Projekt Tarifmodell

Werner Weiser, Franz Wilhelmstötter

12. April 1999

Thema: Optimierung der Gesprächskosten bei der Sprachtelefonie

Lehrveranstaltung: PS Einführung in die Bioinformatik

Gruppe#: 2

Treffpunkt: Jeden Dienstag, um 15:30 im Büro von Roland Schwaiger

Lehrveranstaltungsleiter: Roland Schwaiger

E-Mail:

- Werner Weiser: wweiser@cosy.sbg.ac.at
- Franz Wilhelmstötter: fwilhelm@cosy.sbg.ac.at
- Roland Schweiger: rschwaig@cosy.sbg.ac.at

Inhaltsverzeichnis

Zusammenfassung

Dieses Projekt beschäftigt sich mit dem Problem, aus der Vielfalt der Telekom-Anbieter, einen, für das Gesprächsverhalten des Kunden, adäquaten Mix von Anbietern zu finden.

Ausgehend vom Telefonieverhalten des Kunden und der Kenntnis der Tarifmodelle der unterschiedlichen Telekom-Anbietern (Providern) wird ein Mix von Providern errechnet, welcher die Gesamtkosten der Telefongespräche (über einen begrenzten Zeitraum) minimiert. Die Optimierung kann jedoch nur für bereits geführte Gespräche durchgeführt werden. Unter der Annahme, daß sich das Gesprächsverhalten zeitlich nicht sprunghaft ändert, ist eine Extrapolation des vergangenen Telefonieverhaltens auf das Zukünftige durchaus zulässig.

1 Einleitung

Seit der Liberalisierung des Telekom-Marktes in Österreich per 1.1.1998 hat sich eine erstaunliche Vielfalt von sogenannten »New Entrants« etabliert - oder ist gerade dabei, die Aufbauphase erfolgreich zu bewältigen.

Derzeit sind von der Telekom Control G.m.b.H. (TKC) mehr als 30 Lizenzen für festnetzgebundene Sprachtelefonie erteilt. Da erfahrungsgemäß nicht jeder Anbieter in jedem Produktbereich das beste Preis-Leistungsverhältnis hat, sind die Zeiten des »One-Stop-Shoppings« endgültig vorbei und die für einen Kunden heute optimale Lösung besteht meist aus einer Kombination von Telekom Diensleistungen unterschiedlicher Provider, die durch moderne Least-Cost-Routertechnologie keine Komforteinbusen für den Anwender, bei einem Einsparungspotential von 20% bis 40% seiner Telefonkosten, bringt.

Welche Einsparungen ein Kunde tatsächlich erzielen kann, hängt primär von seinem spezifischen »Call-Verhalten« ab. Ist dies einmal im Detail analysiert, so kann, aus der Vielfalt der am Markt befindlichen Tarife und Produkte, die optimale Kombination für den Kunden ermittelt werden.

Das vorliegende Projekt befaßt sich mit der Erstellung von Modellen, die für ein vorliegendes Call-Verhalten jene Kombinationen von Providern ermitteln helfen, mit denen eine Kostenoptimierung erreicht werden kann.[?]

2 Optimierungsproblem

Die Eingabedaten liegen in Form von sogenannten Call-Detail-Records (CDR) vor - ein CDR setzt sich aus 1 bis m Gesprächen zusammen (vgl. Tabelle ??). Diese werden automatisch aufgezeichnet. Aus ihnen kann für jedes Telefonat die Vorwahl des Anrufers und des Angerufenen, sowie Zeitpunkt und Dauer des Gespräches entnommen werden. Auf der Seite des Telekom-Anbieters ist das genaue Tarifmodell bekannt. Somit kann jedes Gespräch auf seine Kosten abgebildet werden.

	Startvorwahl	Zielvorwahl	Startzeit	Dauer (sec)
1	0662	0664	13:24	345
2	0664	03423	07:45	45
3	07746	05674	15:34	32
4	07756	08857	18:23	123
. . .				
m	02134	09856	09:39	97

Tabelle 1: Aufbau eines typischen CDR

2.1 Formale Problembeschreibung

X ... Menge aller gültigen Telefongespräche

x_i ... das i-te Telefongespräch

SV ... Startvorwahl eines Gesprächs

ZV ... Zielvorwahl eines Gesprächs

Aus der Startvorwahl SV und der Zielvorwahl ZV läßt sich die *Ortszone* des Gesprächs ableiten (z.B. Inlandszone 1 usw.)

SZ ... Startzeit eines Gesprächs

D ... Dauer eines Gesprächs[1] (in Sekunden)

l ... Länge einer Vorwahl

CDR ... Gesprächsaufzeichnung eines abgeschlossenen Zeitraumes (z.B. Monat) eines Kunden (z.B. Fa. Export G.m.b.H.)

$$X = \{x_i : i \in \{1, ..., n\}\}$$

$$x_i \in \{SV \times ZV \times SZ \times D\}$$

$$SV, ZV \in \{0, 1, 2, ..., 9\}^l$$

$$SZ \in \{00:00, 00:01, ..., 23:59\}$$

$$D \in N$$

[1] Die Einheit der Gesprächsdauer ist Sekunden (sec). Wird die Dauer bei einem Provider in Takten gemessen (oder in anderer Form), so werden diese auf Sekunden umgerechnet.

$$CDR \subseteq X$$

Es ist leicht einzusehen, daß die Identifizierung des Start- und Zielortes eines Gesprächs in eindeutiger Weise über die entsprechenden Vorwahlkennungen erfolgen kann. Diese beiden Angaben sind somit ausreichend, um die, für die Kosten wesentliche, Ortszone des Gesprächs zu ermitteln.

P ... Menge der Provider

p ... Anzahl aller gültigen Tarifmodelle

T_{p_i} ... Menge der Tarifmodelle von Provider $i \in \{1, ..., |P|\}$

Y ... Menge der gültigen Tarifmodelle

K ... normierte Gesprächskosten

u_i ... Fixkosten des i-ten Tarifmodells [2]

y_i ... normierte Gesprächskosten des i-ten Tarifmodells

$$Y = \{(y_i, u_i) : i \in \{1, ..., p\}\}$$

$$y_i : \{SV \times ZV \times SZ\} \rightarrow K$$

Ein Tarifmodell setzt sich aus der Funktion y_i ,die jedem Telefonat x_i *(die Dauer des Gesprächs spielt bei den relativen Kosten keine Rolle)* einen normierten Wert[3] zuordnet, und den gesprächsunabhängigen Kosten u_i zusammen.

