universität wien

MAGISTERARBEIT

Titel der Magisterarbeit

„Laufzeitanalyse von Iterated Local Search und Simulated Annealing am Traveling Salesman Problem"

Verfasser

Reinhard Bazant, Bakk.

angestrebter akademischer Grad

Magister der Sozial- und Wirtschaftswissenschaften

(Mag.rer.soc.oec.)

Wien, im Mai 2010

Studienkennzahl lt. Studienblatt:	A 066 951
Studienrichtung lt. Studienblatt:	Magisterstudium Statistik
Betreuer:	a. o. Univ. -Prof. Dr. Walter Gutjahr

INHALTSVERZEICHNIS

1. EINLEITUNG

1.1. Motivation

Wie stellt man 150 Exemplare der regelmäßig erscheinenden Vereinszeitung eines floridsdorfer Fußballklubs im Großraum Jedlersdorf an ebenso viele Mitglieder und Gönner zu, wenn der Gesamtweg den man dabei zurücklegt möglichst kurz sein soll? Mit dieser Frage beschäftigt sich der Schatzmeister des UFK Schwemm wohl schon seit der Vereinsgründung im Jahre 1972. Nachdem die Zeitungszustellung im Laufschritt gelegentlich auch als Trainingseinheit verwendet wird, ist auch in meinen Überlegungen obige Fragestellung schon wesentlich länger Thema, als mir bewusst ist, dass sich schon wesentlich berühmtere Menschen ihren Kopf darüber zerbrochen haben, ohne jedoch einen Algorithmus gefunden zu haben, der das Problem in „akzeptabler" Zeit löst. Im Laufe des 20. Jahrhunderts stieg die Dimension der gelösten TSP von n=318 [Crowder & Padberg, 1980] über n=2392 [Padberg & Rinaldi,1987] bis hin zu 7397 [Applegate, Bixby, Chvatàl & Cook, 1994]
Die Praxisnähe in der Tourenplanung und der Logistik und die beinahe alltägliche Relevanz des Traveling-Salesman-Problems (TSP) ist somit bereits offensichtlich. Betrachtet man die Begriffe „Orte" und „Distanzen" in einem übertrageneren Sinn, lässt sich die Problemstellung auf viele andere Aufgaben aus Gebieten wie z.B. der Elektro- [Korte, 1988] oder Gentechnik adaptieren.
Fokus dieser Arbeit wird es nun nicht sein Algorithmen zu finden, die das TSP möglichst effizient lösen, sondern die Laufzeiten bereits bekannter *Heuristiken*, also Methoden die „gute", aber nicht notwendigerweise optimale Lösungen liefern, aus statistischer Sicht zu analysieren. Da Heuristiken, wie etwa die in dieser Arbeit zentralen *Iterated Local Search (ILS) oder Simulated Annealing (SA)*, häufig nichtdeterministisch sind und somit zufällige Laufzeiten haben, sind statistische Methoden etwa zur Ermittlung von Laufzeitverteilungen oder Prognosen erforderlich. Dafür habe ich eine *ILS*-Heuristik sowie *SA* mit ein paar Variationen, sowie eine Prozedur die zufällige TSP-Instanzen, also konkrete Problemstellungen, generiert, in C++ implementiert. Dieses Programm liefert statistisch gesprochen die Daten für die Analyse.

1.2. Erläuterung „Traveling Salesman Problem“ (TSP)

Formal betrachtet gibt es n Städte, deren paarweise Distanzen zueinander bekannt und in der Distanzmatrix $D=(d_{i,j})_{nxn}$ zusammengefasst sind. Man unterscheidet dabei zwischen symmetrischen ($d_{i,j}=d_{j,i} \; \forall \; i,j \in \{1,2,\ldots,n\}$) und asymmetrischen TSP ($\exists \; (i,j) : d_{i,j} \neq d_{j,i}$). Inhaltlich entspricht die Modellierung durch ein asymmetrisches TSP z.B. Einbahnstraßen, die in die Gegenrichtung erst umfahren werden müssten, oder Höhenunterschiede zwischen zwei Städten, die in eine Richtung einen höheren Benzinverbrauch zu Folge hätten.

TSP werden weiters in metrische oder nichtmetrische TSP unterteilt. Bei metrischen TSP erfüllen die Distanzen $d_{i,j}$ zwischen Ort i und Ort j bezüglich einer bestimmten Metrik die Dreiecksungleichung, es gilt also

$d_{i,j} \leq d_{i,k} + d_{k,j}$ für alle $i,j,k \in \{1,2,\ldots,n\}$, was bei nichtmetrischen TSP nicht zwangsläufig der Fall sein muss. Bei metrischen TSP ist also der direkte Weg von A nach B *immer* der kürzeste, während sich bei nicht-metrischen TSP ein Umweg über C lohnen *könnte*, wenn man von A nach B kommen will (wenn z.B. auf verschiedenen Wegen unterschiedlich schnelle Verkehrsmittel zur Verfügung stehen: der Flug Wien-London hat zeitlich gesprochen in etwa die selbe „Länge“ wie die Autofahrt Wien-Salzburg, die tatsächlich zurückgelegten Wegstrecken sind allerdings verschieden). Häufig verwendete Metriken sind etwa die Euklid’sche-, Manhattan- oder Maximusmetrik.

Der Handlunsgreisende beginnt seine Tour in einer der n Städte, $x_1 \in \{1,2,\ldots,n\}$, muss daraufhin alle anderen Städte besuchen und wieder zur Ausgangsstadt zurückkehren. Es gibt also (n-1)! mögliche Touren, die sich im falle des symmetrischen TSP noch einmal auf die Hälfte reduzieren, da zu jeder Tour wegen $d_{i,j}=d_{j,i}$ eine genau umgekehrte Tour mit gleicher Länge existiert. Jede solche Tour $x=(x_1,x_2,\ldots,x_n)$ ist also eine Permutation des Vektors (1,2,…,n). Um die Verwendung mathematischer Verfahren zur Lösung zu erleichtern, wird das TSP als Graph modelliert. Die Städte $i \in \{1,2,\ldots n\}$ stellen die Knoten des Graphen dar, die Wege zwischen den Städten i und j, $x_{i,j}$, werden durch Kanten repräsentiert.

