zeitpunkt der Losbildung Aspekte der Reihenfolgeplanung nicht berücksichtigt werden können. Die Bearbeitungsreihenfolge der Lose kann ja erst nach der Losbildung bestimmt werden. Wenn beide Schritte unabhängig voneinander durchgeführt werden, ist nicht sichergestellt, daß das Kostenminimum erreicht wird, trotz vielleicht optimaler Lösung der Teilprobleme. Der für die Stufe als Ganzes optimale Produktionsplan kann nur durch eine simultane Los- und Reihenfolgeplanung ermittelt werden. Solche Simultanmodelle lassen sich mit exakten Verfahren für viele praxisrelevante Problemfälle und Datenmengen aber nicht mehr in angemessener Zeit lösen.

Mit den Evolutionären Algorithmen existiert ein Verfahren, welches mächtig genug ist, die beschriebene Komplexität des Planungsproblems abzubilden. Trotzdem können effizient Lösungen gefunden werden. Es wird gezeigt, daß durch eine geeignete Problemkodierung Lose und Reihenfolgen simultan bestimmt werden können. Der simultane Ansatz verspricht bessere Ergebnisse als sequentiell arbeitende Heuristiken. Gleichzeitig können die den Evolutionären Algorithmen innewohnenden Vorteile genutzt werden, insbesondere

Robustheit gegenüber sich ändernden Zielen und eine vereinfachte Implementierung. Statt Algorithmen für die Losbildung, die Ermittlung der Reihenfolge und die Auswahl geeigneter Maschinen zu programmieren, genügt die Implementierung eines einzigen Algorithmus. Auch die Kommunikation zwischen den Teilprogrammen entfällt.

Der Evolutionäre Algorithmus soll wie in Abbildung 5 eingesetzt werden: in einem Schritt: aus den eingehenden Bedarfen wird unter Berücksichtigung der Nebenbedingungen direkt ein Maschinenbelegungsplan erzeugt. In den folgenden Kapiteln wird beschrieben, auf welche Art und Weise die Evolution simuliert werden soll, um das gewünschte Ziel zu erreichen.

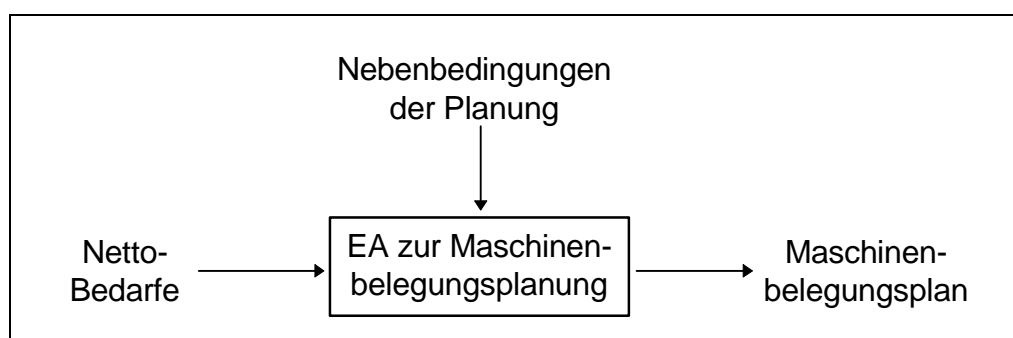


Abbildung 5: Der Evolutionäre Algorithmus als „Black Box“

### 3.2.1 Ansätze in der Literatur der Evolutionären Algorithmen

Eine Übersicht über die in der Literatur beschriebenen Anwendungen Evolutionärer Algorithmen gibt [Bäck1]. Die Anwendungen sind thematisch sortiert und teilweise kurz kommentiert. Von den im Abschnitt „Production Planning“ genannten Quellen lassen sich ungefähr die Hälfte dem Problem des „Scheduling“ zuordnen. Darunter versteht man das Problem der Festlegung der Bearbeitungsreihenfolge von  $n$  Aufträgen auf  $m$  Ressourcen. Ein Auftrag muß dabei unter Umständen nacheinander auf mehreren Ressourcen bearbeitet werden. Auch die Maschinenbelegungsplanung ist ein Scheduling-Problem. Die Konzentration auf das Scheduling, also die Reihenfolgeplanung, bedeutet jedoch bereits eine drastische Einschränkung: die Planung setzt voraus, daß bereits Aufträge existieren, sie setzt also erst *nach* der Losbildung an.

Alle bei [Bäck1] aufgelisteten Quellen implementieren Genetische Algorithmen. Sie stellen das Problem entweder auf der Basis des Travelling-Sales-Person-Problems (siehe Kapitel 2.3.3) oder stellen völlig neue Ansätze dar: [Hilliard] beschreibt beispielsweise einen Algorithmus, der Scheduling-Heuristiken generiert, die dann die eigentliche Planung übernehmen. Andere Ansätze beschäftigen sich mehr mit theoretischen Aspekten und untersuchen z.B. neue, problemspezifische Crossover-Operatoren (z.B. [Lipins]). Diese werden an sehr einfachen Beispielen getestet (keine Rüstzeiten, Verspätungen oder Verfrühungen, einziges Ziel ist die Minimierung der Durchlaufzeit). Da der Einbau problemspezifischen Wissens jedoch zu Lasten der Robustheit geschieht, kann man keine Aussage darüber treffen, ob die guten Ergebnisse dieser Verfahren auch auf andere Zielsetzungen übertragbar sind.

Ein simultaner Ansatz, der auch in der Praxis verwirklicht wurde, findet sich bei [Zimmerman]. Er verwendet eine Evolutionsstrategie zur simultanen Programm-, Losgrößen- und Reihenfolgeplanung. Da von ihm betrachtete Unternehmen produziert für einen anonymen Markt. Zur Produktionsplanung gehört damit auch die Entscheidung über das Produktionsprogramm, d.h. welche Artikel in welcher Menge gefertigt werden sollen. Dies ist leider ein Grund dafür, warum sich sein Verfahren nicht auf das hier geschilderte Problem übertragen läßt: die Entscheidung darüber, welche Teile produziert werden sollen, ist bereits gefallen.

Auch [Schöneburg1] beschreibt eine Anwendung von Evolutionsstrategien, allerdings nur zur Reihenfolgeplanung. Die Losbildung geschieht manuell oder heuristisch. Die Fertigungssituation (Fließproduktion mit drei Bändern) unterscheidet sich deutlich von der des hier betrachteten Unternehmens.

Das hier zu lösende Problem wird von allen Ansätzen nur teilweise abgedeckt. Daher wurde ein eigener Lösungsansatz entwickelt, der Aspekte der in der Literatur beschriebenen Verfahren verwendet, diese aber in einen neuen Zusammenhang setzt und erweitert, um die unternehmensspezifischen Probleme berücksichtigen zu können.

### 3.2.2 Realisierter Ansatz

Um die Prinzipien der Evolution effizient nutzen zu können, muß eine geeignete Transformation des Problems in die Objekte der Evolution gefunden werden. Das Gießereiproblem wird wie folgt durch Individuen, Chromosomen und Gene repräsentiert.

Jedes Individuum beschreibt einen möglichen Maschinenbelegungsplan. Die Maschinen werden aber nicht mit Losen, sondern mit den an die Gießerei gemeldeten Bedarfen belegt. Ein Fertigungsauftrag umfaßt immer nur einen einzigen Bedarf. Bearbeitet eine Maschine nacheinander mehrere Bedarfe für identische Teile, so wird diese Bedarfsgruppe als Los interpretiert. Die Losgröße läßt sich somit nur als Summe ganzer Bedarfe darstellen. Das gleiche Prinzip benutzt auch [Zimmermann], um Lose und Reihenfolge simultan zu bestimmen.

Unter den für das dynamische Losgrößen-Modell von Wagner/Whitin formulierten Nebenbedingungen läßt sich sogar zeigen, daß diese Vorgehensweise Bestandteil jeder optimalen Politik ist ([Kistner] S. 49ff). Auf die Realität läßt sich diese Aussage natürlich nicht immer übertragen, da auch die Annahmen, wie z.B. unbeschränkte Kapazität, nicht erfüllt sind: ein Bedarf kann so groß sein, daß er aus Kapazitätsgründen zwei Maschinen gleichzeitig belegen muß, um ohne Verspätung fertig zu werden. Ein Bedarf wird dann auf zwei Fertigungsaufträge verteilt. Dieser Fall stellt aber keine prinzipielle Einschränkung der vorgeschlagenen Lösung dar, wenn eine Obergrenze für die Bedarfsmenge eingeführt wird. Liegt ein Bedarf über dieser Grenze, wird er solange geteilt, bis die Obergrenze unterschritten wird. Im Extremfall könnte die Bedarfsobergrenze bei einem einzelnen Teil liegen (was aus Effizienzgründen aber nicht zu empfehlen ist). Diese Aufbereitung der Daten findet jedoch außerhalb des Evolutionären Algorithmus statt.

Die Erbinformation des Individuums speichert die Zuordnung der Bedarfe zu Maschinen und ihre Bearbeitungsreihenfolge. Jedes Gen eines Chromosoms repräsentiert einen Bedarf und enthält Informationen über die belegte Maschine. Der Aufbau des Chromosoms orientiert sich an dem von der

Literatur für Reihenfolgeprobleme vorgeschlagenen Ansatz. Zur Bewertung eines Plans müssen die genauen Anfangs- und Endtermine der Fertigungsaufträge bekannt sein, die sich aus den gespeicherten Parametern nicht unmittelbar ablesen lassen. Vor der Bewertung liegt daher eine Dekodierphase, in der aus den gespeicherten Parametern ein konkreter Produktionsplan wird. In diesem Schritt werden die Rüst- und Bearbeitungszeiten in Abhängigkeit von der Reihenfolge, den verwendeten Maschinen und Materialien, sowie den Bedarfsmengen berechnet. Nachdem die Dauer der Aufträge bekannt ist, können sie terminiert werden.

Die Mutations- und Rekombinationsoperatoren manipulieren über die Erbinformation die Parameter der Maschinenbelegung. Dazu werden problemunabhängige Verfahren benutzt, die bereits bei Travelling-Sales-Person-Problemen ihre Qualität unter Beweis gestellt haben ([Oliver]). Die durch das Individuum repräsentierte Maschinenbelegung wird gemäß dem Zielsystem bewertet. Die Qualität des Plans bestimmt die Tauglichkeit des Individuums, welche die Grundlage der Selektion ist. Ausgehend von einer Population zufällig erzeugter Individuen (Pläne) kommt so der Prozeß der Evolution in Gang. Die Realisierung dieses Ansatzes wird in Kapitel 3.3: *Implementierung* beschrieben.

### **3.3 Implementierung**

#### **3.3.1 Repräsentation der Maschinenbelegungsparameter**

Die Maschinenbelegung in der Gießerei legt fest, welche Produkte in welcher Reihenfolge zu welchen Zeitpunkten auf welchen Maschine produziert werden sollen. Die einzelnen Maschinen arbeiten parallel und unabhängig voneinander, so daß je Maschine die zugeordneten Aufträge und die Auftragsfolge lokal betrachtet werden kann. Die Anfangs- und Endtermine der Aufträge lassen sich deterministisch aus einer vorgegebenen Bearbeitungsreihenfolge und den technischen Nebenbedingungen (z.B. Bearbeitungszeiten) ermitteln, wenn dazu Regeln existieren, wie z.B. „Ordne

alle Aufträge lückenlos hintereinander an“. Für die Repräsentation der Maschinenbelegung genügen somit die folgenden drei Parameterarten:

1. Zuordnung von Aufträgen zu Maschinen
2. Auftragsreihenfolge je Maschine
3. Regeln zur Umsetzung der Auftragsreihenfolge in Start- und Endtermine

Die Regeln zur Interpretation der Reihenfolge sollen nicht Gegenstand der Optimierung sein und werden als gegeben und konstant vorausgesetzt. Sie gehen als Teil des problemspezifischen Wissens in den Dekodierungsschritt der Bewertungsfunktion ein. Die Erbinformation muß nur die ersten beiden Parameterarten speichern.