$|Y| = \sum_{i=1}^{|P|} |T_{p_i}|$

Die Anzahl der insgesamt verfügbaren Tarifmodell ist natürlich die Summe der Anzahl der von jedem Provider angebotenen Modelle.

Die normierten Gesprächskosten (z.B. 3,43ATS/sec) mit den Fixkosten[4] bilden in Kombination ein Tarifmodell. Die relativen Kosten werden hier als Funktion aufgefaßt, wobei jeder Startvorwahl, Zielvorwahl und Startzeit in eindeutiger Weise Gesprächskosten (ATS/sec) zugeordnet werden. Das Problem, daß ein Gespräch über mehrere »Zeitzonen«[5] geführt werden kann, wird erst bei der Bestimmung der gesamten Gesprächskosten berücksichtigt. Tabelle **??** zeigt, wie die Funktion y_i dargestellt werden kann - bei einer vollständigen Darstellung wäre die *Tabelle* dreidimensional.

[2]Bei der Anmeldung muß dem Provider bekanntgegeben werden, mit welchem Modell die Gespräche abgerechnet werden sollen; z.B. Fun, Business, Start usw.

[3]Kosten (ATS/sec)

[4]Kosten die fällig werden, selbst wenn kein einziges Gespräch geführt worden ist

[5]Bei der Berechnung der Kosten spielt nicht nur die zu überbrückende Entfernung eine Rolle, sondern auch die Tageszeit zu der ein Gespräch geführt wird. Es wird angenommen, daß vom Anbieter, neben den Ortszonen (Inlandszone 1 usw.), auch eine entsprechende Einteilung in *Zeitzonen* erfolgt.

	Zeitzone (Startzeit)			
Ortszone (Vorwahl)	00:00	01:00	...	23:00
ZV_1	$K_{1,1}$	$K2$		$K_{1,24}$
ZV_2	$K_{2,1}$	$K_{2,2}$		$K_{2,24}$
...			...	
ZV_k	$K_{k,1}$	$K_{k,2}$		$K_{k,24}$

Tabelle 2: Eine Darstellungsform für y_i, unter der Annahme einer festen Startvorwahl.

2.1.1 Optimierungsfunktion

f ... Diese Funktion bestimmt den optimalen Providermix

N ... Menge der Tarifmodelle, die für die Optimierung berücksichtigt wird[6]

M ... Menge der Tarifmodelle, mit denen eine Minimierung der Gesprächskosten erreicht werden kann.

a ... eine naürliche Zahl, die festlegt, aus wievielen Tarifmodellen der »optimale Providermix« maximal bestehen soll [7] .

$$f : \{N, CDR, a\} \rightarrow M$$

$N \subseteq Y$
$M \subseteq N$
$1 \leq a \leq |N|$

Mit der obigen Definition der Funktion f, ist festgelegt mit welchen Eingabedaten der optimale Providermix bestimmt werden soll. Die interne Arbeitsweise (Algorithmus) der Funktion muß jedoch noch festgelegt werden.

Die Aufgabe der Funktion besteht darin, für vorhandene CDRs, einen Providermix mit minimalen Gesprächskosten zu bestimmen. Soll das Optimum (Minimum) der Gesprächskosten mittels *Brute Force* ermittelt werden, so müssen zu diesem Zweck *sehr viele* Möglichkeiten berücksichtigt werden. Tabelle **??** zeigt eine Auflistung *aller* möglichen Zuordnungen eines Gesprächs zu einem Tarifmodell:

Aus dem Minimum der p^m verschiedenen Kosten, kann man direkt den optimalen Providermix ablesen. *Da die Kostenfunktion beliebig ist, ist es nicht möglich davon auszugehen, daß sich der optimale Providermix aus allen verfügbaren Providern zusammensetzt!*

Werden die Kosten einer Providerkombination berechnet, so wird auf jeden Fall die Grundgebür des Anbieters hinzugerechnet, unabhängig davon ob ein Gespräch über diesen Provider geführt wird oder nicht.

[6]Welche Tarifmodelle für die Optimierung letztendlich verwendet werden, kann/muß bei der Optimierungsfunktion explizit angegeben werden.

[7]Diese Einschränkung soll, wenn es notwendig ist, die Rechenzeit für die Optimierung begrenzen.

K_1	x_1, y_1	x_2, y_1	x_3, y_1	...	x_m, y_1
K_2	x_1, y_2	x_2, y_1	x_3, y_1		x_m, y_1
K_3	x_1, y_1	x_2, y_2	x_3, y_1		x_m, y_1
K_4	x_1, y_1	x_2, y_1	x_3, y_2		x_m, y_1
...				...	
K_p	x_1, y_1	x_2, y_1	x_3, y_1		x_m, y_2
K_{p+1}	x_1, y_2	x_2, y_2	x_3, y_1		x_m, y_1
...				...	
K_{p^m}	x_1, y_p	x_2, y_p	x_3, y_p		x_m, y_p

Tabelle 3: Auflistung der Möglichkeiten, die jedem der m Gespräche, eine Providermix zuzuordnen.
In den Kosten K_i sind noch nicht die Fixkosten (Grundgebühr) der Provider enthalten. Um die *gesamten* Kosten zu bestimmen, müssen zu jedem K_i die Grundgebühren, aller für den i-ten Mix verwendeten Tarifmodelle, addiert werden.

Selbst wenn bei einem sehr großen CDRs ($m > 10^5$) - die Anzahl der Provider ist nur vorerst auf 30 [8] beschränkt - eine vorherige Aggregation der Daten vorgenommen wird, steigt die Anzahl der erforderlichen Rechenschritte sehr schnell über eine erträgliche (berechenbare) Grenze.