Aufgrund dieser Struktur zählt das TSP zu den *kombinatorischen Optimierungsproblemen.* Die Anzahl der möglichen Touren bei endlichem n ist somit in jedem Fall ebenfalls endlich. Es wäre also möglich die optimale Tour durch *vollständige Enumeration* (Ermittlung der Tourlänge für alle (n-1)!/2 mögliche Touren) zu finden, was theoretisch zwar möglich, praktisch jedoch wegen des

explosionsartigen Wachstums von n! schon für bereits noch nicht einmal groß wirkende Probleminstanzen nicht sinnvoll und effizient ist. Würde die Ermittlung der Länge *einer* bestimmten Tour eine Millisekunde Rechenzeit in Anspruch nehmen, würde sich die Laufzeit bei vollständiger Enumeration wie folgt verhalten:

n	n!	Laufzeit
1	1	0,001 sek
2	2	0,002 sek
3	6	0,006 sek
4	24	0,024 sek
5	120	0,120 sek
6	720	0,720 sek
7	5040	5,040 sek
8	40320	40,32 sek
9	362880	6,048 min
10	3628800	1,48 std
11	39916800	11,088 std
12	479001600	5,544 Tage
13	6227020800	72,072 Tage
14	87178291200	2,764 Jahre
15	1307674368000	41,47 Jahre

Tabelle 1: Theoretische Laufzeiten für vollständige Enumeration

Notation:

Die Städte werden in der Menge V (bzw. V_n falls im konkreten Kontext die Dimension von Bedeutung sein sollte) zusammengefasst.

Menge aller möglichen Touren wird in der Folge als T bezeichnet.

L: T→R ist die Zielfunktion, sie gibt für jede Tour xϵT ihre reellwertige Länge zurück.

Ein TSP mit n Städten wird im Folgenden als TSP(n) bezeichnet.

1.3. Problemklassen

In der Komplexitätstheorie werden Probleme anhand ihres Lösungsaufwands klassifiziert. Ein Algorithmus A heißt polynomial, wenn es ein Polynom p(n) gibt, das die Laufzeit $T_A(n)$ in Abhängigkeit von der Eingabegröße n dominiert.

Die komplexitätsklasse P umfasst all jene Probleme, für die es einen polynomialen Algorithmus gibt, also alle „polynomial lösbaren" Probleme.

Die komplexitätsklasse NP umfasst alle Probleme, für die es möglich ist die Optimalität einer Lösung durch einen polynomialen Algorithmus zu überprüfen.

Ein Problem p heißt NP-vollständig, wenn *alle* Probleme aus NP auf p rückführbar sind, d.h. wenn sie in polynomialer Zeit auf p umformuliert werden können, sodass aus der Lösung wieder in polynomialer Zeit eine Lösung des ursprünglichen Problems berechnet werden kann [Cook, 1971].
Da die Lösung jedes Problems, das polynomial lösbar ist, natürlich auch polynomial überprüft werden kann, umfasst die Klasse NP die Klasse P. Ob gleichzeitig auch P die Klasse NP umfasst und somit P=NP gelten würde, ist nicht bekannt. Es würde dafür genügen ein NP-vollständiges Problem in polynomialer Laufzeit zu lösen, da dann alle Probleme aus NP auf dieses polynomial zurückgeführt und somit auch mit polynomialem Aufwand gelöst werden könnten.
Das TSP gehört ebenfalls zu den NP-vollständigen Problemen [Garey & Johnson, 1979], es ist also kein Algorithmus bekannt, der es in polynomialer Laufzeit exakt löst.

1.4. Lösungsverfahren

1.4.1. Exakt

Aufgrund seiner NP-Vollständigkeit gibt es also keinen polynomialen Algorithmus, der das TSP exakt löst. Neben dem bereits erwähnten Verfahren der vollständigen Enumeration, das praktisch jedoch nur auf kleine Instanzen angewendet werden kann, liefert eine Branch-and-Cut Strategie auch für größere Probleminstanzen exakte Lösungen auf zwar deutlich effizienterem Weg als vollständige Enumeration, jedoch nach wie vor nicht in gewünschter polynomialer Laufzeit.
Für Branch-and-Cut wird das TSP in ein lineares Programm umformuliert: Die Entscheidungen werden als Variable $x_{i,j} \in$ {0=Kante (i,j) kommt nicht in Tour vor,1=Kante (i,j) kommt in Tour vor} formuliert. Da diese Belegung alleine noch keine Tour im TSP Sinne definiert, müssen noch Ungleichungen eingeführt werden, die Bedingungen an die Entscheidungsvariablen stellen, um:

(1) zu Gewährleisten, dass zu jedem Knoten genau *eine* Kante hin und genau *eine* Kante wegführt:

$$\sum_{j \in V} x_{i,j} = 2$$

(2) zu verhindern, dass die Belegung der Entscheidungsvariablen einzelne, voneinander getrennte Zyklen modellieren.

$$\sum_{i \in S, j \notin S} x_{i,j} \geq 2 \qquad \forall (S \neq \{\}) \subset V$$

Der Hauptvorteil der Branch-and-Cut Methode ist außerdem, dass sie eine untere Schranke für die optimale Tour liefert [Held and Karp 1970,1971] und somit Aussagen über die Güte einer heuristischen Lösung getroffen werden kann.

1.4.2. Heuristisch

Heuristiken sind näherungsweise Lösungsverfahren, die aus beschränkten Ressourcen „gute“, jedoch nicht zwangsläufig optimale Lösungen liefern. Die „Güte“ der Lösung an sich kann üblicherweise dadurch nicht quantifiziert werden. Im Kontext des TSP unterscheidet man zwischen Eröffnungs- (Heuristiken die eine „gute Tour“ konstruieren) und Verbesserungsverfahren (Heuristiken die bereits vorhandene Touren verbessern).