Für die Kodierung der Auftragsreihenfolge bietet es sich an, einen Ansatz zu wählen, wie er in der Literatur der Genetischen Algorithmen für das Travelling-Sales-Person-Problem vorgeschlagen wird ([Goldberg] oder [Schöneburg2]). In diesem Fall repräsentiert jedes Gen genau eine Stadt. Das Chromosom ist ein Array fester Länge von Genen. Die Gene sind eindeutig identifizierbar. Die Reihenfolge der Gene im Chromosom bestimmt die Reihenfolge, in der die Städte besucht werden.. Die Evolution darf nur die Anordnung der Gene im Chromosom ändern, d.h. sowohl die Anzahl als auch die Struktur der Gene bleibt konstant.

Für die Übertragung auf das hier betrachtete Maschinenbelegungsproblem gelten folgende Entsprechungen: Die Funktion (siehe Kapitel 2.3.1) eines Gens ist die Repräsentation eines Bedarfs. Das Chromosom ist ein Array fester Länge von Genen. Die Auftragsreihenfolge ist identisch mit der Anordnung der Gene im Chromosom, d.h. der erste Auftrag wird durch das Gen mit dem Index Null bestimmt, der zweite durch das Gen mit dem Index Eins, usw. Wie bereits erwähnt, umfaßt jeder Auftrag nur einen einzigen Bedarf, damit Reihenfolge und Lose (aufeinanderfolgende Bedarfe für identische Artikel) in einem Schritt manipuliert werden können.

Die Bedarfe sind in einer Tabelle angeordnet und durchnummeriert. Über diese Nummer kann jeder Bedarf eindeutig identifiziert und wiedergefunden werden. Sie übernimmt die Funktion eines Primärschlüssels im Sinne einer relationalen

Datenbank. Im Gen ist die Bedarfsnummer gespeichert, so daß die Bewertungsfunktion Zugriff auf die Attribute des Bedarfssatzes hat, z.B. Menge und Artikeltyp. Zwischen Gen und Bedarf existiert eine 1:1-Beziehung. Daraus folgt, daß jedes Gen einzigartig ist, und jedes Individuum genau so viele Gene enthält, wie Bedarfe existieren. Während der simulierten Evolution dürfen keine Gene hinzukommen, verdoppelt oder entfernt werden, da dies die Konsistenz der resultierenden Maschinenbelegung zerstören würde. Eine ähnliche Beziehung realisiert auch [Schöneburg1].

Wie wird nun die Zuordnung der Bedarfe zu Maschinen kodiert? Eine Möglichkeit ist es, für jede Maschine ein eigenes Chromosom anzulegen. Dies bildet zwar den Sachverhalt korrekt ab, da Zuordnungen und Reihenfolgen je Maschine dargestellt werden können. Der überwiegende Teil der Veröffentlichungen zu Evolutionären Algorithmen befaßt sich aber mit Ansätzen, die auf einem einzigen Chromosom basieren. Um die dort gemachten Erfahrungen mit den verschiedensten Mutations- und Rekombinationsstrategien verwenden zu können, müssen alle Parameter in einem Chromosom kodiert werden. Der Wert eines Gens (siehe Kapitel 2.3.1) wird bisher noch nicht verwendet. Dieser soll nun zur Bezeichnung der belegten Maschine dienen. Wie die Bedarfe sind auch die Maschinen numeriert. Auch hier übernimmt die Maschinenummer die Funktion eines Primärschlüssels. Der Wert eines Gens entspricht der Nummer der Maschine, die den repräsentierten Bedarf produzieren soll. Während sich die Funktion eines Gens (Bedarfsrepräsentation) nicht ändern darf, sind Mutationen des Werts zulässig und notwendig. Nur durch seine Veränderung können Alternativen erzeugt werden, in denen die Bedarfe von unterschiedlichen Maschinen produziert werden.

Der Wertebereich des Gens wird durch den zugehörigen Bedarf eingeschränkt: jeder Bedarf bezieht sich auf ein Gußteil, dieses bestimmt die Werkzeugkombination (die Form), die zur Produktion verwendet werden muß. Die Werkzeugkombination wiederum determiniert die Maschinen, in die sie eingebaut werden kann. Die Nummern dieser zulässigen Maschinen bilden den Wertebereich des Gens.

Um die Bearbeitungsfolge der Bedarfe einer bestimmten Maschine zu ermitteln, werden alle nicht zu dieser Maschine gehörenden Gene ausgeblendet. Die verbleibenden Gene sind identisch mit dem maschinenspezifischen Chromosom der zuerst angedachten Lösungsmöglichkeit. Die Reihenfolge der Gene im Restchromosom repräsentiert dann die gesuchte Bearbeitungsfolge.

Einige Anordnungen von Genen lassen sich sofort als suboptimal erkennen. Dies ist z.B. der Fall, wenn mehrere Bedarfe für ein Gußteil die gleiche Maschine belegen, sie aber nicht in der Reihenfolge ihrer Fälligkeitstermine bearbeitet werden. Angenommen, zwei Bedarfe  $b_1$  und  $b_2$  für dasselbe Teil haben die Fälligkeiten  $t_1$  und  $t_2$ , wobei gilt  $t_1 < t_2$ . Wird  $b_2$  vor  $b_1$  auf derselben Maschine produziert, so kann diese Reihenfolge in Bezug auf das Ziel, Verspätungen und Verfrühungen zu minimieren, nicht optimal sein. Dieses Problem kann dadurch behoben werden, daß die entsprechenden Gene im Chromosom vertauscht werden. Da die getauschten Gene identische Gußteile repräsentieren, hat diese Operation keinen Einfluß auf die Bearbeitungsreihenfolge der produzierten Gußteile, selbst wenn zwischen  $b_1$  und  $b_2$  andere Teile mit anderen Werkzeugen oder Legierungen produziert würden. Die Rüst- und Bearbeitungszeiten ändern sich durch diese Operation folglich nicht.

Ein Chromosom, in dem solche Anordnungen nicht auftreten, soll als *normalisiert* bezeichnet werden. In der Implementierung überwacht die Normalisierungsfunktion das Chromosom und ordnet, falls nötig, die Gene neu an. Dadurch braucht das Verfahren weniger Schritte bei der Suche nach dem Optimum, da offensichtlich suboptimale Individuen in tauglichere umgewandelt werden. Die Normalisierung benutzt dabei Wissen über die Optimierungsziele, stellt also eine problemspezifische Ergänzung dar, die im Standardalgorithmus nicht vorgesehen ist.

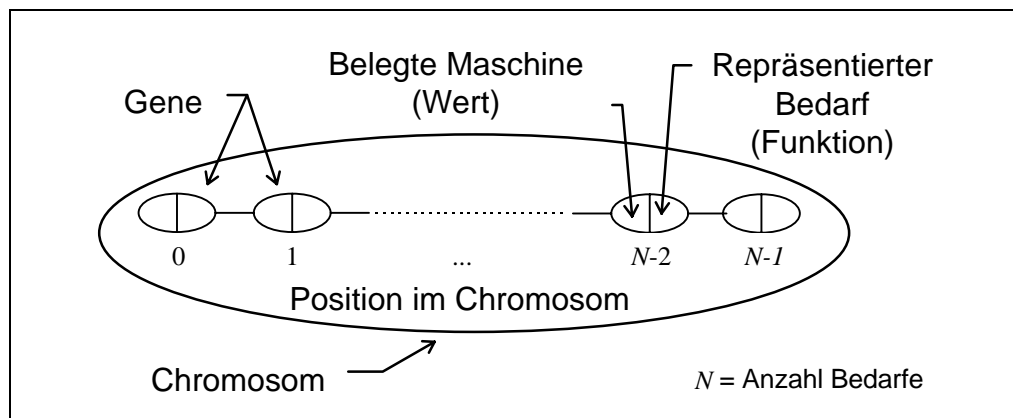


Abbildung 6: Aufbau des Chromosoms

Zusammenfassend kodiert das Chromosom die Parameter der Maschinenbelegung wie folgt (Abbildung 6): jedes Gen hat die Aufgabe, einen Bedarf zu repräsentieren. Der Wert des Gens entspricht der Nummer der Maschine, die diesen Bedarf produzieren soll. Der Wertebereich ist durch das zur Produktion benötigte Werkzeug eingeschränkt. Das Chromosom ordnet die Gene in einer sequentiellen Folge an. Es enthält so viele Gene, wie Bedarfe existieren. Durch die Position der Gene im Chromosom wird die Bearbeitungsreihenfolge abgebildet. Durch die Normalisierung von Chromosomen soll der implementierte Evolutionäre Algorithmus schneller gegen das Optimum konvergieren.

### 3.3.2 Bewerten der Maschinenbelegung

Die Bewertung der Individuen orientiert sich an den in Kapitel 3.1.3 geschilderten Optimierungszielen. Je besser diese durch die repräsentierte Maschinenbelegung verwirklicht werden, desto besser soll auch die Bewertung ausfallen. Ziel der Optimierung ist die Minimierung verschiedener Zeitkriterien. Um diese zu bestimmen, ist es notwendig, aus den im Chromosom gespeicherten Parametern den konkreten Maschinenbelegungsplan zu erzeugen. Dies beinhaltet die Berechnung der anfallenden Rüst- und Bearbeitungszeiten, sowie die Ermittlung der Anfangs- und Endtermine der Fertigungsaufträge. Wenn dies geschehen ist, werden alle Zeiten des gleichen Typs summiert, um ein Maßzahlen für die zugehörigen Zeitkriterien zu erhalten. Beispielsweise werden alle Auf-, Ab- und Umrüstzeiten sowie die

Werkstoffwechselzeiten über alle Maschinen addiert um das Rüstzeitkriterium zu quantifizieren.

Jedes Kriterium ist eine Zeitgröße, die in Minuten oder Stunden gemessen wird. Die Maßeinheit aller Kriterien ist also gleich. Das Zielsystem wurde so formuliert, daß jedes Zeitkriterium zu minimieren ist. Als Gesamtbewertung kann daher die Summe der Einzelkriterien verwendet werden. Je kleiner diese Summe ist, desto besser ist das Individuum. Weil aber jedes Teilziel eine unterschiedliche Bedeutung für das gesamte Zielsystem haben kann, werden die Einzelkriterien gewichtet. Beispielsweise ist die Einhaltung von Lieferterminen wichtiger als die Minimierung der Bearbeitungszeit. Vor der Summenbildung werden die ermittelten Zeiten mit einem Faktor multipliziert, der ihre Bedeutung im Zielsystem widerspiegelt.

Zwar ist das Aggregationsverfahren trivial, die Wahl der richtigen Gewichte stellt jedoch ein Problem dar: die Einzelkriterien müssen gemäß den Vorstellungen des Unternehmens gewichtet werden. Je mehr die Bewertungsfunktion von der Realität abweicht, desto weniger stimmen die vorgeschlagenen Lösungen mit dem tatsächlichen Optimum überein. Da die hier benötigten Gewichte dem Unternehmen selten in dieser Form bekannt sind, und sich die Gewichte auch im Zeitablauf ändern können, wäre eine Benutzerschnittstelle sinnvoll, die auch unscharfe Aussagen akzeptiert, wie z.B. „Verspätungen haben einen großen Einfluß auf die Qualität des Plans“ oder „Die Verringerung von Rüstzeiten ist wichtiger als die Vermeidung von Verfrühungen“. Exakt wird die Bewertung aber erst dann, wenn die ermittelten Zeiten mit den anfallenden Kosten belastet werden. Die exakte Bestimmung der Gewichte ist jedoch nicht Gegenstand dieser Arbeit, eine annähernd realistische Bewertung soll ausreichen, die Anwendbarkeit Evolutionärer Algorithmen zu demonstrieren.