2.2 Algorithmus der Kostenfunktion

Nachdem die Form der Optimierungsfunktion festgelegt wurde, soll nun ein Algorithmus entwickelt werden, der dieser Form entspricht. Es wird zunächst von einer erschöpfenden Suche (Brute Force) ausgegangen. Das Ergebnis soll als Grundlage für die Fittnesfunktion des GA's dienen. Aus diesem Grund wird der Formulierung der Algorithmen entsprechend ausführlich gestaltet:

Zunächst werden die Kosten eines einzelnen Gespräches mit einem bestimmten Tarifmodell ermittelt.

```
/**
/*Berechnet die Kosten eines Gesprächs
/*mit einem bestimmten Tarifmodell
*/
FUNCTION call_cost(INT: GesprächNummer,
                   INT: TarifmodellNummer,
                   GESPRÄCH_ARRAY(n,m): cdr,
                   TARIFMODELL_ARRAY(k,l,i): modell)
         RETURN FLOAT IS
```

[8] Da einem Provider mehrere Tarifmodelle zugeordnet sein könne, multipliziert sich diese Zahl noch einmal.

```
BEGIN
  VORWAHL: StartVorwahl, ZielVorwahl;
  INT OrtsZone;
  INT ZeitZone;
  INT Dauer;
  FLOAT Kosten = 0;

  StartVorwahl = cdr(GesprächNummer, 1);
  ZielVorwahl  = cdr(GesprächNummer, 2);
  Dauer = cdr(GesprächNummer, 3);

  /*Ermittlung der kleinsten gemeinsamen Ortszone aus
    Startvorwahl und Zielvorwahl */
  OrtsZone = to_ortszone(StartVorwahl, ZielVorwahl);

  /*Im ARRAY modell sind die normierten Kosten aller
    verfügbaren Tarifmodelle gespeichert. */

Kosten = modell(TarifmodellNummer, OrtsZone, ZeitZone)
               * Dauer;

         RETURN kosten;
END call_cost;
```

Der Datentyp **TARIFMODELL_ARRAY** enthält alle verfügbaren Tarifmodelle. Mit dem ersten Index k werden die verschiedenen Tarifmodelle durchlaufen, der zweite Index l bestimmt die Ortszonen und i die Zeitzonen. Bei all diesen Operationen wird davon ausgeangen, daß die Tarifmodelle auf einheitliche Orts- und Zeitzonen umgerechnet wurden. Der Datentyp **GESPRÄCH_ARRAY** enthält eine gesamten CDR (Menge von Gesprächen).

```
/**
/*Berechnet die niedrigsten Kosten eines CDR
/*für die vorgegeben Menge von Tarifmodellen
*/
FUNCTION cdr_cost(FIXKOSTEN_VECTOR(i): VectorGrundKosten,
                  TARIFMODELL_ARRAY(k,l,i): modell,
                  GESPRÄCH_ARRAY(n,m): cdr,
                  TARIFMODELL_VECTOR(i): tarifmodelle)
         RETURN FLOAT IS
BEGIN
  FLOAT GrundKosten = 0;            //Grundgebühr
  FLOAT GesamtKosten = 0;
  FLOAT GesamtKostenTemp1 = 10E50;
  FLOAT GesamtKostenTemp2 = 10E50;

/*Die Grundkosten müssen für alle für die Optimierung
  verwendeten Tarifmodelle berücksichtigt werden.
  Die Indizes von FIXKOSTEN_VECTOR(i) und TARIFMODELL_ARRAY
  sind aufeinander abgestimmt, d.h. gleicher Index,
```

```
  gleiches Tarifmodell*/
FOR i IN 1..VectorGrundKosten'length LOOP
  GrundKosten = GrundKosten + VectorGrundKosten(i);
END LOOP;
GesamtKosten = GrundKosten;

/*Durchlaufen der einzelnen Gespräche*/
FOR i IN 1..cdr(1)'length LOOP
  /*Durchlaufen der (berücksichtigten) Tarifmodelle
   FOR j IN 1..tarifmodelle'length LOOP

GesamtKostenTemp1 = call_cost(i, tarifmodelle(j), cdr, modell);
     IF GesamtKostenTemp1 < GesamtKostenTemp2 THEN
       GesamtKostenTemp2 = GesamtKostenTemp1;
     END IF;
   END LOOP;
   GesamtKosten = GesamtKosten + GesamtKostenTemp2;
END LOOP;

RETURN GesamtKosten;
END cdr_cost;
```

Bei dieser Funktion wird davon ausgegangen, daß die gesamten Gesprächskosten genau dann minimiert werden, wenn jedes einzelne Gespräch mit dem billigsten Tarifmodell geführt wird. Der Parameter *tarifmodelle* enthält die Indizes der zu berücksichtigenden Tarifmodelle.

```
/**
/*Ermittelt den besten Providermix aus allen Tarifmodellen
*/
FUNCTION best_modell(INT: a) RETURN BESTMODELL_VECTOR IS
BEGIN
INT: n; //Anzahl aller Tarifmodelle
FLOAT: kosten = 10E50;
FLOAT: kostenTemp = 10E50;
FIXKOSTEN_VECTOR(i): VektorGrundKosten;
TARIFMODELL_ARRAY(k,l,i): modell;
GESPRÄCH_ARRAY(n,m): cdr;
/*dieser Vektor legt fest, welche Tarifmodelle für die
  Optimierung verwendet werden sollen*/
TARIFMODELL_VECTOR(i,k): tarifmodelle;
/*Enthält die Indizes des besten Tarifmodellmixes*/
BESTMODELL_VECTOR(i): best_mix;

n = modell(1)'length;

/*Ist a die maximale Anzahl der zu berücksichtigenden
  Tarifmodelle, so müssen alle i-elementigen (1<= i <=a)
  Teilmengen aller Tarifmodelle berücksichtigt werden*/
```

```
FOR i IN 1..a LOOP
  tarifmodelle = getSubSets(n,i);
  FOR j IN 1..tarifmodelle(1)'length LOOP
    kosten = cdr_cost(VektorGrundKosten,
                      modell, cdr, tarifmodelle(j));
    IF kosten < kostenTemp THEN
      best_mix = tarifmodelle(j);
      kostenTemp = kosten;
    END IF;
  END LOOP;
END LOOP;

RETURN best_mix;
END best_modell;
```