- Eröffnungsverfahren:
  - *Nearest-Neighbor-Heuristik* (NNH): Diese Heuristik entspricht wohl am ehesten der menschlichen Intuition. Von jedem Ort aus wird immer der nächstgelegene Ort besucht, bis alle Orte erreicht wurden.
  - *Farthest-Neighbor-Heuristik* (FNH): Ist eng mit der NNH verwandt, jedoch wird hier jeweils nicht der Nächstgelegene, sondern der am weitesten entfernte Ort als nächstes besucht.
  - *Insertion-Heuristiken:* Hier wird von einer Kante ausgegangen. Danach werden nach unterschiedlichen Kriterien sukzessive weitere Knoten in die Tour aufgenommen.
  - *Minimum-Spanning-Tree-Heuristik (MSTH):* In der Graphentheorie ist ein MST ein Graph, der alle Knoten miteinander verbindet und minimale Länge besitzt. Solch ein MST wird für das TSP ermittelt. Danach wird durch Verdopplung der Kanten ein Zyklus erzeugt, der zu einer TSP-Tour reduziert werden kann.

❖ Verbesserungsverfahren

- *Variable Neighborhood Search (VNS):* Eine bereits vorhandene Tour wird verbessert, indem die Nachbarschaft k-*ten Grades* nach Verbesserungsrichtungen abgesucht wird. Die Nachbarschaft k-ten Grades einer Tour x, N^k_x, ist die Menge aller TSP-Touren, die durch Entfernen von k Kanten aus x entsteht, wenn diese durch andere k Kanten ersetzt werden. Jede Tour eines TSP der Dimension n besitzt daher n*(n-1)/2 Nachbarn 2. Grades.
  VNS wählt zufällig ein Element aus der Nachbarschaft von x aus und akzeptiert dieses, falls der daraus resultierende Zielfunktionswert besser ist, als der von x.

  1. Finde zufällige Startlösung x_0 und berechne ihre Länge $l_0=L(x_0)$.
     Setze $x:=x_0$ und $l:=l_0$
  2. Wiederhole bis Abbruch: Wähle zufälliges $z \epsilon N^k_x$. Ist $L(z)<x$, dann setze $x:=L(z)$ und $x:=z$

  Eine Variante von VNS ist *Variable Neighborhood Descent (VND)*. VND ist der Grundbaustein von ILS_r.
  Die VND-Heuristik arbeitet wie folgt:

  1. Finde zufällige Startlösung x_0 und berechne ihre Länge $l_0=L(x_0)$.
     Setze $x:=x_0$ und $l:=l_0$
  2. Falls $\min(\{ L(z) \mid z \epsilon N^k_x \}) < l$ dann setze $x:= \arg\_\min(\{ L(z) \mid z \epsilon N^k_x \})$ und $l := \min(\{ L(z) \mid z \epsilon N^k_x \})$ und wiederhole 2
  3. Lokales Minimum x gefunden

Diese Einzelheuristiken können metaheuristisch kombiniert werden. Verbesserungsverfahren können beispielsweise Startlösungen verbessern, die Eröffnungsverfahren geliefert haben.
Wie allerdings am schematischen Algorithmus für die VND zu erkennen war, bricht diese ab, sobald ein lokales (nichtnotwendigerweise aber globales) Optimum gefunden wurde. Sie spürt also jenes lokale Optimum auf, das von der (möglicherweise zufälligen) Startlösung zwangsläufig erreicht wird. Die Selbe

Startlösung liefert somit in jedem Fall immer wieder dasselbe lokale Optimum. Diese Arbeit wird sich mit Varianten von VNS mit k=2 beschäftigen, also Nachbarschaftssuchen, die die Nachbarschaft 2. Grades untersuchen.

Um diesem Sog eines lokalen Minimums zu entgehen gibt es nun einige metaheuristische Strategien:

- *Simulated Annealing (simuliertes Abkühlen, SA)*[1]*:*
  Die Motivation für simulated Annealing stammt aus der Physik und wird von der Natur geliefert. Beim langsamen Abkühlen eines erhitzten Metalls haben die Atome genügend Zeit sich anzuordnen und so stabile Strukturen zu bilden. Simulated Annealing in Verbindung mit einem Local-Search-Algorithmus (wie etwa VNS) bedeutet nun, dass nicht zwangsläufig *immer* eine Verbesserungsrichtung eingeschlagen wird, sondern dass mit einer gewissen Wahrscheinlichkeit (die im physikalischen Kontext der Temperatur entspricht) auch eine Tour aus der Nachbarschaft gewählt wird, die eine Verschlechterung der Gesamttourlänge zur Folge hat. Dadurch kann der Algorithmus ein lokales Minimum (in dem es definitionsgemäß ausschließlich Verschlechterungsrichtungen gibt) auch wieder verlassen. Das „Abkühlen“ aus der Physik entspricht in der Informatik einer monoton fallenden Temperaturfolge, die bewirkt, dass die Wahrscheinlichkeit eine Verschlechterungsrichtung einzuschlagen (also das Unruhepotenzial der lokalen Suche) mit Fortdauer kleiner wird.
  Das TSP war eines der ersten Optimierungsprobleme, auf das SA angewandt wurde. ([Kirkpatrick et al. 1983] und [Cerny 1985])

- Iterated local search (ILS)
  Bei der iterierten lokalen Suche, entgeht die Heuristik dem Steckenbleiben in einem lokalen Minimum dadurch, dass die lokale Suche nach Auffinden eines lokalen Optimums neu gestartet wird, so lange, bis ein festgelegtes Abbruchkriterium erfüllt ist.