Der eigentliche Aufwand der Bewertung liegt in der Dekodierung des Chromosoms verborgen. Wie wird aus der Erbinformation die Maschinenbelegung abgelesen? Dies soll anhand des Beispiels in Abbildung 7 erläutert werden. Dort wird gezeigt, wie in drei Schritten aus dem Chromosom die

zeitliche Belegung einer Maschine abgeleitet wird. Dieser Vorgang wird für jede Maschine wiederholt werden, um den kompletten Maschinenbelegungsplan zu erhalten.

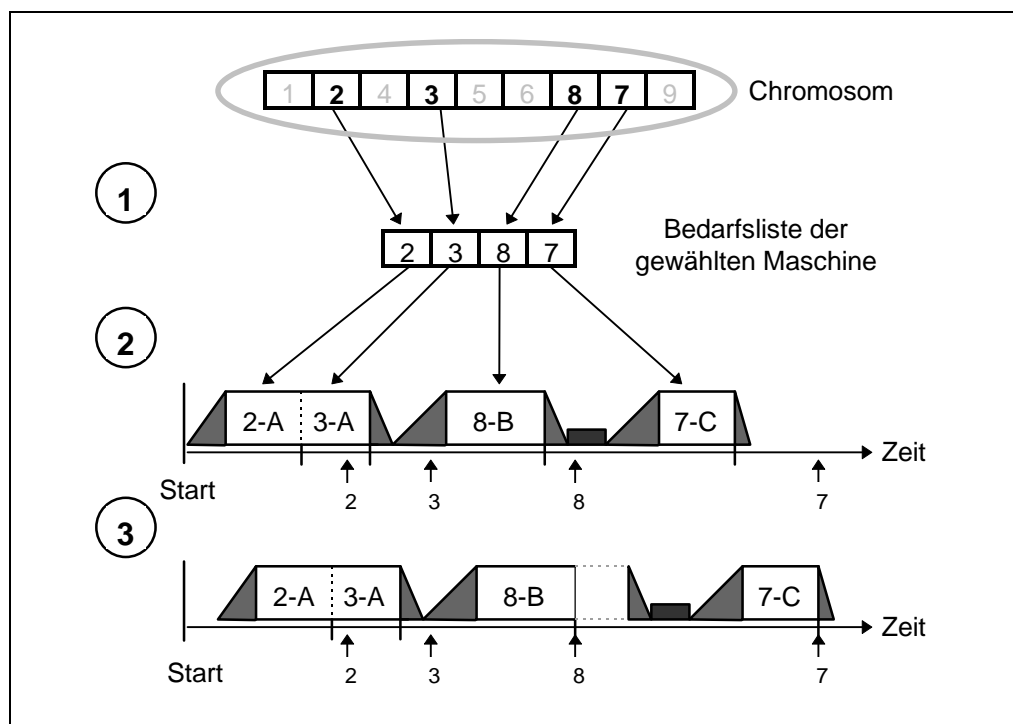




Abbildung 7: Dekodierung des Chromosoms in die Belegung einer Maschine

Im ersten Schritt werden die nicht zur gewählten Maschine gehörenden Gene aus dem Chromosom gefiltert. Der Wert des Gens entspricht der Nummer der belegten Maschine. In der Abbildung ist der Wert farbig dargestellt, die zur gewählten Maschine gehörenden Gene sind **fett** gedruckt. Die Reihenfolge der Gene ändert sich durch den Filterungsprozess nicht. Die Zahl im Gen steht für seine Funktion, d.h. den repräsentierten Bedarf. Am Ende von Schritt Eins bleibt eine Liste von Bedarfen, die von der Maschine zu produzieren ist.

In Schritt Zwei werden die Tätigkeiten und ihre Dauer ermittelt, die zur Produktion der Bedarfe in der angegebenen Reihenfolge nötig sind. Diese Tätigkeiten werden vorläufig lückenlos hintereinander auf der Zeitachse angeordnet. Dazu werden zunächst die Werkzeugkombinationen (Formen) ermittelt, die zum Guß der den Genen zugeordneten Bedarfen benötigt werden. In der Abbildung sind sie mit Großbuchstaben A, B und C bezeichnet. Anschließend werden je zwei Gene paarweise betrachtet: Beziehen sie sich auf

unterschiedliche Werkzeugkombinationen, wird die Tätigkeit des Abrüstens und des Aufrüstens eingefügt. Diese sind durch Dreiecke  symbolisiert. Bestehen die Gußteile aus verschiedenen Legierungen, wird zwischen dem Formenaus- und Einbau noch ein Werkstoffwechsel notwendig, dieser ist durch ein kleines Rechteck  dargestellt. Der eigentliche Produktionsvorgang wird mit einem weißen Rechteck gekennzeichnet, in dem die Nummer des auslösende Bedarfs und die benötigte Form steht.

Ein Rüstvorgang hat zwei Phasen: zuerst wird der Rahmen eingebaut, dann die eigentlichen Einsätze. Das Abrüsten geschieht in der umgekehrten Reihenfolge. Die Dauer dieser Vorgänge ist je Form und Maschine festgelegt. Dazu ist für jede mögliche Kombination von Form und Maschine ein Datensatz vorhanden, in dem die Rüstzeiten gespeichert sind. Er besteht aus den vier Attributen Rahmeneinbau, Einsatzeinbau, Rahmenausbau und Einsatzausbau. Eine Besonderheit der Gießerei ist es, daß mehrere Produkte mit dem selben Rahmen gegossen werden und sich nur durch den verwendeten Einsatz unterscheiden. Dies wird während der Dekodierung berücksichtigt, in diesem Fall ergibt sich die Rüstzeit nur aus dem Wechsel der Einsätze, da der Rahmen ja auf der Maschine bleibt. Im Beispiel in Abbildung 7 wird davon ausgegangen, daß die Maschine am Anfang der Planung unbelegt ist. Der erste Auftrag verursacht somit die volle Aufrüstzeit.

Die Werkstoffwechselzeit ist je Maschine konstant und somit ein Attribut derselben. Die zur Produktion benötigte Zeit läßt sich ebenfalls sehr einfach ermitteln: genau wie für die Rüstzeiten ist für jede mögliche Kombination von Form und Maschine die Zeit bekannt, die je Schuß benötigt wird. Dieser Wert wird im gleichen Datensatz wie die Rüstzeiten gespeichert. Je Schuß werden so viele Gußteile produziert, wie Nester in der Form enthalten sind. Die Bearbeitungszeit berechnet sich also aus der Bedarfsmenge, dividiert durch die Anzahl der je Schuß produzierten Teile, multipliziert mit der Zeit je Schuß.

Im in Abbildung 7 gezeigten Beispiel produzieren die ersten beiden Aufträge gleiche Teile. Daher entfallen alle Rüstzeiten zwischen diesen Produktionsvorgängen. Nun sollte auch deutlich werden, warum die

nacheinanderliegenden Bedarfe als Lose interpretiert werden können. Die Gußteile B und C bestehen aus unterschiedlichen Legierungen. Daher wird Zeit für einen Werkstoffwechselzeit benötigt.

Ordnet man alle Tätigkeiten beginnend im Zeitpunkt Null lückenlos hintereinander an, erhält man Anfangs- und Endtermine. Im dritten Schritt werden die Endtermine der Produktionsprozesse so weit wie möglich in Richtung der Fälligkeit des zugehörigen Bedarfs verschoben. Dadurch wird die Verfrühung zu minimiert. In Abbildung 7 sind die Produktionsendtermine auf der Zeitachse markiert, die spätesten Endtermine (Fälligkeiten) sind durch senkrechte Pfeile mit der Nummer des Bedarfs gekennzeichnet. Ein Verschiebung entlang der Zeitachse ist nur nach rechts möglich, da die Belegung aufgrund der Arbeitsweise von Schritt Zwei keine Lücken zwischen Start und dem Ende der letzten Tätigkeit aufweist.

Der Test auf Verschiebung beginnt mit der letzten Produktionstätigkeit. Ist das Produktionsende kleiner als der zugehörige Bedarfstermin, wird so verschoben, daß Produktionsende gleich der Fälligkeit ist. Die eventuell links angrenzenden Rüsttätigkeiten werden um die gleiche Distanz mitverschoben. Dies ist mit 7-C geschehen. Für alle anderen Aufträge geschieht folgendes: Liegt das Produktionsende früher als der späteste Endtermin, so wird der Auftrag entweder bis zum spätesten Endtermin oder bis zum Anfang der folgenden Belegung verschoben, je nachdem welcher Termin früher liegt. Für 8-B trifft der erste Fall zu, für 3-A und 2-A der zweite, da eine Verschiebung zum spätesten Termin zu einer Überlappung führen würde.

Als Ergebnis des letzten Schritts hat man nun die geplante Belegung einer Maschine. Wiederholt man diese Schritte für alle Maschinen, erhält man die Maschinenbelegung, wie sie an die Werkstatt weitergegeben werden kann.

### 3.3.3 Implementierte Zufallsprozesse

Evolutionäre Algorithmen basieren auf Zufallsprozessen, die in den Phasen der Selektion, Rekombination und Mutation benutzt werden. Die simulierte Evolution setzt daher eine Funktion, die Zufallszahlen erzeugen kann, voraus.

Eine solche Funktion wird Zufallszahlengenerator genannt. Durch ein Computerprogramm erzeugte Zufallszahlen sind jedoch letztendlich immer deterministisch, so daß genaugenommen nur von Pseudo-Zufallszahlen gesprochen werden darf.

Von einem guten Zufallszahlengenerator fordert man eine möglichst hohe Unabhängigkeit von aufeinanderfolgenden Werten, eine große Periode (Anzahl der Werte, nach der sich die Zufallszahlenfolge zu wiederholen beginnt) und eine Gleichverteilung der Zufallszahlen. Dadurch soll die deterministische Natur möglichst gut vor dem Anwender verborgen werden ([Mül-Clo]). Aus der Sicht der Implementierung wird muß auch der Aspekt der Effizienz berücksichtigt werden. Gerade Evolutionäre Algorithmen benötigen sehr viele Zufallszahlen.

In ANSI C++ ist in der Standardbibliothek die Funktion *rand()* enthalten, die zufällige Integer-Werte im Intervall von 0 bis `RAND_MAX` generiert. In der Implementierung benutzen Compiler *Borland C++ V4.0 für Windows* ist sie durch ein multiplikatives Kongruenzverfahren mit der Periodenlänge  $2^{32}$  implementiert ([Borland]). Die multiplikativen Kongruenzgeneratoren bieten eine gute Mischung aus Effizienz und statistischer Güte. Daher wurde auf die Implementierung eines eigenen Zufallszahlengenerators verzichtet und der Nachteil, daß die Standardfunktion nur einen Zufallszahlenstrom verwaltet, in Kauf genommen. Da der gewählte Lösungsansatz diskreter Natur ist, kommt auch die Realisierung mit diskreten Zufallsprozessen aus. Diese werden wie folgt auf die Standardfunktion *rand()* zurückgeführt:

- *Uniform(Num)*: Diese Funktion liefert gleichverteilte Zufallszahlen aus dem Intervall  $[0, Num - 1]$ . Die Grenzen des Intervalls sind durch den Aufbau von C++-Arrays begründet. In einem C++ Array mit *Num* Elementen wird auf das erste Element mit dem Index 0 und auf das letzte mit dem Index *Num-1* zugegriffen. Funktionen, die nach diesem Schema implementiert sind, lassen sich ohne Ballast verwenden und sind außerdem mit dem C++ Sprachgebrauch konsistent.