Die Funktion *best_ modell* verwendet die beiden vorher definierten Funktion um den besten Tarifmodellmix zu erstellen. Dieser Algorithmus berücksichtigt dabei alle Teilmengen aus der Menge der Tarifmodelle. Die hier verwendete Funktion getSubSets bestimmt alle i-elementigen Teilmengen einer n-elementigen Menge (dabei werden die Elemente der Menge von 1 bis n durchnummeriert), das Ergebnis wird in einem Array der Dimension $\left[\binom{n}{i}, i\right]$ gespeichert. Da die Umsetzung dieser Funktion nicht sofort einsichtig ist, wird die JAVA-Klasse angegeben, die diese Funktion enthält.

```
public class subset
{
/**
/*Auf die Menge aller k-elementigen Teilmengen
/*einer n-elementigen Menge wird eine natürliche
/*Ordnung definiert. Anhand dieser Ordnung, kann
/*ein Nachfolger ermittelt werden.
*/
private static int[] nachfolger(int[] vorgaenger, int n, int k)
{
int[] vektor_temp = new int[k];
vektor_temp = vorgaenger;
if(vorgaenger[0] < n)
{
  vektor_temp[0] = vorgaenger[0] + 1;
  return vektor_temp;
}
for(int i = 1; i < k; i++)
{
```

```
  if(vorgaenger[i] + 2 <= vorgaenger[i-1])
  {
    vektor_temp[i] = vorgaenger[i] + 1;
    for(int j = i; j > 0; j--)
    {
      vektor_temp[j-1] = vektor_temp[j] + 1;
    }
    return vektor_temp;
  }
}
return null;
}

private static int binom(int n, int k)
{
if(k == 1)
{
  return n;
}
if((k == 0) || (k == n))
{
  return 1;
}
return (binom(n-1,k-1) + binom(n-1,k));
}

public static int[][] getSubSets(int n, int k)
{
int[][] result = new int[binom(n, k)][k];
for(int i = 0; i < k; i++)
{
  result[0][i] = i + 1;
}
for(int i = 1; i < binom(n, k); i++)
{
  result[i] = nachfolger(result[i-1], n, k);
}
return result;
}

public static void main(String[] args)
{
}
}
```

3 Genetische Algorithmen

3.1 Ein kleiner biologische Überblick: Ursachen der Evolution

Die Frage nach den Faktoren, die die Evolution hervorrufen, ist im Prinzip von Darwin im Jahr 1859 beantwortet worden. Darwin läßt durch seine Selektionstheorie die Anpassung sekundär aus einer ungerichteten Variabilität entstehen, die - wie man heute weiß - auf Mutation zurückzuführen ist. Durch die selektive Bevorzugung werden bestimmte Varianten durch die Umweltfaktoren ausgelesen. Die Selektion des jeweils Geeigneten wirkt, wie Experimente zeigen, sehr schnell. Es ist keineswegs nötig, daß nur bestimmte Varianten leben bleiben und alle anderen sofort durch den Kampf ums Dasein vernichtet werden. Es genügt, wenn die Überlebensrate bzw. die durchschnittliche Nachkommenzahl bestimmter Varianten nur um ein geringeres höher ist als die anderer, um sich schließlich in der Population durchzusetzen.Durch Konkurenzzuchten im gleichen Milieu ist vielfach nachgewiesen, daß Erbstämme einer Art meistens verschiedene Vitalität besitzen, deren Ausmaß durch Umweltfaktoren beeinflußt wird. In der Natur sind fast alle Populationen aus zahlreichen erbverschiedenen Varianten zusammengesetzt, so daß die Selektion wirken kann. Je reichhaltiger an Erbanlagen eine Population ist, desto intensiver kann Selektion umbildend wirken.

Die moderne Evolotionstheorie unterscheidet sieben Evolutionsfaktoren:[9]

1. Mutation
2. Gen-Transfer
3. Rekombination
4. Selektion
5. Populationswellen
6. Isolation
7. Anmidation

Mutation und Gen-Transfer schaffen neues Erbmaterial. Mutationen sind durch verschiedene Umwelteinflüsse hervorgerufene Erbveränderungen, die meist negative Konsequenzen für den Organismus haben. Unter Gentransfer versteht man die Einfügung von neuem genetischen Material in einen Genotyp.[10]

Rekombination ist das Entstehen neuer Genotypen durch Neuordnung von Erbfaktorne. Im Sinne der Mendelschen Gesetze kommt es zur Neuverteilung der Chromosomen im Zuge des crossing over[11] besteht außerdem die Möglichkeit des interchromosomalen Austausches. Während die Bedeutung des Gen-Transfers für die Evolution noch nicht abgeschätzt werden kann, besteht über Mutation und Rekombination prinzipiell Klarheit: Sie sind die Quelle ständig neu entstehender Varianten, die sich in der Selektion bewähren können.

[9] Die erstem vier Faktoren werden bei genetischen Algorithmen verwendet.
[10] Gesamtheit der Erbfaktoren eines Lebewesens
[11] In der Bioinformatik ist der Begriff *Crossover* gebräuchlicher

Bei der Selektion handelt es sich um einen statistischen Vorgang, der auf der unterschiedlichen Eignung der Individuen basiert: Wer stärker als das Durchschnittsindividuum zur Fortpflanzung[12] kommt, sichert seinen Genen eine größere Verbreitung als diese.

Unter Populationswellen versteht man starke Bestandsschwankungen in einer Population. Die Geschwindigkeit, mit der sich genetische Veränderungen durchsetzen, ist in kleinen Populationen besonders hoch. Dieses als genetische Drift bezeichnete Phänomen ist ebenfalls aus der Resistenzentstehung bekannt: Eingriffe in den Massenwechsel von Schadorganismen mit Insektiziden führt zur starken Verringerung der Population und zu einem anschließenden Neuaufbau einer selektiven Restgruppe mit Resistenzfaktoren.