[1] siehe http://www-i1.informatik.rwth-aachen.de/~algorithmus/Algorithmen/algo41/algo41.pdf

- *Tabu-Suche:*

  Bei der Tabu-Suche [Glover 1986] wird das „Steckenblieben“ in einem lokalen Minimum dadurch verhindert, dass der bisherige Weg, den *Local Search* zur Auffindung eines lokalen Optimums eingeschlagen hat, tabuisiert wird und beim nächsten Versuch nicht noch einmal durchlaufen werden kann.
  Tabu-Suche umgeht in gewisser Weise also die „*Steepest-Descent*“ Idee der sturen Local Search Heursitik und wählt nicht zwangsläufig die *beste* Verbesserungsrichtung, sondern eine noch nicht tabuisierte. Die Tabu-Suche erlaubt *uphill-moves,* im Gegensatz zu SA, das *uphill moves* jederzeit mit einer bestimmten Wahrscheinlichkeit erlaubt, nur nach Erreichen eines lokalen Minimums und anschließender Tabuisierung des gegangenen Wegs.

2. PRÄPARIERUNG

2.1. Allgemeines

2.1.1. Zufall und Zeitmessung

In der Sprache C++ wurde die vollständige Enumeration, eine ILS_r-Heursitik sowie SA implementiert. Die wesentlichen und vor allem sprachspezifischen dabei verwendeten Elemente sind die Zufalls- und die Zeitmesskomponente:

- Zufall: Die Zufallsfunktion *rand()* aus der Library <stdlib.h>, mit der aktuellen Sytemzeit geseedet, lieferte dabei die zufällige Komponente: Einerseits bei der Generierung der simulierten TSP-Instanzen (metrisch oder nichtmetrisch), andererseits im Verlauf der Heuristiken selbst, bei ILS bei der zufälligen Auswahl einer Startlösung zu Beginn bzw. nach Auffinden eines lokalen Minimums und bei SA zum simulieren der Wahrscheinlichkeiten.
- Zeitmessung: Die Funktion clock() aus der library <time.h> zählt die CPU-Ticks vom Programmstart weg. Diese werden in Variablen vom Typ clock_t gespeichert. Durch Subtraktion der durch die zu Beginn und am Ende der Heuristik gesetzten clock()-Aufrufe gezählten CPU-Ticks kann mit Hilfe der Konstante CLOCKS_PER_SEC die Laufzeit in Sekunden ermittelt werden.

Die vom Programm simulierten Laufzeiten, die ermittelten, approximativen Löungswerte sowie die durch Erfolgsanteile geschätzten Wahrscheinlichkeiten werden in .txt-Dateien geschrieben und von der Statistiksoftware R eingelesen und ausgewertet.

2.1.2. Die „Güte“ des Zufalls

Nun soll kurz getestet werden, wie vertrauenswürdig die Zufallsfunktion rand() arbeitet, wenn als *seed* die jeweils aktuelle Systemzeit verwendet wird. Erwünscht wäre, tatsächlich aus einer Uniform([0;1])-Verteilung ziehen zu können. Dazu schreibt das Programm ein Sample S_{c++} von n=10000 seiner generierten Zufallszahlen. Zusätzlich wird in R mit dem Befehl *runif(10000)* ein Sample S_R aus einer Uniform([0;1])-Verteilung gezogen. Der Mittelwerte und die Varianz aus S_{C++}

ergeben, verglichen mit den erwarteten, theoretischen Größen bereits einen ersten Aufschluss über die Struktur dieses Samples:

	geschätzt	theoretisch
Erwartungswert	0,4983577	0,5
Varianz:	0,08234998	0,083

Tabelle 2: Erwartungswert und Varianz der C++ Stichprobe

Die Hypothese H_0: $\mu=0,5$ kann durch einen t-Test zum Niveau $\alpha=0,05$ nicht gegen die Alternative H_1: $\mu \neq 0,5$ verworfen werden.

p-Wert	KI
0.5671	[0,4927326; 0,5039828]

Tabelle 3: t-Test auf Mittelwert=0,5

Die Varianz aus S_{C++} wird mit der Varianz aus S_R bezüglich der Hypothesen

$$H_0: \sigma_{C++}^2=\sigma_R^2$$
$$H_1: \sigma_{C++}^2 \neq \sigma_R^2$$

zum Niveau $\alpha=0,05$ mittels „F-Test auf Gleichheit der Varianzen“ getestet:

Varianzquotient	p-Wert	KI
1,007	0,7417	[0,9679142; 1,0468599]

Tabelle 4: Test auf Gleichheit der Varianzen

Bei einem geschätzten Varianzquotienten von 1,007 ist dieser Unterschied bei einem p=0,7417 nicht 0,05-signifikant.

Passende Mittelwerte und Varianzen alleine sind natürlich noch kein Indiz, dass S_{C++} tatsächlich auch der gewünschten Uniform([0;1])-Verteilung folgt. Um die Verteilung von S_{C++} zu visualisieren bedienen wir uns zunächst eines Histogramms:

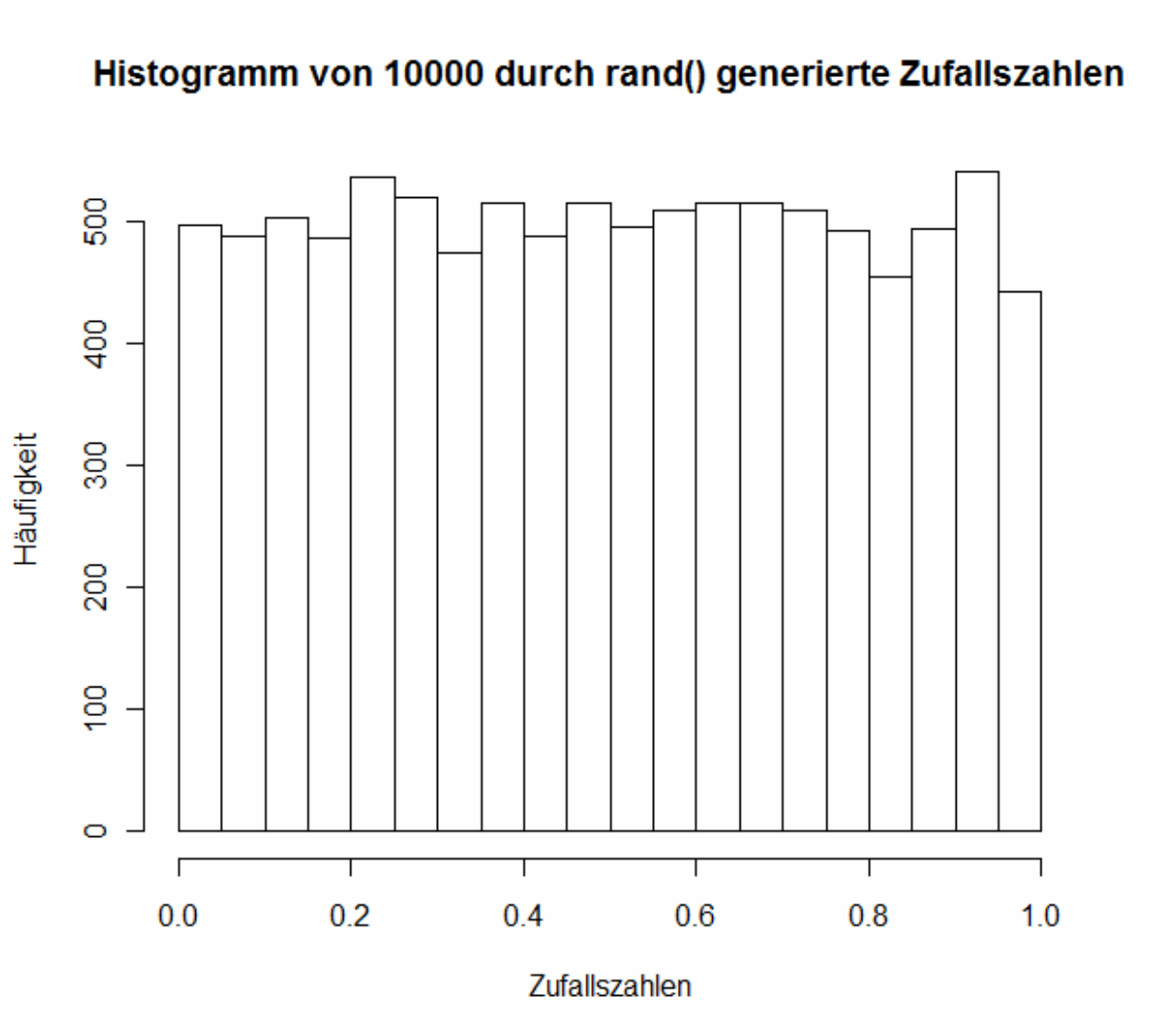

Abbildung 1: Histogramm der C++ Stichprobe

Von diesem Ergebnis einigermaßen überzeugt blicken wir dann auf den Vergleich der Quantile von S_{C++} mit den theoretischen Uniform([0;1])-Quantilen:

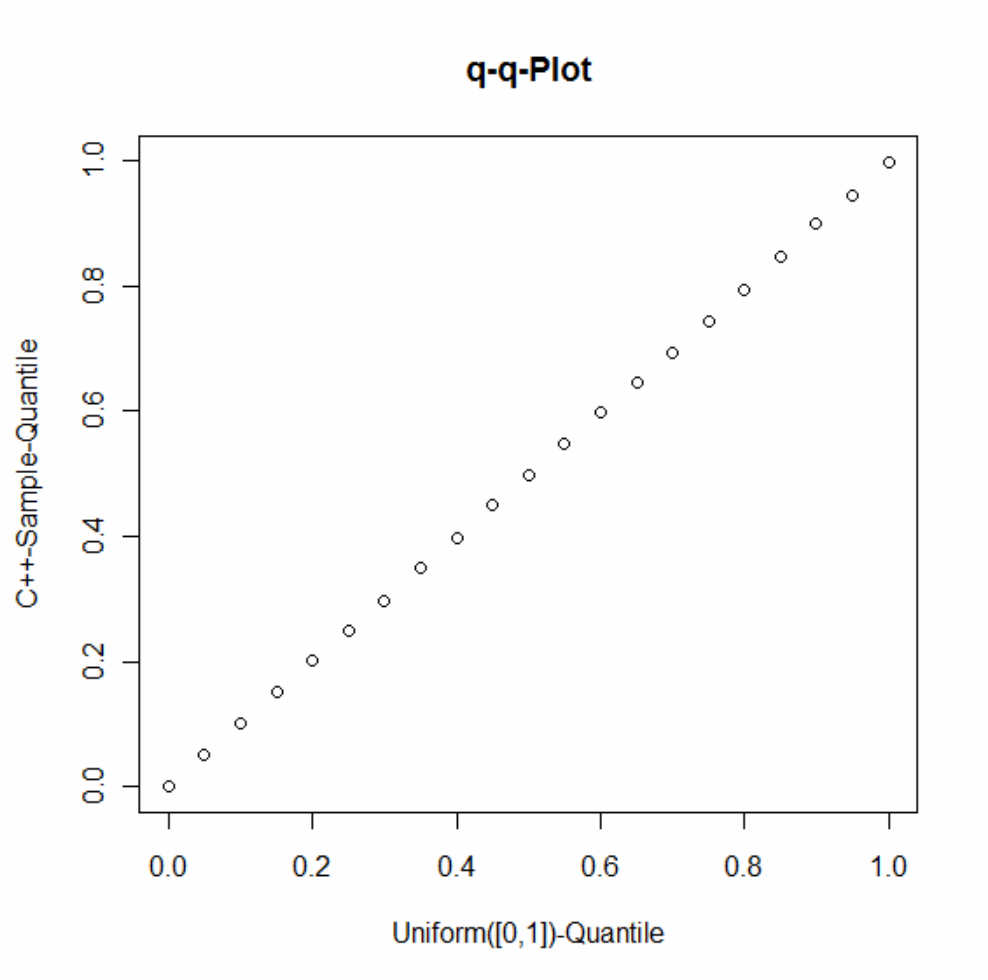

Abbildung 2: q-q-Plot der C++ Quantile gegen die theoretischen

Quantil(Unif)	Quantil(Daten)	Chi-Quadrat
0	0	0,0000000
0,05	0,051	0,0000200
0,1	0,101	0,0000100
0,15	0,152	0,0000267
0,2	0,202	0,0000200
0,25	0,25	0,0000000
0,3	0,297	0,0000300
0,35	0,35	0,0000000
0,4	0,398	0,0000100
0,45	0,45	0,0000000
0,5	0,498	0,0000080
0,55	0,548	0,0000073
0,6	0,598	0,0000067
0,65	0,646	0,0000246
0,7	0,6943	0,0000464
0,75	0,743	0,0000653
0,8	0,794	0,0000450
0,85	0,848	0,0000047
0,9	0,9	0,0000000
0,95	0,945	0,0000263
1	0,999	0,0000010

Tabelle 5: Theoretische vs. Beobachtete Quantile

Die rechte Spalte zeigt die χ^2-Testsummanden. Die Teststatistik C für den χ^2-Test ergibt sich durch:

$$C = \sum_{i=1}^{n} \frac{(q_{i_o} - q_{i_e})^2}{q_{i_e}}$$

wobei q_{i_o}, q_{i_e} die beobachteten bzw. erwarteten Quantile sind.

C ist dann χ^2 Verteilt mit n-1 Freiheitsgraden.

und beträgt:

C	1,000673981
df	20
p-Wert	1

Tabelle 6: Ergebnis des Chi-Quadrat-Tests

Auch der Kolmogorov-Smirnoff-Anpassungstest (K-S-Test) auf Gleichverteilung, der im Gegensatz zum gerade durchgeführten χ^2-Test nicht die Summe der gewichteten Abweichungen vom erwarteten Quantilswert, sondern die Maximalabweichung berücksichtigt, fällt bei einem p-Wert=0,4962 nicht 0,05-signifikant aus.
Die verwendete Zufallsfunktion ist also offensichtlich zulänglich.

2.1.3. Generierung der Distanzmatrizen

Das Programm erzeugt Instanzen aus 4 möglichen Klassen: symmetrische und asymmetrische, jeweils metrisch oder nichtmetrisch. Die Distanzmatrix $D=(d_{i,j})$ i,j=1,…,n wird wie folgt erzeugt:

- Nichtmetrische TSP-Instanzen: Hierbei werden die n Orte zufällig auf einen Kreis mit Durchmesser 1 verstreut. Die Distanzen liegen also alle im Intervall [0,1] und folgen nicht der Dreiecksungleichung. Für die symmetrische Variante gilt zusätzlich $d_{i,j}=d_{j,i} \ \forall$ i,j. Die Distanzmatrix D wird also in den Eintragungen $i \neq j$ mit Uniform([0;1])-Realisierungen versehen. Die Diagonalelemente die die Distanz eines Ortes zu sich selbst repräsentieren, werden 0 gesetzt. Sie sind ohnehin für die Heuristik irrelevant.

```
for(int i=0;i<dim;i++)
{
   for(int j=i;j<dim;j++)
   {
            if(i!=j)
           {
                  distmat[i][j]=float(rand()%100)/100;
                  distmat[j][i]=distmat[i][j];
                  //für die asymmetrische Variante:
                  //distmat[j][i]=float(rand()%100)/100;
            }
            else
            {
                  distmat[i][j]=float(0);
            }
      }
   }
```

- ❖ Metrische TSP-Instanzen: Das Erfüllen der Dreiecksungleichung bezüglich einer bestimmten Metrik (in dieser Arbeit wird nur die Euklid`sche Distanz betrachtet) wird durch zweidimensionale Darstellung jedes Punktes erreicht. Die Normen der Vektoren zwischen Punkt $i=(x_i/y_i)$ und $j=(x_j/y_j)$ werden in $D_{i,j}$ geschrieben:

```
for(int i=0;i<dim;i++)
{
      x_koor[i]=float(rand()%100)/100;
      y_koor[i]=float(rand()%100)/100;
}
for(int i=0;i<dim;i++)
{
      for(int j=i;j<dim;j++)
      {
            distmat[i][j]=sqrt(pow((x_koor[i]-
            x_koor[j]),2)+pow((y_koor[i]-y_koor[j]),2));
           distmat[j][i]=distmat[i][j];
      }
}
```

2.2. Vollständige Enumeration

Jedes TSP(n) mit n<∞ hat auch einen endlichen Lösungsraum. Dieser Lösungsraum ist bei einem n-dimensionalen TSP eine (n-1)! elementige Menge, da am Ende wieder an den Ursprungsort zurückgekehrt wird und eben dieser Ausgangsort somit noch keine „echte“ Entscheidung darstellt.
Bei vollständiger Enumeration werden *alle* möglichen Permutationen berechnet und bezüglich der gegebenen Distanzmatrix ausgewertet. Die Lösung, der nicht zwangsläufig eindeutige globale Minimierer, ist jene Permutation, die die kürzeste Rundreise repräsentiert.

```
for(int i=0;i<dim;i++)
{
      vektor[i]=i+1;
}
do
{
      next_permutation(vektor,vektor+dim);
      wert=0;
      for(int n=0;n<(dim-1);n++)
      {
            dummy=distmat[vektor[n]-1][vektor[n+1]-1];
            wert=wert+dummy;
      }
      wert=wert+distmat[vektor[0]-1][vektor[dim-1]-1];
      dummy=0;
      if(wert<bestwert)
      {
            bestwert=wert;
            for(int j=0;j<dim;j++)
            {
                  bestvektor[j]=vektor[j];
            }
      }
      tmp=1;
      for(int p=0;p<(dim-1);p++)
      {
            if(vektor[p]<vektor[p+1]){tmp++;}
      }
}while(tmp!=dim);
```

Die verwendete Funktion next_permutation aus der Bibliothek <algorithm> bildet für einen übergebenen Vektor die lexikographisch (eine Zeichenkette a ist lexikographisch kleiner als eine gleich lange Zeichenkette b, wenn das erste Zeichen

von a, in dem sich die beiden Zeichenketten unterscheiden kleiner ist als das von b) nächste Permutation. So werden alle (n-1)! möglichen Permutationen in der do-Schleife durchgeprüft, bis die Abbruchbedingung erfüllt ist, die besagt, dass wieder die lexikographisch erste Permutation, mit der gestartet wurde, erreicht ist.
Die Variable bestwert und das Array bestvektor[dim] enthalten zum Abbruchzeitpunkt das Minimum bzw. den (nichtnotwendigerweise eindeutigen) Minimierer.