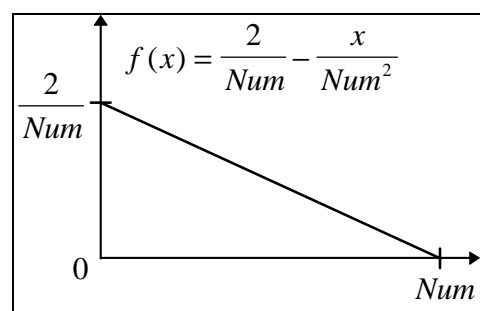
$$Uniform(Num) = \frac{rand()}{RAND\_MAX + 1} * Num$$

- *Bernoulli(P)* liefert einen booleschen Wert. *TRUE* wird mit der Wahrscheinlichkeit *P*, und dementsprechend *FALSE* mit der komplementären Wahrscheinlichkeit  $1-P$  erzeugt. *P* ist in Promille anzugeben.

$$Bernoulli(P) = \begin{array}{ll} TRUE & \text{falls } Uniform(1000) < P*1000 \\ FALSE & \text{sonst} \end{array}$$

- *Triangle(Num)* erzeugt dreiecksverteilte Zufallszahlen im Intervall  $[0, Num - 1]$ . Die zugehörige

Wahrscheinlichkeitsfunktion ist in der nebenstehenden Abbildung dargestellt. Die Dreiecksverteilung wird überall dort benutzt, wo die Auswahlwahrscheinlichkeit



proportional zu einem anderen Kriterium sein soll. Z.B. sollen bei Mutationen kleine Änderungen häufiger vorkommen als große. Die zu bestimmende Zufallszahl gibt dann die Änderungsgröße an. 0 wäre die kleinste Mutation und  $Num-1$  die größte. Im Gegensatz zu der in Evolutionsstrategien verwendeten Normalverteilung läßt sich mit der Dreiecksverteilung das Intervall, aus dem die Zufallszahlen stammen, exakt angeben. Dies ist wichtig, da hier mit einem diskreten Ansatz gearbeitet wird. Bei der Normalverteilung wäre es möglich, daß Werte außerhalb des zulässigen Bereichs generiert würden, die von der Implementierung abgefangen werden müssen. Die Länge des Intervalls kann für die Mutation als maximale Schrittweite verstanden werden. Evolutionsstrategien interpretieren statt dessen die Streuung der Normalverteilung als Schrittweite.

Auch für die Selektion läßt sich die Dreiecksverteilung gut verwenden, denn durch einen kleinen Trick kann eine Auswahl proportional zur Tauglichkeit mit einer Skalierung (siehe Kapitel 2.3.2) kombiniert werden: die Individuen werden gemäß ihrer Tauglichkeit sortiert angeordnet, das

Beste befindet sich an der Position 0. Die gezogene Zufallszahl bestimmt dann die Position des gewählten Individuums. Durch die Sortierung ist die Auswahlwahrscheinlichkeit proportional zur Bewertung, und durch die Linearität der Verteilung sind die Wahrscheinlichkeitsunterschiede zwischen den Individuen konstant (Skalierung).

Schließlich läßt sich ein Zufallszahlengenerator für die Dreiecksverteilung auch effizienter als die Normalverteilung realisieren. Mit der Inversionsmethode können im Intervall  $[0,1]$  gleichverteilte Zufallszahlen  $z$  in eine beliebige Verteilung transformiert werden, indem die Umkehrfunktion der gewünschten Verteilungsfunktion auf  $z$  angewandt wird, also  $F^{-1}(z)$  (vgl. [Mül-Clo]). Damit erhält man folgende Formel für die Dreiecksverteilung:

$$\text{Triangle}(Num) = Num - \left| Num * \sqrt{1 - \frac{\text{rand}()}{\text{RAND\_MAX} + 1}} \right|$$

### 3.3.4 Rekombinationsstrategien

Die Anwendung Genetischer Algorithmen hat eine Fülle verschiedener Crossover-Operatoren hervorgebracht, die mehr oder weniger auf bestimmte Anwendungsgebiete spezialisiert sind. Auch das Konvergenzverhalten, d.h. wie schnell und sicher sich die Lösung gegen das gesuchte Optimum strebt, wird durch den gewählten Operator beeinflusst. Leider gibt es kein universelles Verfahren, das optimale Konvergenz garantiert. Der jeweils beste Operator kann von Problemklasse zu Problemklasse variieren. Um die optimale Rekombinationsstrategie für den hier betrachteten Fall zu finden, wurden verschiedene Crossover-Varianten, die ursprünglich für das Travelling-Sales-Person-Problem entwickelt wurden, implementiert und getestet. Da sich beide Aufgabenstellungen auf ein Permutationsproblem zurückführen lassen, wird vermutet, daß sich die für das TSP-Problem gewonnenen Erkenntnisse übertragen lassen. Außerdem gewährleisten diese Operatoren die formale Gültigkeit des Chromosoms, nämlich daß keine Gene verloren gehen oder vervielfältigt werden. Die in den Evolutionsstrategien angewandten Rekombinationsverfahren lassen sich auf Permutationsprobleme nicht anwenden.

Die Implementierung stützt sich auf die bei [Oliver] beschriebenen Crossover-Operatoren. Da bereits beim Aufbau des Chromosoms die Anwendung permutierender Operatoren berücksichtigt wurde, lassen sich die dort beschriebenen Arbeitsweisen nahezu unverändert übernehmen. Die Schnittstelle zur Rekombinationsfunktion wurde wie folgt standardisiert: Eingangsparameter sind zwei Elternindividuen. Aus diesen werden *zwei* zueinander inverse Nachkommen erzeugt. Dies schließt die Bevorzugung des Erbgutes eines Elternteils aus, keine Erbinformationen geht verloren (siehe Abbildung 2). Im folgenden werden die implementierten Rekombinationsstrategien beschrieben:

- **No Crossover (NOX):** Wie der Name schon sagt, wird in diesem Fall keine Rekombination des Erbmaterials durchgeführt, sondern die Nachkommen sind identische Kopien der Elternindividuen. Dieses Verfahren wurde implementiert, um feststellen zu können, ob die Einführung von Rekombination zu verbessertem Konvergenzverhalten führt.
- **Order Crossover (OX):** Zuerst wählt man zufällig zwei Genpositionen, diese werden zu den Grenzen des Crossover-Intervalls. Jede Position besitzt die gleiche Auswahlwahrscheinlichkeit. Die Gene im Intervall von Elter 1 werden positionsgleich in Kind 2 kopiert. Die dann noch freien Positionen in Kind 2 werden mit Genen gefüllt, die im Crossover-Intervall nicht vorkommen. Dazu benutzt man die Reihenfolge der Gene aus Elter 2, beginnend mit dem ersten Gen hinter der rechten Intervallgrenze. Abbildung 8 verdeutlicht dieses Verfahren. Analog wird für Elter 2 und Kind 1 vorgegangen. Der OX-Operator erhält die absolute Position der Gene eines Elternteils im Crossover-Intervall und die Reihenfolge, d.h. die relative Position, der Gene des anderen Elternteils.

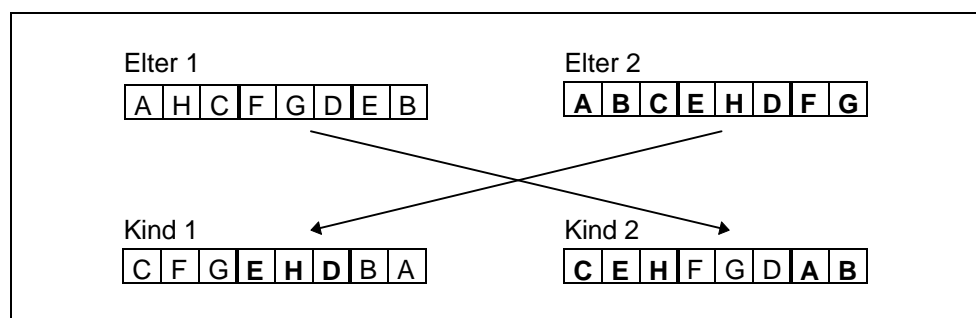


Abbildung 8: Order Crossover

- **Partially Matched Crossover (PMX):** Die Elternchromosomen werden zunächst unverändert in die Nachkommen kopiert, dann wird genau wie beim OX-Verfahren ein Crossover-Intervall bestimmt. Die Gene im Intervall werden ausgetauscht. Dadurch können in einem Chromosom Gene doppelt oder überhaupt nicht mehr auftauchen. Deswegen wird für jedes doppelte Gen in Kind 1, das sich nicht im Crossover-Intervall befindet, die Stelle gesucht, an der es im Elter 2 vorkommt. Das Gen in Kind 1 wird dann durch das Gen im Elter 1, das sich an der gefundenen Position befindet, ersetzt. Dies wird solange wiederholt, bis die Gültigkeit des Chromosoms wiederhergestellt ist, d.h. keine doppelten oder fehlenden Gene mehr auftreten. Analog wird Kind 1 komplettiert. Abbildung 9 zeigt ein Beispiel für die Anwendung des PMX-Operators. Die PMX-Strategie erhält die absoluten Positionen der Gene im Crossover-Intervall und die einiger Gene außerhalb.

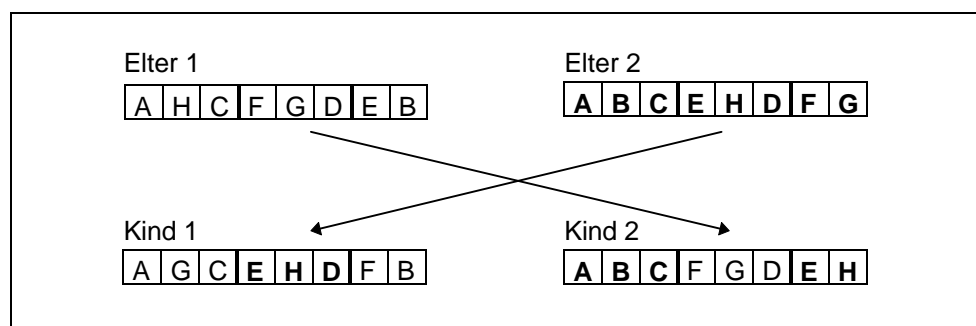


Abbildung 9: Partially Matched Crossover

- **Cycle Crossover (CX):** Diese Rekombinationsstrategie wurde unter anderen Gesichtspunkten entwickelt als OX und PMX: jede Position im Nachkommen soll mit einem Gen besetzt werden, das an gleicher Position in einem der beiden Elternteile vorkommt; die Nachkommen sollen Permutationen der Eltern sein ([Oliver]). Die Arbeitsweise soll anhand des Beispiels in Abbildung 10 demonstriert werden: das Gen in der ersten Position von Kind 1 stammt aus einem zufällig gewählten Elter, im Beispiel ist dies Elter 1. An dieser Position steht in Elter 2 das Gen **B**. Aufgrund der Zielsetzung des Verfahrens kann dieses Gen nicht mehr aus Elter 2 in das

Kind gelangen, da diese Position bereits durch das Gen A besetzt ist. Gen B muß daher aus Elter 1 stammen, dort befindet es sich an letzter Stelle. Genauso wird für **G** und **A** verfahren. A ist jedoch bereits in Kind 1 enthalten, so daß das Verfahren abbricht. Für die nächste freie Position (2) entscheidet wieder der Zufall über den Elter, aus dem das Gen genommen werden soll. Die zwischen den Zufallsentscheidungen gewählten Gene bilden einen sogenannten Zyklus. Die Nummer des Zyklus ist in Abbildung 10 unterhalb der Kind-Chromosomen eingetragen. Im Beispiel wird in den folgenden beiden Zyklen jeweils Elter 2 gewählt. Sind alle Positionen belegt, ist die Arbeit des Cycle Crossovers getan. Kind 2 beruht auf der Inversion der für Kind 1 getroffenen Zufallsentscheidungen.