Kommt es zur Isolation besonderer Genotypen, so können sie zum Ausgangspunkt für abweichende Populationen werden.

Ähnlich ist die Situation bei der Annidation, bei der einzelne Varianten in einem bestimmten Lebensbereich besonders günstige Lebensverhältnisse finden.[**?**]

3.2 Einleitung

Bei Genetischen Algorithmen (GA) handelt es sich um eine Klasse von Algorithmen, die sich Optimierungsstrategien[13] von biologischen Systemen abschauen (siehe Kapitel **??**; Ursachen der Evolution), diese abstrahieren und auf die jeweilige Problemstellung anpassen.

So ergeben sich neue Ansätze für viele konventionell schwierig oder gar nicht handhabbare Optimierungsprobleme, die so in akzeptabler Zeit lösbar werden. Die Leistungsfähigkeit der GA's läßt sich durch folgende Überlegung verstehen: Jede parametrisierbare Optimierungsaufgabe, bei der jedem Parametersatz ein Wert zugewiesen werden kann, der für die Qualität der Optimierung steht ist mittels GA lösbar. Die Rechenzeiten können jedoch bei vielen Ausgangsdaten, vielen Parametern bzw. komplizierten Modellen Tage dauern; für viele Anwendungen zu lange. Optimierungsprobleme treten in nahezu allen Bereichen der Naturwissenschaften, Technik und Logistik auf. Allerdings ist die Bandbreite der Probleme sehr groß, und die Anzahl der mittlerweile existierenden Verfahren bzw. Algorithmen sehr groß und nahezu unüberschaubar.

Optimierungsprobleme können in drei Klassen eingeteilt werden:

1. Parameteroptimierung: Probleme bei denen bspw. Parameter eines mathematischen Modells zu optimieren sind.

2. Subset-Selection: Probleme, bei denen aus einer Menge eine Untermenge ausgewählt werden muß.

3. Kombinatorische Probleme: Probleme, bei denen die Anordnung von Elementen eine Rolle spielt. Ein gängiges Beispiel ist das sogenannte traveling salesman Problem[14] .

[12] Bei genetischen Algorithmen wird die »stärke« eines Individuums durch die Fitnessfunktion bestimmt. Wie sich dieser dann »fortpflanzt« wird später näher erklärt

[13] Optimierung kann man als den Versuch auffassen, Eigenschaften eines Systemes in geschickter Weise so zu verändern, daß eine gestellte Aufgabe besser oder optimal bewältigt oder sogar gelöst werden kann.

Parameter: Unter einem Parameter (=Variable) wird eine veränderliche Größe des betrachteten Systems verstanden, dessen Veränderung in irgendeiner Weise mit dem Ergebnis einer Qualitätsfunktion in Zusammenhang steht.

Qualitätsfunktion: Eine Qualitätsfunktion[15] ist ein Kriterium, das die qualitative Beurteilung des betrachteten Systemes in Abhängigkeit von den Parametern erlaubt.

Suchraum: Unter einem Suchraum wird der Raum verstanden, den die Parameter aufspannen.

Ein grundlegendes Problem, das mit der Auswahl geeigneter Optimierungsstrategien in engem Zusammenhang steht, ist die Größe des Suchraumes. Zur Veranschaulichung der Problematik wird die Größe anhand eines einfachen Beispiels gezeigt:

Es soll ein System mit zwei Parametern optimiert werden, wobei beide Parameter (a1; a2) im Bereich:

$$100 \geq a_n \geq 0,1\,; Schrittweit = 0,1$$

liegen. D.h. für jeden Parameter gibt es in diesem Genauigkeitsbereich 1000 mögliche Werte die er annehmen kann. Der Suchraum, der durch die beiden Parameter aufgespannt wird, umfaßt somit $1000 \times 1000 = 10^6$ Möglichkeiten. Sollte noch ein dritter Parameter mit denselben Eigenschaften dazukommen, so würde die Anzahl der Möglichkeiten um drei Zehnerpotenzen steigen.

Ausgehend von diesen Überlegungen erscheint es einsichtig, daß ein Durchprobieren aller möglichen Werte nur in sehr niedrig-dimensionalen Räumen möglich ist. Sobald die Anzahl der Dimensionen, oder die erforderliche Präzision größer wird, scheitern die Methoden aufgrund der wachsenden Rechenzeiten[16].

Bei einfachen Problemstellungen kann also die Brute-Force Methode durchaus eine angemessene Strategie sein, zumal die Garantie besteht, die optimale Lösung zu finden, bei größeren Systemen jedoch stößt dieser Ansatz sehr schnell an die Grenzen des berechenbaren, und andere Verfahren müssen angewandt werden.

3.3 Konventionelle Suchmethoden

3.3.1 Monte-Carlo -Methode : Zufalls-Suche

Eine Strategie, die zur Suche in von Parametern aufgespannten Suchräumen denkbar ist, ist eine Suche, die auf reinem Zufall basiert. Die Vorgangsweise ist eine sehr einfache:

Es werden ein gewisse Anzahl von Punkten zufällig im Suchraum ausgewählt und die »Höhe« (Funktionswert) an diesen Punkten gemessen. Der *beste* Wert

[14] Bei dieser Aufgabe soll die optimale Anordnung von Städten auf einer Reiseroute bestimmt werden. Als optimal könnte bspw. die kürzeste Strecke definiert werden, oder auch die Strecke, die die geringsten Kosten verursacht, etc. Probleme dieser Kategorie werden auch mit dem Begriff Sequencing (SEQ) bezeichnet.

[15] In weiterer Folge wird die Qualitätsfunktion auch als Fitnessfunktion bezeichnet.

[16] vgl. dazu die Problemstellung des Findens eines optimalen Tarifmodellmixes/Profidermixes; Seite **??**

wird ausgewählt und als Optimum verwendet. Es ist klar, daß die Wahrscheinlichkeit mit diesem Algorithmus ein globales Optimum zu finden äußerst gering ist, bzw. der Aufwand extrem hoch wird. Allerdings können solche Strategien verwendet werden, um die Rauhigkeit oder auch Steilheit in bestimmten Bereichen der Suchlandschaft abzuschätzen. Der Vorteil dieser Methode ist, daß die Suche nicht gefahr läuft nur ein lokales Optimum zu ermitteln.