2.3. Die Iterated-Local-Search$_r$ Heuristik (ILS$_r$)

Die ILS$_r$ Heursitik ist eine wie in 1.4.2. beschriebene Variante der VNS mit k=2. Sie sucht die Nachbarschaft 2. Grades einer möglichen Lösung ab und wählt als nächste mögliche Lösung diejenige, die unter allen Nachbarn die größtmögliche Verbesserung zur Folge hat, solange bis es in der Nachbarschaft keine bessere Lösung gibt, dann ist ein lokales Minimum gefunden. Der Parameter r beschreibt die Suchtiefe, d.h. wie oft nach Auffinden eines neuen kleinsten Minimums die Heuristik wieder mit einer neuen zufälligen Startlösung begonnen wird. Im Fall der ILS$_r$ Heuristik wird so oft mit einer neuen zufälligen Startlösung begonnen, bis dadurch r mal kein besseres lokales Minimum als das bisher kleinste lokale Minimum gefunden werden konnte. Wird ein neues bestes lokales Minimum gefunden, wird wieder in jedem Fall zu mindest r mal neu gestartet:

```
while(erfolglos<r)
{
      for(int i=1;i<dim;i++)
      {
            vektor[0]=1;
            do
            {
                  schalter=1;
                  kandidat=(rand()%dim)+1;
                  for(int j=0;j<i;j++)
                  {
                        if(vektor[j]==kandidat)schalter=0;
                  }
                  if(schalter==1)vektor[i]=kandidat;
            }while(schalter==0);
            vektor[dim]=1;
      }
      dummy=0;
      wert=0;
      for(int n=0;n<dim;n++)
      {
           dummy=distmat[vektor[n]-1][vektor[n+1]-1];
           wert=wert+dummy;
```

```
        }
        dummy=0;
        float wertrec=1000000000;
        float bestwert=1000000000;
        int abbruch=0;
        while(abbruch==0)
        {
              int tauschi=-1,tauschj=-1;
              for(int i1=1;i1<dim;i1++)
              {
                    for(int j1=i1;j1<dim;j1++)
                    {
                          if(i1==j1-1)
                          {
                                wert1=wert - distmat[vektor[i1-1]-
                                1][vektor[i1]-1] - distmat[vektor[j1]-
                                1][vektor[j1+1]-1] + distmat[vektor[i1-1]-
                                1][vektor[j1]-1] + distmat[vektor[i1]-
                                1][vektor[j1+1]-1] - distmat[vektor[i1]-
                                1][vektor[j1]-1] + distmat[vektor[j1]-
                                1][vektor[i1]-1];
                          }
                          else
                          {
                                wert1=wert - distmat[vektor[i1-1]-
                                1][vektor[i1]-1] - distmat[vektor[i1]-
                                1][vektor[i1+1]-1] - distmat[vektor[j1-1]-
                                1][vektor[j1]-1] - distmat[vektor[j1]-
                                1][vektor[j1+1]-1] + distmat[vektor[i1-1]-
                                1][vektor[j1]-1] + distmat[vektor[j1]-
                                1][vektor[i1+1]-1] + distmat[vektor[i1]-
                                1][vektor[j1+1]-1] + distmat[vektor[j1-1]-
                                1][vektor[i1]-1];
                          }

                          if(wert1<wertrec)
                          {
                                wertrec=wert1;
                                tauschi=i1;
                                tauschj=j1;
                          }
                    }
              }
              if(wertrec+1<wert)
              {
                    tausch(vektor,tauschi,tauschj);
                    wert=wertrec;
              }
              else abbruch=1;
        }
        if(wert<bestwert)
        {
              bestwert=wert;
              erfolglos=0;
        }
        else erfolglos++;
}
```

Die while-Schleife zu Beginn wird solange durchlaufen, bis r mal kein besseres lokales Minimum gefunden wurde.
Der in grau gehaltene Block wählt daraufhin jedes Mal zu Beginn der Schleife, eine neue zufällige Startlösung *vektor* aus, danach wird der Wert des Startvektors ermittelt und in *wert* gespeichert. Außerdem werden *wertrec* und *bestwert* als groß genug initialisiert. Die Variable *abbruch* wird deklariert und als 0 definiert. Sie bestimmt wie lange das Kernelement, die VND (die darauffolgende while-Schleife) durchlaufen wird.
Im roten Block werden die Lösungswerte aller Nachbarn des Vektors *vektor* berechnet. Danach wird im blauen Teil in die beste Verbesserungsrichtung gegangen, sofern es eine gibt. Gibt es keine einzige, ist ein lokales Minimum erreicht.
Daraufhin wird im grünen Part dieses neu gefundene lokale Minimum mit dem bisherigen besten lokalen Minimum verglichen und entweder ersetzt, oder der *erfolglos*-Zähler, der die große while-Schleife steuert, wird inkrementiert.