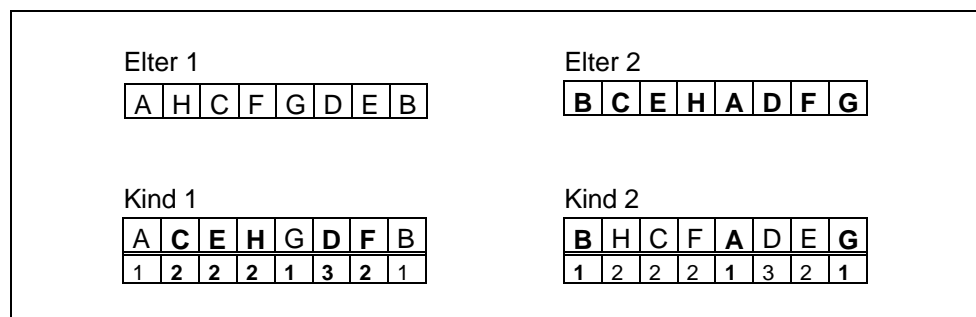


Abbildung 10: Cycle Crossover

Sowohl [Oliver] als auch [Schöneburg2] haben diese Rekombinationsstrategien experimentell an einem TSP für 30 bzw. 25 Städte getestet. Beide kamen zu folgendem Ergebnis: jeder Operator führte zu Konvergenz, wobei das OX-Verfahren das beste Konvergenzverhalten zeigte, gefolgt von PMX und CX. Die absoluten Unterschiede der gefundenen Lösungen waren nur gering. In dieser Reihenfolge wurden sie auch für das hier untersuchte Maschinenbelegungsproblem implementiert und getestet. Andere Strategien, wie z.B. das von [Liepins] vorgeschlagene Greedy Crossover, das heuristisches Wissen über optimale Anordnungen nutzt, z.B. beim Job-Shop-Scheduling, wurden in diese Arbeit nicht berücksichtigt.

### 3.3.5 Mutation der Maschinenbelegung

Ein Gen kodiert drei Informationen: den repräsentierten Bedarf, die belegte Maschine, und die Position in der Liste der von einer Maschine zu

produzierenden Bedarfe. Da Bedarfe weder vervielfältigt noch entfernt werden dürfen, können nur die letzten beiden Daten manipuliert werden. Dazu werden zwei verschiedene Mutationsoperatoren eingesetzt, deren Eintrittswahrscheinlichkeiten unabhängig voneinander sind. Für jedes Gen gelten die gleichen Mutationswahrscheinlichkeiten.

Die Maschinenmutation ist der erste Operator. Er ändert die vom Bedarf belegte Maschine. Dazu wird die Menge der zulässigen Maschinen bestimmt. Aus dieser Menge wird per Zufall die neu zu belegende Maschine gewählt (diese kann auch gleich der bereits belegten sein). Durch die Beschränkung auf die Menge der zulässigen Maschinen wird sichergestellt, daß der Maschinenbelegungsplan seine formale Gültigkeit behält. Mutationen sollen bevorzugt zu kleinen Änderungen führen. Die Auswirkung eines Maschinenwechsels sind jedoch leider sehr stark situationsabhängig, d.h. ohne Wissen über die konkrete Maschinenbelegung läßt sich das Ausmaß der Änderung nicht abschätzen. Daher wurde in den ersten Experimenten jede Maschine mit der gleichen Wahrscheinlichkeit gewählt. Der Versuch, Maschinen zu bevorzugen, deren Geschwindigkeit von der aktuell gewählten minimal abweicht, führte zu keinen erkennbaren Vorteilen. In Anlehnung an eine im Unternehmen verwendete Heuristik („Versuche zuerst die schnellsten Maschinen zu belegen“) wurde der Mutationsoperator schließlich so modifiziert, daß schnellere Maschinen eine höhere Auswahlwahrscheinlichkeit besitzen. Dadurch konvergiert das Verfahren schneller, und es wurden insgesamt bessere Lösungen erzielt.

Der zweite Mutationsoperator ändert die Position eines Gens im Chromosom. Dadurch ändert sich analog die Bearbeitungsreihenfolge der Bedarfe. Es wird angenommen, daß die Auswirkung auf die Maschinenbelegung proportional zur Distanz ist, um die das Gen verschoben wird. Kleine Distanzänderungen sollen daher häufiger auftreten. Die Verteilung der Zufallsvariable, die die neue Position des Gens bestimmt, wurde wie folgt konstruiert.

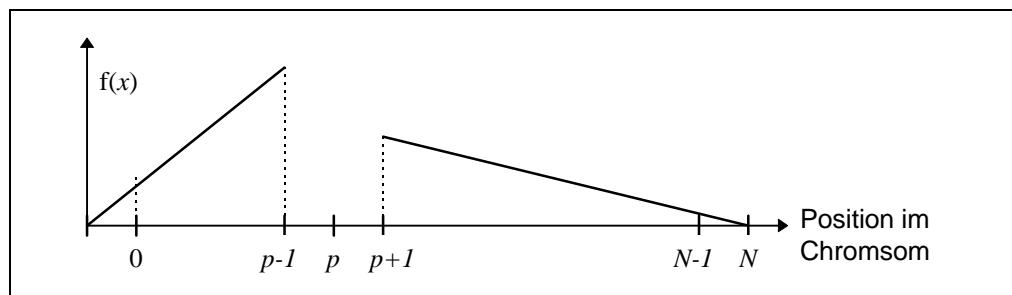


Abbildung 11: Wahrscheinlichkeitsfunktion der Positions-Mutation

Wenn das zu mutierende Gen nicht am Anfang oder Ende des Chromosoms positioniert ist, wird zufällig die Richtung bestimmt, in die das Gen verschoben wird. Jede Richtung hat die gleiche Wahrscheinlichkeit. Abhängig von der getroffenen Wahl, wird die Distanz ermittelt. Sie folgt einer Dreiecksverteilung über dem Intervall von 1 bis zum linken bzw. rechten Rand des Chromosoms. Die resultierende Wahrscheinlichkeitsfunktion für die neue Position des Gens zeigt Abbildung 11. Der Verlauf der Funktion hängt von den Parametern  $p$  und  $N$  ab.  $p$  ist die Position des zu verschiebenden Gens,  $N$  gibt die Anzahl der Gene im Chromosom an, wobei die Anfangsposition gleich 0 ist. Da die Entscheidung darüber, ob eine Positions-Mutation stattfinden soll oder nicht, bereits getroffen wurde, nimmt die Zufallsvariable nie den Wert  $p$  an. Die unterschiedliche Neigung der Geraden auf beiden Seiten von  $p$  läßt sich dadurch erklären, daß die Summe der Wahrscheinlichkeiten auf jeder Seite 0,5 betragen muß, die Intervalle aber eine unterschiedliche Länge besitzen.

### 3.3.6 Generationenübergang

Die Erzeugung einer neuen Generation geschieht wie in Kapitel 2.3.5 beschrieben. Zur Selektion der Elternpaare (Kante (1)) werden die Individuen der Elternpopulation gemäß ihrer Qualität sortiert, wobei das beste Individuum an erster Stelle steht. Nun werden je zwei Zufallszahlen generiert, die die Positionen der zu reproduzierenden Eltern angeben. Die Zufallszahlen folgen einer Dreiecksverteilung über den zur Verfügung stehenden Positionen der Eltern, wobei das Maximum auf der ersten Position liegt. Dieses Verfahren stellt zweierlei sicher: erstens werden sich die besten Individuen am häufigsten

fortpflanzen, zweitens sind Fortpflanzungswahrscheinlichkeiten geglättet, d.h. die Unterschiede zwischen den einzelnen Individuen sind konstant.

Die Übergangspopulation setzt sich aus allen so erzeugten Kinder und den besten unveränderten Elternindividuen, der sogenannten Elite, zusammen. Die Einführung der Elite gewährleistet, dass die beste, jemals gefundene Lösung nicht verloren geht. Für die Gütefunktion hat dies einen monoton fallenden Verlauf zur Folge. In der Übergangspopulation befinden sich mehr Individuen, als in die Elternpopulation der Folgegeneration übernommen werden können. Die nun nötige Selektion wird streng deterministisch durchgeführt: nur die besten Individuen gelangen in die nächste Generation.

Die Größe der Eltern- und Kindpopulationen sowie der Elite ist unter folgenden Einschränkungen frei variierbar: die Anzahl der Elitemitglieder ist kleiner als die Anzahl der Eltern, es gibt mehr Kinder als Eltern und die Anzahl der Kinder ist gerade (da bei jeder Reproduktion immer zwei Kinder erzeugt werden).

### **3.3.7 Selbstadaption**

Da sich durch Selbstadaption das Konvergenzverhalten verbessert, und das Problem der optimalen Parameterwahl wegfällt, soll auch dieses Prinzip in den realisierten Algorithmus eingebaut werden. Dies gelingt zwar nur in einem bescheidenen Maß, aber die Testergebnisse zeigen, daß sich auch mit einfachen Methoden sichtbare Erfolge erzielen lassen.

Gegenstand der Selbstadaption sind ausschließlich die Mutationsparameter, d.h. die Eintrittswahrscheinlichkeiten für Position- und Maschinenänderung, sowie die maximale Schrittweite. Diese Parameter werden als weitere Eigenschaften des Individuums angesehen und gleichberechtigt neben dem Chromosom gespeichert. Die anderen Parameter, Rekombinationswahrscheinlichkeit und Populationsgrößen, lassen sich nicht als Attribute des Individuums interpretieren und werden daher in dieser sehr einfachen Version der Selbstadaption nicht berücksichtigt.

Damit die im Individuum gespeicherten Mutationsparameter ebenfalls der Evolution unterliegen, müssen sie bei der Reproduktion an die Nachkommen weitergegeben werden. Der Mutationsoperator wird folgendermaßen erweitert: vor der Mutation des Chromosoms werden zunächst die Kontrollparameter mutiert. Die Wahrscheinlichkeit und Schrittweite dieser Änderung ist konstant. Je Parameter wird außerdem eine Ober- und Untergrenze festgelegt. Die neuen Mutationsparameter bestimmen anschließend die Mutation des Chromosoms. Individuen mit guten Parametereinstellungen werden häufiger bessere Nachkommen haben, so daß sich die optimalen Parameter über die ganze Population ausbreiten. Sind zu verschiedenen Zeitpunkten der Simulation andere Einstellung optimal, können sich die Parameter der neuen Situation durch Mutation und Selektion automatisch anpassen.

### **3.4 Erprobung**

Ziel der Erprobung ist es, Aussagen über die Qualität des Verfahrens zu treffen. Dazu wird ein Ausschnitt der Fertigungsstruktur in der Gießerei modelliert, und eine Bedarfsliste erzeugt, die einerseits der realen Auftragsstruktur ähnelt, aber so angepaßt ist, daß es auf einigen Maschinen zu Engpässen kommt. Für diese Situation soll der implementierte Algorithmus mit verschiedenen Parametereinstellungen Maschinenbelegungspläne erstellen. Anhand der Testergebnisse wird dann das Konvergenzverhalten der Varianten mit beurteilt. Die so gefundene ideale Einstellung kann dann in die rechnergestützte Plantafel übernommen werden und mit der Komplexität der Praxis konfrontiert werden. Dann wird sich herausstellen, inwieweit der Evolutionäre Algorithmus der bisherigen Planung über- bzw. unterlegen ist.

#### **3.4.1 Modellierte Planungssituation**

Zur Produktion stehen fünf Maschinen zur Verfügung, die sich durch Geschwindigkeit und technische Eigenschaften unterscheiden, d.h. nicht jede Maschine kann jedes Gußteil produzieren. Mit diesen Maschinen können insgesamt zehn verschiedene Teile gegossen werden, für jedes Teil sind im Schnitt drei Maschinen zulässig. Für die zehn Einsätze (Gußformen) stehen

sieben Rahmen zur Verfügung, drei bzw. zwei Einsätze können mit dem gleichen Rahmen verwendet werden (Einsparen von Rüstzeiten), während für die restlichen fünf Einsätze individuelle Rahmen benötigt werden. In der Gießerei werden zwei Legierungen verwendet. Jedes Gußteil ist mit einer Werkzeugkombination assoziiert, die aus Vereinfachungsgründen mit derselben Nummer erhalten.