3.3.2 Hill Climbing

Eine sofort einsichtige Strategie ein zumindest lokales Optimum in einem Suchraum zu finden, ist, sich von einem beliebig gewählten Punkt immer in Richtung der größten Steigung (zum lokalen Optimum) zu orientieren. So ist garantiert, zumindest das nächstliegende lokale Optimum zu finden. Hierzu wird der SIMPLEX-Algorithmus verwendet.

Der SIMPLEX Algorithmus verwendet die kleinste geschlossene geometrische Figur, die ausreicht um die Steigung in einem Gebiet zu ermitteln, eben den sogenannten SIMPLEX: Im Falle einer zwei-dimensionalen Suchlandschaft (= zwei Parameter) ist dies ein Dreieck, im drei-dimensionalen Fall eine Pyramide. Wird nun z.B. ein Dreieck in eine zwei-dimensionale Landschaft gelegt so kann man anhand der Höhe der drei Eckpunkte des Dreiecks leicht die annähernde Richtung des nächsten Gipfels abschätzen. Die Suche nach dem Maximum oder Minimum erfolgt prinzipiell nach dem folgenden Schema :

1. Ein initiales SIMPLEX (also im zwei-dimensionalen Fall ein Dreieck) wird an zufälliger Position in den Suchraum gelegt.
2. Die Höhe der Eckpunkte wird ermittelt.
3. Aus den Höhen der Eckpunkte kann nun die ungefähre Richtung des Optimums ermittelt werden. Ein neues SIMPLEX, das sich näher dem Optimum befindet wird ermittelt.
4. Der Eckpunkt mit der schlechtesten Bewertung wird entlang der Verbindungslinie der besten Punkte gespiegelt. Der schlechteste Punkt im SIMPLEX wird durch den neuen (gespiegelten) Punkt ersetzt, wodurch ein neues SIMPLEX entsteht.
5. Wiederholung ab Punkt 2. Sobald das Optimum erreicht ist, wird diese Prozedur abgebrochen.

Die Einschränkungen der Hill-Climbing Algorithmen stellen sich folgendermaßen dar:

- Es besteht keine Garantie ein globales Optimum zu finden, da immer nur das nächst gelegene Optimum gefunden wird.

Eine bessere Strategie ist,mehrere Suchläufe mit unterschiedlichen Ausgangspunkten durchzuführen. Wird immer dasselbe Optimum gefunden, so ist es wahrscheinlich , das globale Optimum gefunden zu haben. Sicherheit kann auch so nicht erzielt werden.

Trotzdem sind Hill-Climbing Strategien in vielen Fällen eine günstige Wahl:

- Wenn die Bestimmung einzelner Werte sehr aufwendig oder teuer ist,

- die Beschaffenheit der Suchlandschaft und ein günstiger Ausgangspunkt bekannt sind, bzw. die Suchlandschaft so einfach ist (nur ein Optimum), daß dieses Verfahren als ausreichend betrachtet wird,
- man sich sicher ist, bereits nahe am gesuchten (globalen) Optimum zu sein, und nur die letzten Schritte zum Optimum fehlen,
- das Suchen des globalen Optimums nicht möglich oder nicht notwendig ist (z.B. im Falle der Optimierung von Ausbeuten chemischer Prozesse, wo schon eine deutliche Verbesserung zum nächsten lokalen Optimum als Erfolg gewertet werden kann).

3.4 Biologische Mechanismen

Diese wurden bereits im einleitenden Kapitel **??** dargelegt, soll an dieser Stelle, jedoch noch kurz wiederholt werden.

- Selektion: Unter Selektion versteht man die Bewertung der Fähigkeiten eines Individuums in Wechselwirkung mit seiner Umwelt und der daraus resultierenden Fortpflanzung der am besten angepaßten Individuen. Mit anderen Worten, diejenigen Individuen, die durch ihre Eigenschaften über eine bessere Überlebensfähigkeit in der jeweiligen Umwelt verfügen, werden sich stärker vermehren und somit auch ihre genetische Information.
- Paarung, Crossover: Paarung und Fortpflanzung laufen immer unter Informationsaustausch zwischen den Chromosomen/Genen der Eltern-Individuen ab. Dieser Informationsaustausch findet in dem genetischen Mechanismus der Crossover genannt wird statt, wobei der Informationsträger in der Natur Makromoleküle, nämlich die *Desoxyribonucleinsäure* (DNS, DNA) sind. Unter dem Selektionsdruck sollten die fähigsten Individuen überleben.
- Mutation: Durch Mutation können neue Strukturen entstehen. Eigenschaften also, die noch nicht in der Erbsubstanz der Art zu finden sind! Aufgrund der Eigenschaften der Codierung der Erbinformation in einem Makromolekül, der DNS , kann es zu Fehlern beim Kopieren, durch Einwirkung von Strahlung, chemischen Substanzen, etc. kommen. Diese Fehler werden Mutationen genannt. Die meisten Mutationen werden von der Selektion wieder ausgemerzt, weil sie entweder keine entscheidende Verbesserung gebracht haben, oder sogar lebensunfähig sind. Einige mutierte Individuem können jedoch überleben und für neue (bisher unter Umständen noch nicht dagewesene) Eigenschaften der Art sorgen.

Nun stellt sich die Frage, wie man die evolutionäre Strategie verwenden kann um Optimierungsprobleme zu lösen. Evolution im biologischen Sinne bedeutet , daß die Eigenschaften einer Art sich so verändern (verbessern), daß diese Art über bessere Chancen in der Umwelt verfügt. Durch die Evolution wird letztlich die Erbinformation der Individuen verändert.

3.5 Codierung

Die Art der Repräsentation des Gens für die Umsetzung im Computer ist willkürlich. Dennoch sollte eine Codierung gewählt werden, auf deren Basis sich die aus der Genetik kopierten Strategien möglichst gut verwirklichen lassen. An Stelle des genetischen Codes im biologischen Sinne, tritt eine Binärzahl, d.h. man definiert das Computergen als eine Folge einer bestimmten Anzahl von 0 und 1. Mit diesem Gen werden dann, die Operationen Crossover, Mutation und Selektion durchgeführt. Da das binäre Gen einen realen Sachverhalt, also z.B. Variablen repräsentiert, ist die Länge dieses binären Gens abhängig von der Problemstellung. Selbstverständlich ist auch die Art der Codierung stark von der Art des Problemes abhängig.

NUM steht für numerical parameter estimation also Parameter Optimierung,

SUB steht für Subset-Selection (Probleme, bei denen aus einer Gruppe von Elementen ausgewählt werden soll)

SEQ steht für Sequencing Probleme (also Anordnungsprobleme, bzw. auch allgemeiner kombinatorische Probleme fallen in diese Gruppe).

3.5.1 NUM: Parameter Estimation Probleme

Im Modell liegen üblicherweise mehrere Parameter, im Definitionsbereich der reellen Zahlen vor, deren Optimum gefunden werden soll. Innerhalb des Algorithmus wird aber nur mit dem binären Gen gearbeitet. Das bedeutet, daß diese Parameter in irgendeiner eindeutigen Form in ein binäres Gen umgewandelt werden müssen. Bei ungünstiger Wahl der Codierung kann die Leistungsfähigkeit des Algorithmus vermindert werden.

Einfache Binäre Codierung: Die einfachste Möglichkeit der Codierung ist es, die Parameter zu natürlichen Zahlen und diese in Binärzahlen umzuwandeln und aneinanderzuhängen:

1. Zunächst muß die reelle Zahl, die durch den Parameter ausgedrückt wird, in eine natürliche Zahl umgeformt werden, das geschieht üblicherweise durch folgende Transformation: $P_t = int(10^n \bullet P)$. Die Kommastelle des reelen Parameters wird je nach gewünschter Genauigkeit nach rechts verschoben und dann auf eine Ganzzahl gerundet. Dieser transformierte Parameter ist aus den natürlichen Zahlen, kann also ohne weiteres in eine binäre Zahl umgerechnet werden.

   P_t ... transformierter Parameter

   P ... ursprünglicher Parameter

   n ... die zu berücksichtigenden Dezimalstellen

2. Die binären Parameter werden nun in einer Folge aneinandergereiht. Diese Reihe wird jetzt als Gen definiert, der Repräsentant des ganzen Parametersatzes.

Binäre Codierung mit Gray-Code: Die binäre Codierung ist jedoch nicht optimal, da jedem Bit ein anderes Gewicht zukommt. Ein Invertieren, also Umdrehen des niederwertigsten Bits verursacht eine Veränderung des Parameters um den Wert 1, des zweiten Bits um 2, der dritten um 4, des n-ten um 2^{n-1}. Das ist normalerweise nicht wünschenswert. So findet oft anstelle des einfachen binären Codes der Graycode Verwendung . Das Prinzip der Transformation bleibt allerdings das gleiche. Auch in den folgenden Operationen ändert sich nichts, das Gen liegt auch hier in Form von 0 und 1 vor. Selbstverständlich müssen die Parameter, bevor sie wieder in das Modell eingesetzt werden, rücktransformiert werden, d.h. innerhalb des GA wird nur mit den binären oder gray codierten Genen gearbeitet, extern - also in den Modellen natürlich mit den reellen Parametern!

3.5.2 SUB: Subset Selection

Unter Subset Selection versteht man das Problem, aus einer Menge von Elementen nach bestimmten Kriterien eine Untermenge auszuwählen. Auch hier sind verschiedene Codierungen möglich:

Variante 1: Die Anzahl der Bits im Bitstring ist so groß wie die Anzahl der Elemente der Menge aus der gewählt wird. Jede Stelle im Gen entspricht dem Vorhandensein (1) oder der Abwesenheit (0) des entsprechenden Elementes in der ausgewählten Menge. (01001001) beispielsweise bedeutet, daß das 2., das 5. und das 8. Element ausgewählt sind.

Variante 2: Die Anzahl der Elemente die ausgewählt werden sollen ist bekannt: Bezeichnen wir die Anzahl der Elemente die gewählt werden sollen mit n:

Das Gen besteht aus n Variablen. Jede dieser n Variablen codiert das Vorhandensein eines Elementes. Hätte das binäre Gen z.B. den Wert 001011101111, so bedeutet daß Element 1 (001), 3 (011), 5 (101) und 7 (111) ausgewählt sind, wenn binär codiert wird.

Das Problem bei Variante 1 ist, daß die Anzahl der gewählten Elemente mit einer sinnvollen Bewertungsfunktion begrenzt werden muß, der Nachteil der Variante 2 liegt klarerweise darin, daß die Anzahl der selektierten Elemente im vorhinein bestimmt werden muß und daß im Algorithmus verhindert werden sollte, daß zwei oder mehrere gleiche Elemente ausgewählt werden.

3.5.3 SEQ: Sequencing

Eine Implementierung von Sequencing-Problemen (also z.B. das traveling salesman Problem) könnte ähnlich angesetzt werden, wie die Variante 2 von SUB. Allerdings müßte natürlich im Algorithmus garantiert werden, daß es zu keinen Wiederholungen der Elemente kommt.

3.6 Algorithmus

3.6.1 Initialpopulation

Der erste Schritt einer GA-Optimierung besteht darin, eine Population zu generieren. Die Populationsgröße liegt im allgemeinen zwischen 50 und 1000 Individuen, wobei man unter einem Indiviuum ein einzelnes Gen (also einen Parametersatz) versteht. Die Werte der Ausgangspopulation werden in den meisten Fällen mit einem Zufallsgenerator bestimmt. Werden hierbei spezielle Werte gewählt,besteht die Gefahr, den Algorithmus zu beeinflussen.