Je Gußteil sind für jede zulässige Maschine die entsprechenden Rüstzeiten für den Ein- und Ausbau der Rahmen und Einsätze angegeben, sowie die Zeit, die pro Schuß benötigt wird. Je Schuß soll nur ein Teil produziert werden. Dies vereinfacht die Dateneingabe im Testfall. Auf das Prinzip des Verfahrens hat dies keinen Einfluß, da dieser Faktor ja nur bei der Umrechnung der Bedarfsmenge in die Schußzahl eine Rolle spielt. Weiterhin wird die Werkstoffwechselzeit berücksichtigt, die für jede Maschine definiert ist.

Auf die Berücksichtigung des Betriebskalenders, d.h. in welchen Zeiträumen produziert wird, und in welchen nicht (Wochenenden), wurde verzichtet. Es wird angenommen, daß die Maschinen ohne Unterbrechung arbeiten. Im realisierten Leitstand existieren Funktionen, die terminierte Arbeitszeiten aufgrund des Betriebskalenders in echte Zeiten umrechnen können. Diese konnten in der hier implementierten Testversion aber nicht verwendet werden.

Der Planungszeitraum ist in 90 Perioden (Tage) unterteilt, indem 59 Bedarfe erfüllt werden müssen. Da jeder Bedarfssatz einem Gen entspricht, ist die Komplexität mit einem TSP-Problem für 59 Städte vergleichbar. Der Bedarf für einige Gußteile wurde so hoch gewählt, daß die Kapazität der günstigsten Maschinen zur rechtzeitigen Fertigstellung nicht ausreicht. Hier treten Engpässe auf, die durch Vorverlegung oder Ausweichen auf andere Maschinen überbrückt werden müssen.

Die genauen Details der modellierten Planungssituation sind in der Datei DATA.TXT (siehe Anhang A: *Inhalt der Diskette*) gespeichert, die vor jeder Planung eingelesen wird. Dadurch kann das Verhalten der Implementierung unter verschiedenen Bedingungen getestet werden.

### 3.4.2 Durchgeführte Experimente

Der Schwerpunkt der Erprobung liegt auf den vier verschiedenen Rekombinationsstrategien. Jede Strategie wird in drei Variationen getestet:

- 1) im Standardfall, so wie sie in der Literatur beschrieben ist
- 2) mit Normalisierung der Chromosomen
- 3) wie 2) mit zusätzlicher Selbstadaption der Mutationsparameter

Insgesamt werden also zwölf verschiedene Varianten untersucht. Die Parametereinstellungen des Populationsverhaltens bleiben in allen Varianten konstant. Die Werte der Parameter wurden in Vorabexperimenten während der Implementierung ermittelt, und stellen einen Kompromiß zwischen den Varianten dar. Prinzipiell könnten für jede Variante optimale Parameterwerte ermittelt werden, dies würde jedoch einerseits einen erheblichen Aufwand bedeuten, andererseits einen Vergleich erschweren, da in jeder Generation eine unterschiedliche Anzahl von Individuen ausgewertet würde. Die Vermutung, daß die jeweils optimalen Populationsparameter nahe zusammen liegen, konnte durch die Vorabexperimente erhärtet werden. Die Elternpopulation soll 28 Individuen umfassen, aus denen in jeder Generation 38 Kinder erzeugt werden. Die Elite besteht aus einem einzigen Individuum. Die Rekombinationswahrscheinlichkeit beträgt 60%, dies ist ein Wert, den auch [Goldberg] empfiehlt.

Die Mutationsparameter sind nur bei Experimenten ohne Selbstadaption konstant. Die Eintrittswahrscheinlichkeit eines Maschinenwechsels beträgt je Bedarf 2%, die eines Positionswechsels 3%. Die maximale Schrittweite bei einer Positionsmutation ist gleich der halben Länge des Chromosoms, beträgt also 29 Gene bei 59 Bedarfen. Die hier implementierten Varianten mit Selbstadaption optimieren diese Parameter während der simulierten Evolution selbständig, so daß eine Ermittlung günstiger Parametereinstellungen entfällt.

Für die Optimierungskriterien werden die folgenden Gewichte gewählt. Aus den in Kapitel 3.3.2 genannten Gründen stimmen sie nur in der Tendenz:

- Verspätungen: das wichtigste Optimierungsziel wird mit vier multipliziert

- Rüstzeiten: das Rüsten verursacht höhere Kosten, während der Stillstandszeit können keine Deckungsbeiträge erwirtschaftet werden. Daher geht dieses Kriterium doppelt in die Bewertung ein.
- Die Arbeitszeit bleibt unverändert, der Gewichtungsfaktor beträgt Eins.
- Verfrühungen werden als weniger wichtig angesehen und gehen daher nur zu einem Viertel in die Bewertung ein.

Für jede zu testende Variante wird, beginnt mit einer zufällig erzeugten Startpopulation, die Evolution über 600 Generationen simuliert. Die Bewertung des besten Individuums und die ihr zugrundeliegenden Kriterien jeder zehnten Generation werden protokolliert. Da der Algorithmus stochastischen Einflüssen unterliegt, wird jedes Experiment 20-mal wiederholt und das arithmetische Mittel der Einzelergebnisse bestimmt. Dadurch wird die Verfälschungsgefahr aufgrund von zufälligen Abweichungen verkleinert.

### 3.4.3 Experimentelle Ergebnisse

Die Ergebnisse der Experimente sind in den Abbildungen 12 bis 15 dargestellt. Die Bewertung des besten Individuums je Generation ist auf der Ordinate aufgetragen, die Nummer der Generation auf der Abszisse. Jedes Diagramm enthält die Testergebnisse aller drei möglichen Varianten einer Rekombinationsstrategie. In der Legende ist hinter dem Liniensymbol die beste in allen 20 Optimierungsläufen erzielte Bewertung vermerkt.

Die reine Mutation ohne Rekombination ist am ehesten mit der klassischen Evolutionsstrategie vergleichbar. Eine Verbesserung der Güte im Laufe der Generationen ist eindeutig zu erkennen. Bemerkenswert ist, daß die Variante mit Selbstadaption nicht wie erwartet das beste Konvergenzverhalten zeigt, sondern gerade in den letzten Generationen stagniert und sogar noch von der reinen Mutation eingeholt wird. Eine mögliche Erklärung könnte ein häufiges „Steckenbleiben“ in lokalen Optima sein. Durch die Selbstadaption wird die Schrittweite und die Mutationswahrscheinlichkeit dabei so klein, daß das lokale Optimum nicht mehr verlassen werden kann. Zumindest absolut wird mit Selbstadaption aber das beste Ergebnis erzielt. Das beste Konvergenzverhalten zeigt die Mutation mit Normalisierung.

Für die Rekombination mittels Order Crossover läßt sich prinzipiell das gleiche feststellen wie für die reine Mutation. Eine Konvergenz ist zwar zu erkennen, aber der Verlauf aller Alternativen erscheint im Vergleich zur NOX-Variante nicht erfolgversprechend. Die Variante mit Normalisierung hat das beste Konvergenzverhalten und erreicht auch absolut das beste Ergebnis.

Beim Partially Matched Crossover zeigt sich eine klare Abstufung der Varianten, die mit den Erwartungen übereinstimmt. Die besten Konvergenzeigenschaften werden durch Einsatz von Normalisierung und Selbstadaption erreicht, gefolgt von ausschließlicher Normalisierung und dem Standardverfahren. Erst gegen Ende der simulierten Evolution führt die Selbstadaption zu einer stärkeren Verringerung der Konvergenzgeschwindigkeit. Durch Selbstadaption verursachte zu kleine Mutationswahrscheinlichkeiten, und ein daraus resultierendes häufigeres Steckenbleiben in lokalen Optima ist wie bei der reinen Mutation eine mögliche Erklärung.

Beim Cycle Crossover werden die Unterschiede zwischen den verschiedenen Versionen noch deutlicher. Über den gesamten Evolutionsverlauf dominiert das Verfahren mit Normalisierung und Selbstadaption. Auch die absolut besten Lösungen bestätigen die grafisch erkennbare Rangordnung.

Die Simulation von 600 Generationen dauert bei allen Varianten ungefähr gleich lang. Ein PC mit einem Intel486 DX2 Prozessor und einer Taktrate von 66MHz benötigt ungefähr 1 - 1,5 Minuten.

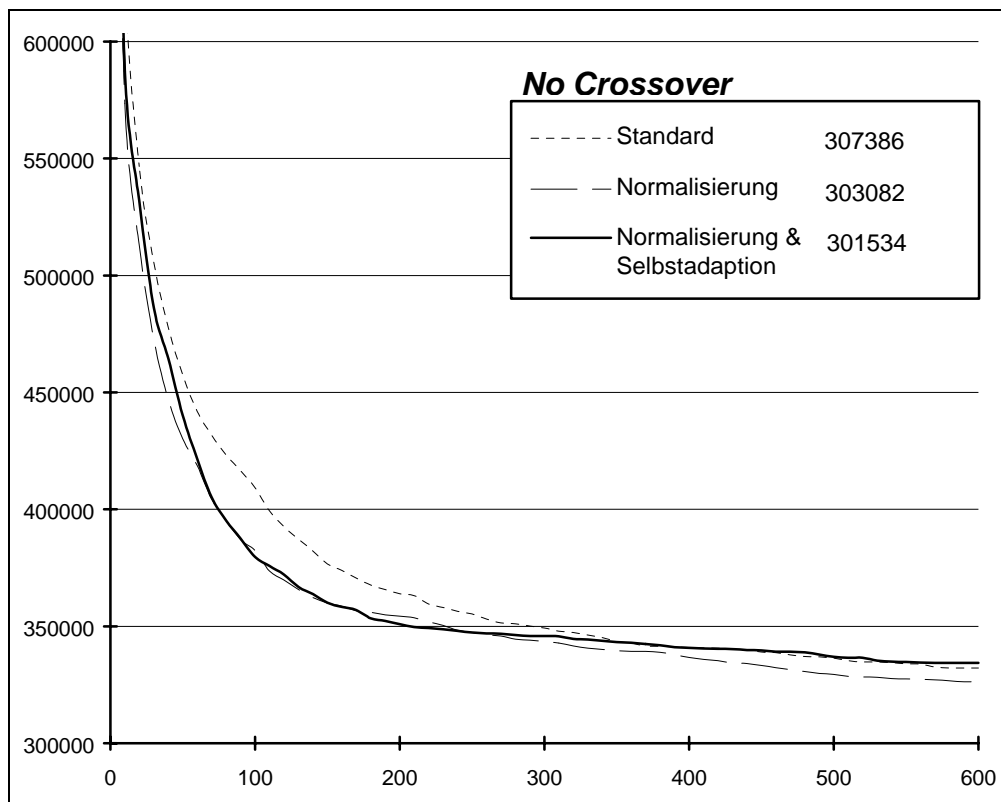


Abbildung 12: Testergebnisse für NOX

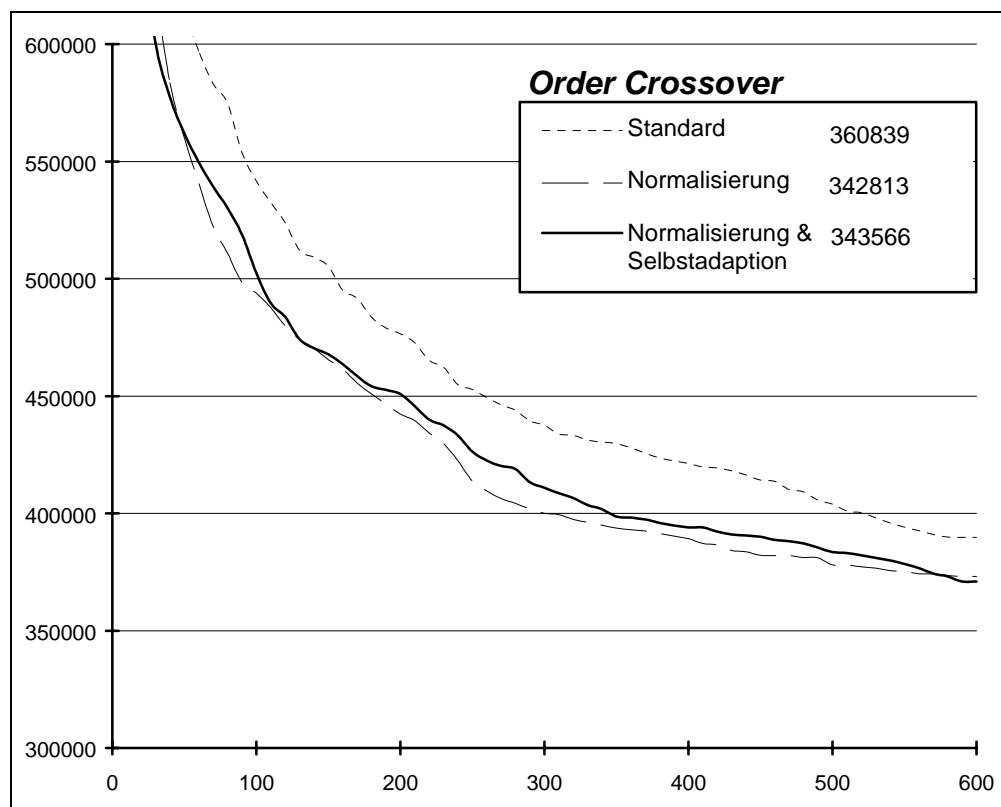


Abbildung 13: Testergebnisse für OX

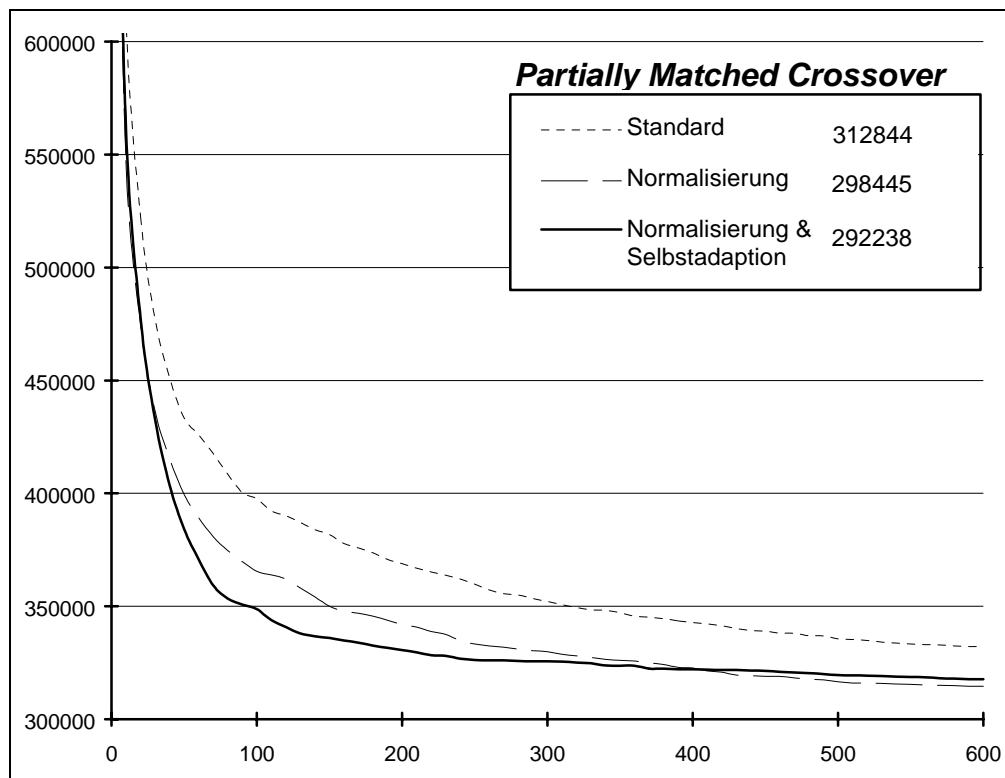


Abbildung 14: Testergebnisse für PMX

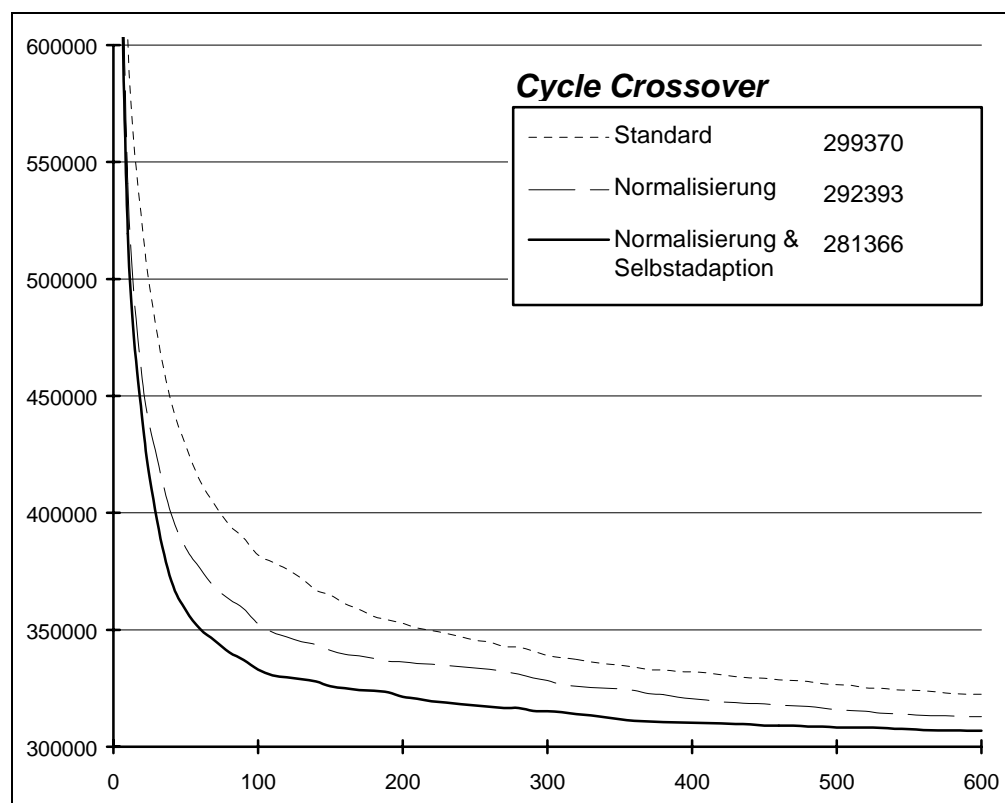


Abbildung 15: Testergebnisse für CX

### 3.4.4 Zusammenfassung und Auswertung

Ein Vergleich der Varianten aller Crossover-Operatoren zeigt deutlich, daß die Einführung der Normalisierung stets zu einer Verbesserung des Konvergenzverhaltens führt. Die Selbstadaption führt nur bei dem PMX und dem CX Verfahren zu einer Verbesserung.

In Abbildung 16 werden die besten Varianten jeder Rekombinationsstrategie in einem Diagramm zusammengefaßt. Die besten Konvergenzeigenschaften, zeigt OX, gefolgt von PMX, NOX und OX. Auffällig ist das mit Abstand schlechteste Konvergenzverhalten des Order Crossovers. Dies überrascht, wenn man bedenkt, daß gerade dieses Verfahren in Travelling-Sales-Person-Problem das Optimum am besten bestimmt. Auch die Rangordnung der Verfahren ist genau umgekehrt zu der von [Oliver] und [Schöneburg2] experimentell ermittelten.

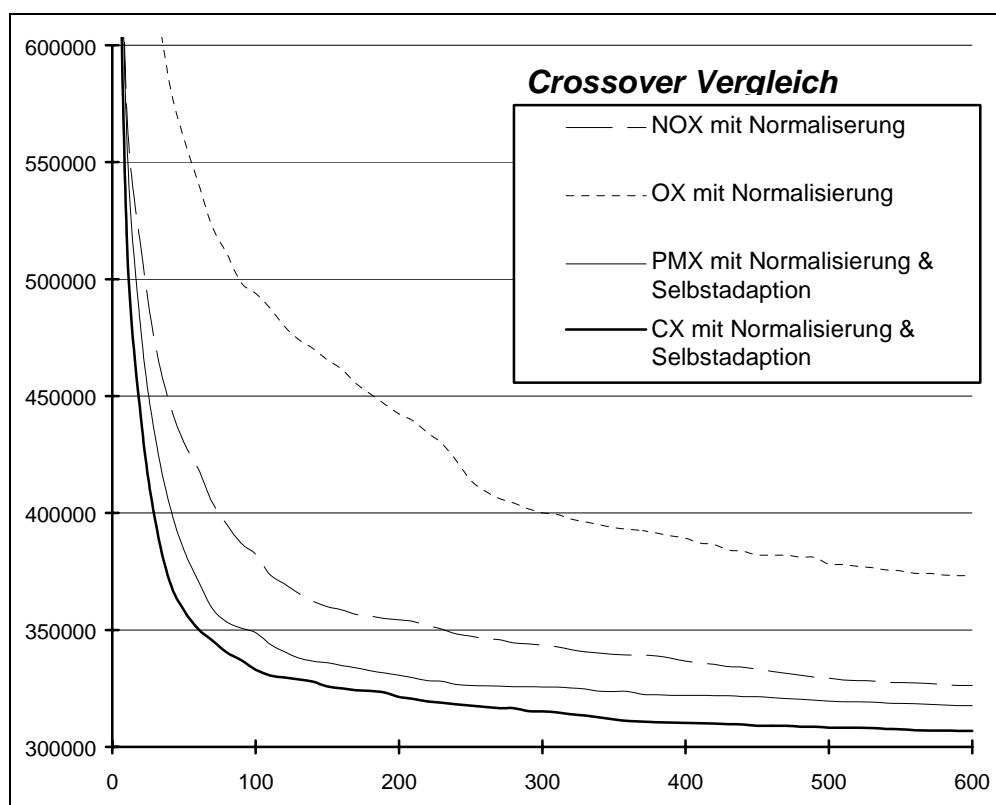


Abbildung 16: Vergleich der besten Crossover-Varianten

Betrachtet man die Arbeitsweise der Crossover-Verfahren genauer, wird dieses Ergebnis erklärbar: der Order Crossover erhält nur die relative Reihenfolge der Gene im Chromosom. Eine Gensequenz, die bei einem Elter am Ende des

Chromosoms lag, kann im Kind-Chromosom an den Anfang gelangen. Bei der Planung der kürzesten Rundreise spielt die absolute Position einer Stadt keine Rolle für die Bewertung, d.h. es ist vollkommen egal ob eine Stadt die erste oder die letzte der Tour ist. Nur die relative Anordnung bestimmt die Länge des Weges. In der Maschinenbelegungsplanung stellt sich die Situation anders dar, hier ist die absolute Position eines Gens sehr wohl von Bedeutung: jedes Gen repräsentiert ja einen Bedarf, der einen Fälligkeitstermin hat. Eine Änderung der absoluten Position in der Bearbeitungsreihenfolge vergrößert oder verkleinert somit die Verspätung und Verfrühung eines Bedarfs. Da das OX-Verfahren dies nicht berücksichtigt, wird seine Anwendung die Tauglichkeit der Nachkommen wesentlich häufiger verschlechtern als sie zu verbessern. Bei der Entwicklung des Cycle Crossovers dagegen wurde explizit das Ziel verfolgt, einen Nachkommen so zu konstruieren, ohne die absoluten Positionen der Elterngene zu verändern. Daher ist es nur folgerichtig, daß mittels CX die besten Ergebnisse bei der Maschinenbelegung erzielt werden. Unter Berücksichtigung der unterschiedlichen Problemstruktur und Arbeitsweise der Rekombinationsstrategien wird das Testergebnis durchaus verständlich.