3.6.2 Bewertung der Population (Selektion)

Der nächste Schritt ist die Bewertung der Population. Es werden also die Gene zu den Parametern rücktransformiert und in das jeweilige Modell eingesetzt. Je nach der Güte der Anpassung erhält jedes Gen eine Bewertung. Der numerische Wert dieser Bewertung wird auch als Fitness des Gens bezeichnet. An dieser Stelle folgt die Selektion, für die mehrere Implementationen zur Verfügung stehen:

- Die neue Population ersetzt die alte (Eltern-) Population vollständig
- Die n besten Individuen werden in die neue Generation übernommen
- Die Vererbung erfolgt proportional zur Fitness der Individuen, d.h. die Besten werden mehrfach vererbt, die schlechteren weniger oft, diejenigen, deren Fitness niedriger als der Durchschnitt ist, werden verworfen.

Weiters ist es möglich, die Alterung in die Selektion eingehen zu lassen. Problematisch bei vielen Implementationen ist die Tatsache, daß das besten Gen verloren gehen kann. So wird oft ein Seitenweg für das beste Individuum gewählt, indem es gespeichert, und unverändert in die neue Population wieder eingesetzt wird. Durch diese Vorgangsweise ist garantiert, daß die Fitness des jeweils besten Gens der Population im Laufe der Optimierung nur Ansteigen oder wenigstens gleich bleiben, jedenfalls aber nicht schlechter werden kann.

3.6.3 Paarung (Mating)

Bei der Paarung werden aus Individuen aus der Population Paare gebildet, die dann dem Prozeß des Crossover unterworfen werden. Die Paare werden üblicherweise zufällig ausgewählt. Die Anzahl der Paare die gebildet werden, können über den Parameter *probability of mating* (also Wahrscheinlichkeit, daß eine Paarung zweier Gene erfolgt) gesteuert werden.

3.6.4 Crossover

Unter Crossover versteht man einen Informationsaustausch zwischen den beiden gepaarten Individuen. Das entspricht dem Austausch von Bits zwischen den beiden Bitstrings. Die einfachste Implementation ist das sogenannte one-point Crossover. Hierbei wird eine Stelle im Bitstring gewählt (der Crossover Punkt), und ab hier werden die restlichen Bits der zu paarenden Gene ausgetauscht. Im allgemeinen werden two-point oder uniform Crossover verwendet. Two-point Crossover ist vergleichbar mit one-point crossover, nur mit dem Unterschied, daß das Gen eben nicht nur an einer Stelle, sondern an zwei Stellen geschnitten und Information getauscht wird. Die meist günstigste Variante ist das uniform-Crossover. Dabei besteht eine definierte Austauschwahrscheinlichkeit zwischen allen Bits der beiden Bitstrings. Das bedeutet, daß von links nach rechts mit einer bestimmten Wahrscheinlichkeit die Bits an gleicher Position ausgetauscht werden. Die Bedeutung des Crossover liegt darin, daß Individuen Information austauschen und somit mit einer gewissen Wahrscheinlichkeit das Tochterindividuum über bessere Eigenschaften verfügt als die Eltern.

3.6.5 Mutation

Der letzte Schritt des Algorithmus ist die Mutation:

An dieser Stelle werden mit einer bestimmten Wahrscheinlichkeit einzelne Bits in den Genen invertiert. Diese Wahrscheinlichkeit sollte eher niedrig angesetzt werden ,ansonsten besteht die Gefahr, daß zuviele gute Gene zerstört werden.

3.6.6 Abbruchkriterium

Die neu entstandene Population muß bewertet werden und der oben beschriebene Ablauf wird wiederholt: Paarung, Crossover, Mutation, Bewertung, ..., solange bis entweder ein Konvergenzkriterium erfüllt wird, oder eine bestimmte Anzahl von Generationen erreicht wird. Ein solcher Lauf sollte aber auch mehrmals wiederholt werden um Auskunft über die Qualität der Ergebnisse zu erhalten.

3.6.7 Bewertung

Als Bewertung versteht man das Einsetzen der Parameter in das jeweilige Modell und die Ermittlung eines Qualitätskriteriums. Die Anzahl der Bewertungen im Laufe einer Optimierung kann sehr rasch steigen. Die Zeit, die die Einzelbewertung dauert, sollte also ein Minimum sein.

3.7 Evolutionäre Algorithmen

Eine den genetischen Algorithmen sehr ähnliche Strategie ist die Klasse der evolutionären Algorithmen (EA). Der essentielle Unterschied zwischen EAs und GAs liegt in der Codierung. Während Parameter aus den reellen Zahlen im Falle von genetischen Algorithmen zu Bitstrings transformiert werden, werden bei evolutionären Algorithmen die Parameter direkt verwendet und bilden einen Vektor. Dieser Vektor ist nun die Ausgangsbasis aller weiteren Schritte, die ähnlich definiert sind wie bei GAs.

4 Sitzungsprotokolle

4.1 09.03.1999

Besprechungspunkte:

- Allgemeine Definition des Problems
- Literatursuche; nach möglicherweise bereits bestehender Lösungsansätze des Problems (Internet).
- Vergleich mit herkömmlichen Methoden
- Organisatorisches: erstellen von: Zusammenfassung, Literaturverzeichnis, Einleitung

4.2 16.03.1999

Besprechungspunkte:

- Zur Dokumentation:
  - Titelblatt: Thema, Gruppenmitglieder, E-Mail, Gruppennummer, Lehrveranstaltungsleiter
  - Protokoll jeder Sitzung
- Offene Fragen:
  - Was geschieht wenn Gespräch Zeitzone überschreitet
  - Wie können Start- und Zielort bestimmt werden
  - kleinste Zeiteinheit (Takte) eines Gesprächs
  - wie wird Tarifmodell repräsentiert
  - Definition eines CDRs
  - wie erfolgt die Aufnahme der Gründgebühr in das Tarifmodell

4.3 23.03.1999

Besprechungspunkte:

- Zur Dokumentation:
  - Noch bestehende Fehler korrigieren; Beschreibung der verwendeten Bezeichnungen der Gleichungen
- Zusammenfassung über GAs, EC, EP
- Konventionelle Optimierungsmethoden
- Fitnessfunktion

Literatur

[1] Entwicklungsprojekt Tarifmodell; Smart.com; DI Peter Werzl

[2] Kurzes Lehrbuch der Zoologie; Adolf Rmane, Volker Storch, Ulrich Welsch; Gustav Fischer Verlag, 6. Auflage (S 323)