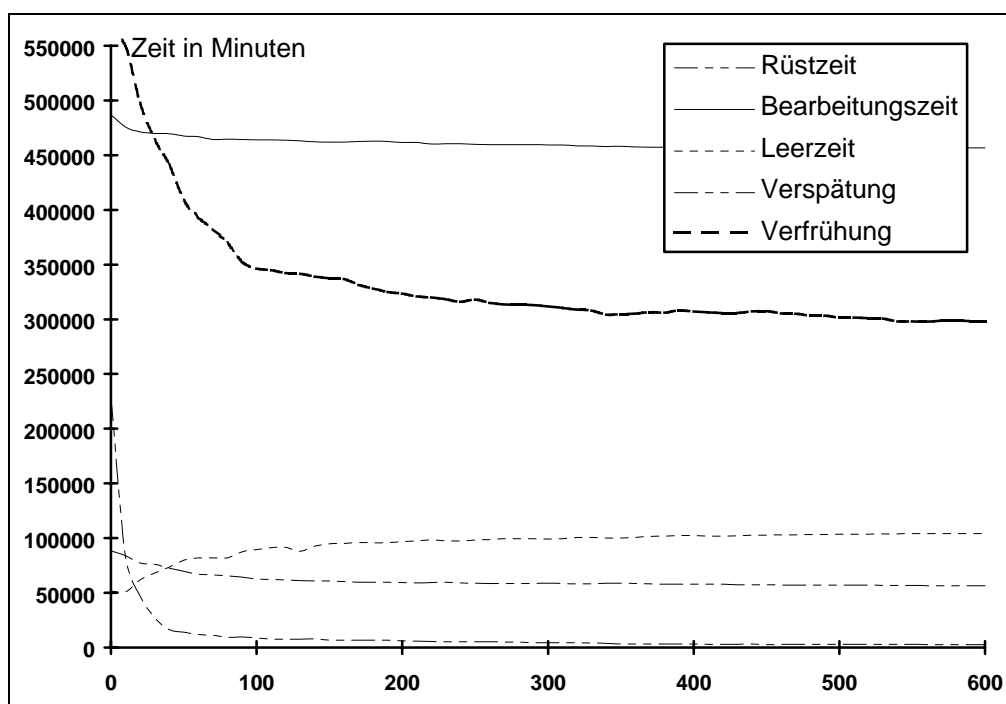


Abbildung 17: Verlauf der Einzelkriterien während der Optimierung mit Cycle Crossover, Normalisierung und Selbstadaptation

## 4 Schlußbemerkungen

Es wurde gezeigt, daß ein komplexes Problem der Praxis, die Belegung von Druckgußmaschinen, durch Evolutionäre Algorithmen bearbeitet werden kann. Die Testergebnisse belegen, daß sich die Qualität der vorgeschlagenen Lösungen im Laufe der simulierten Evolution deutlich verbessern. Die dazu benötigte Rechenzeit von wenigen Minuten liegt in einem für die praktische Anwendung akzeptablen Rahmen. Durch Optimierung der Implementierung kann die benötigte Rechenzeit noch weiter gesenkt werden.

Das nur aus theoretischen Gesichtspunkten heraus entwickelte Cycle-Crossover weist für das hier betrachtete Problem überraschenderweise das beste Konvergenzverhalten auf. Eine nähere Betrachtung der Arbeitsweise dieses Rekombinationsoperators läßt vermuten, das er für Scheduling-Probleme, in denen Fälligkeitstermine beachtet werden müssen, besonders geeignet ist.

Einige Aspekte konnten nicht mehr behandelt werden und bedürfen noch weiterer Untersuchungen. So fehlt z.B. eine absolute Einschätzung der erzielten Lösungen, da die Position des Optimums nicht bekannt ist. Ein Vergleich mit der manuellen Planung wird erst dann möglich sein, wenn der Evolutionäre Algorithmus in die Plantafel integriert wurde. Dann lassen sich aufgrund derselben Ausgangsdaten Maschinenbelegungen manuell und evolutionär planen. Die Güte beider Pläne wird durch dieselbe Bewertungsfunktion ermittelt. Auch die Bewertungsfunktion muß noch genauer den Zielvorstellungen des Unternehmens angepaßt werden.

In der Praxis wird der Maschinenbelegungsplan nicht in bestimmten Abständen komplett neu erzeugt, sondern er wird kontinuierlich den sich verändernden Bedingungen angepaßt. Evolutionäre Algorithmen sind für diese Art der Planung geradezu prädestiniert: der existierende Plan ist die Ausgangslösung (die Startpopulation) der durch die Evolution den neuen Erfordernissen angepaßt wird. Die guten Eigenschaften des Plans bleiben dabei erhalten.

## Anhang A: Inhalt der Diskette

Auf der beiliegenden Diskette befindet sich der komplette Quelltext. Die Implementierung des Evolutionären Algorithmus befindet sich in der Datei `EA.CPP` und umfaßt etwas mehr als 1000 Zeilen C++ Code. Das Programm zur Simulation des Datenbankzugriffs ist in den Dateien `DATABASE.H` und `DATABASE.CPP` abgelegt. Zur Entwicklung wurde der *Compiler Borland C++ V4.0 für Windows* benutzt. Durch die ausschließliche Verwendung von Standardbibliotheken ist die Portierbarkeit des Quelltextes gewährleistet.

Die Datei `DATA.TXT` enthält die getestete Planungssituation. Diese wird vor jedem Planungslauf eingelesen. Sie ist in Form einer ASCII-Datei gespeichert und kann mit jedem Texteditor manipuliert werden.

Schließlich sind auch noch unter MS-Windows ausführbare Programme der implementierten Varianten sowie die Ergebnisse der verschiedenen Probeläufe vorhanden. Die Dateien sind nach folgendem Schema benannt: der Name beginnt mit einem Kürzel für den verwendeten Crossover-Operator (`NO`, `OX`, `PMX` oder `CX`). Darauf kann ein `N` (für Normalisierung) und ein `A` (für Selbstadaption) folgen. Die Bezeichnung `CX-N-A` bedeutet also, daß Cycle-Crossover mit Normalisierung und Selbstadaption kombiniert wurde, während `NO` für eine reine Mutationsstrategie ohne jede Erweiterung steht. Die Testdaten enden mit `.DAT`, die ausführbaren Programme mit `.EXE`.

Die Testdaten sind als Text in einem Tabellenformat gespeichert. Dadurch können sie von jeder Standardanwendung (z.B. Word oder Excel) importiert werden. Die einzelnen Zeilen der Tabelle enthalten die durchschnittlichen Werte von 20 Experimenten für folgende Kriterien: Rüstzeit, Bearbeitungszeit, Leerzeit, Verspätung, Verfrühung sowie die Gesamtbewertung. Die letzte Zeile enthält in ihren Spalten die besten in allen 20 Experimenten ermittelten Werte für jedes Kriterium.

## Anhang B: Literaturverzeichnis

- [Ablay] Ablay, Paul: Optimieren mit Evolutionsstrategien. In: Spektrum der Wissenschaft (Juli 1987) Seite 104 - 115.
- [Borland] *Borland International, Inc.*: Programmier- und Referenzhandbücher des Borland C++ V4.0 für Windows Programmpakets. Langen, 1993.
- [Bäck1] *Bäck, Thomas; Hoffmeister, Frank; Schwefel, Hans-Paul*: Applications of Evolutionary Algorithms. Report of the Systems Analysis Research Group SYS 2/92. Universität Dortmund, Fachbereich Informatik, 1992.
- [Bäck2] *Bäck, Thomas*: Evolutionary Algorithms in Theory and Practice. Dissertation Universität Dortmund, 1994.
- [Davis] *Davis, Lawrence*: Job shop scheduling with genetic algorithms. In [Grefenstette1] Seite 160-169.
- [Glaser] *Glaser, Horst; et. al.*: PPS Produktionsplanung- und Steuerung. 2. Auflage, Wiesbaden: Gabler, 1992.
- [Goldberg] *Goldberg, David E.*: Genetic Algorithms in Search, Optimization, and Machine Learning. Reading, Massachusetts: Addison-Wesley Publishing Company, 1989.
- [Grefenstette1] *Grefenstette, J. John* (Hg.): Proceedings of the First International Conference on Genetic Algorithms. Hillsdale, New Jersey: Lawrence Erlbaum Associates, 1985.
- [Grefenstette2] *Grefenstette, J. John* (Hg.): Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms. Hillsdale, New Jersey: Lawrence Erlbaum Associates, 1987.
- [Hilliard] *Hilliard, M.R.; Liepins, G.E.; Palmer, M.; Morrow, M.; Richardson, J.*: A Classifier-based system for discovering scheduling heuristics. In [Grefenstette2], Seite 231-235.
- [Hoffmeister] *Hoffmeister, Frank; Bäck, Thomas*: Genetic Algorithms and Evolution Strategies: Similarities and Differences. Technical Report No. SYS-1/92, University of Dortmund, 1992.
- [Holland] *Holland, John H.*: Adaption in Natuaral and Artifical Systems. Cambridge, Massachusetts: University of Michigan Press, 1975.
- [Kistner] *Kistner, Klaus-Peter; Steven, Marion*: Produktionsplanung. Physica-Verlag Heidelberg, 1990.
- [Lehner] *Lehner, F; Auer-Rizzi, W.; Bauer, R.; Breit, K.; Lehner, J.; Reber, G.*: Organisationslehre für Wirtschaftsinformatiker. München: Carl Hanser Verlag 1991.

- [Liepins] *Liepins, G.E.; Hilliard, M.R.; Palmer, M.; Morrow, M.:* Greedy Genetics. In: [Grefenstette2] Seite 90 - 99.
- [Mül-Clo] *Müller-Clostermann, Bruno:* Foliensammlung zur diskreten Simulation. Vorlesungsskript Universität-Gesamthochschule-Essen, 1993/94.
- [Oliver] *Oliver, I. M.; Smith, D.J.; Holland, J.R.C.:* A Study of Permutation Crossover Operators on the Traveling Salesman Problem. In: [Grefenstette2] Seite 224 - 230.
- [Rechenberg] *Rechenberg, Ingo:* Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Stuttgart: Friederich Frommann Verlag (Günther Holzboog), 1973.
- [Schöneburg1] *Schöneburg, E.; Heinzmann, F.:* PERPLEX: Produktionsplanung nach dem Vorbild der Evolution. In *Wirtschaftsinformatik* April 1992 (Heft 2).
- [Schöneburg2] *Schöneburg, Eberhard et. al.:* Genetische Algorithmen und Evolutionsstrategien. Bonn: Addison-Wesley Publishing Company, 1994.
- [Schwefel] *Schwefel, Hans-Paul:* Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie. Basel - Stuttgart: Birkhäuser Verlag, 1977.
- [Stroustrup] *Stroustrup, Bjarne:* Die C++ Programmiersprache. Zweite Auflage, Bonn: Addison-Wesley Publishing Company, 1992.
- [Whitley] *Whitley, D.; Starkweather, T.; D'Ann Fuquay:* Scheduling problems and traveling salesmen: The genetic edge recombination operator. In [Grefenstette2], Seite 133 - 140.
- [Zimmermann] *Zimmermann, August:* Evolutionsstrategische Modelle bei einstufiger, losweiser Produktion. Frankfurt am Main: Verlag Peter Lang, 1985.

## **Anhang C: Eidesstattliche Versicherung**

„Ich versichere an Eides Statt durch meine Unterschrift, daß ich die vorstehende Arbeit selbständig und ohne fremde Hilfe angefertigt und alle Stellen, die ich wörtlich oder annähernd wörtlich aus Veröffentlichungen entnommen habe, als solche kenntlich gemacht habe und mich auch keiner anderen als der angegebenen Literatur oder sonstiger Hilfsmittel bedient habe. Die Arbeit hat in dieser oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen.“

Marl, den 19.12.1994

Christian Rodemeyer