2.4. Simulated Annealing (SA)

Die auf SA basierende Heuristik arbeitet auch mit dem Prinzip der Nachbarschaftssuche. Der wesentliche Unterschied zur ILS_r Heuristik besteht allerdings darin, dass die Suche bei SA nicht zwangsläufig in eine Verbesserungsrichtung geht, sondern mit einer bestimmten, mit der Schrittanzahl der Heuristik gegen 0 gehenden Wahrscheinlichkeit, auch vom (lokalen) Minimum weg. Was im ersten Moment vielleicht ineffizient wirkt hat den besonderen Sinn, dass verhindert wird, in einem lokalen Minimum „festzusitzen“. Die ILS_r Heuristik vermeidet dies indem sie, je nach vorher definierter Suchtiefe, an zufälliger Stelle neu gestartet wird.
Die SA-Heuristik zur approximativen Minimierung einer Zielfunktion $f: K \rightarrow R$ arbeitet wie folgt:

1. Wähle eine Folge $t_n \rightarrow 0$ und Verweildauern D_n. Wähle eine zufällige Starlösung $x \epsilon K$ und berechne f(x). Setze $n:=1$, $t:=t_n$ und $i:=1$.
2. Falls $i<D_n$ wähle zufälligen Nachbarn y von x, sonst setze $i:=1$, $n:=n+1$ und $t:=t_n$

3. Betrachte f(y). Falls $f(y) \leq f(x)$ setze x:=y, falls außerdem $f(y) < f(x_{best})$ setze x_{best}:=y. Falls f(y)>f(x) setze x:=y mit Wahrscheinlichkeit[1] $\exp(-\frac{f(y)-f(x)}{t})$.
4. Wurde x nicht ersetzt, setze i:=i+1, sonst setze i:=1, n:=n+1 und $t=t_n$.
5. Falls ¬ Abbruchkriterium, gehe zu 2.

Ein passendes Abbruchkriterium (z.B. t<s für ein s>0) sorgt dafür, dass die Heuristik mit einem bestimmten x_{best} endet. Die C++ Implementierung der Optimierungsheuristik mittels SA sieht wie folgt aus:

```
for(int i=1;i<dim;i++)
{
        vektor[0]=1;
        do
        {
             schalter=1;
             kandidat=(rand()%dim)+1;


             for(int j=0;j<i;j++)
             {
                     if(vektor[j]==kandidat)schalter=0;
             }

             if(schalter==1)vektor[i]=kandidat;
        }while(schalter==0);
        vektor[dim]=1;
}

tempcount=1;
dummy=0;
wert=0;
for(int n=0;n<dim;n++)
{
        dummy=distmat[vektor[n]-1][vektor[n+1]-1];
        wert=wert+dummy;
}
dummy=0;


do
{
      do
      {
            int safe;
            int i1=(rand()%(dim-1))+1;
            int j1=(rand()%(dim-1))+1;
```

[1] „Trotzwahrscheinlichkeit“

```
while(i1==j1)
{
            j1=(rand()%(dim-1))+1;
}

if(i1>j1)
{
       safe=i1;
       i1=j1;
       j1=safe;
       safe=0;
}

if(i1==j1-1)
{
      wert1=0;
      wert1=wert - distmat[vektor[i1-1]-1][vektor[i1]-1] -
      distmat[vektor[j1]-1][vektor[j1+1]-1] +
      distmat[vektor[i1-1]-1][vektor[j1]-1] +
      distmat[vektor[i1]-1][vektor[j1+1]-1] -
      distmat[vektor[i1]-1][vektor[j1]-1] +
      distmat[vektor[j1]-1][vektor[i1]-1];
}
else
{
      wert1=0;
      wert1=wert - distmat[vektor[i1-1]-1][vektor[i1]-1] -
      distmat[vektor[i1]-1][vektor[i1+1]-1] -
      distmat[vektor[j1-1]-1][vektor[j1]-1] -
      distmat[vektor[j1]-1][vektor[j1+1]-1] +
      distmat[vektor[i1-1]-1][vektor[j1]-1] +
      distmat[vektor[j1]-1][vektor[i1+1]-1] +
      distmat[vektor[i1]-1][vektor[j1+1]-1] +
      distmat[vektor[j1-1]-1][vektor[i1]-1];
}
if(wert1<=wert)
{
      t=t*f;
       tempcount=1;
       wert=wert1;
       tausch(vektor,dim,i1,j1);
       if(wert<=bestann)
       {
            bestann=wert;
            for(int i=0;i<dim;i++)
            {
                  bestvekann[i]=vektor[i];
            }
            tempcount=tempcount+1;

      }
}
else
{
      ws=exp(-(wert1-wert)/t)*1000;
      zufall=(rand()%1000);
      if(ws>=zufall)
      {
            t=t*f;
```

```
                        tempcount=1;
                        wert=wert1;
                        tausch(vektor,dim,i1,j1);}
                    else
                    {
                        tempcount=tempcount+1;
                    }
                }

        }while(tempcount<=TEMPDAUER);
        tempcount=1;
        t=t*DECREASEFAKTOR;
}while(t>0.00000001);
```

Im roten Teil wird ein zufälliger Startvektor generiert. Dies ist bei der SA-Unterstützten Heuristik im Gegensatz zu ILS_r nur ein einziges Mal zu Beginn erforderlich. Ohne Beschränkung der Allgemeinheit wird an Ort 1 gestartet und am Ende *wieder* an Ort 1 zurückgekehrt. Danach wird die Länge der Starttour ermittelt und in wert gespeichert.
Die darauf folgenden schwarzen do-Schleifen prüfen die Abbruchbedingung (Temperatur<voreingestellter Wert) und die Temperaturverweildauer.
Im Grünen Part werden zufällig zwei Orte i1 und j1 ausgewählt und es wird überprüft, ob ein 2-opt-Tausch (siehe Abschnitt 5.3.1.) dieser beiden Orte eine Verbesserungsrichtung zur Folge hätte oder nicht.
Im Blauen Bereich erfolgt die Selektion gemäß 3. der Heuristik. Die dabei verwendetet Funktion tausch() wird im Anhang, in Abschnitt 7.3.3 angeführt.

Notation: t steht für die Starttemperatur, d für die Verweildauer in jeder Temperaturstufe. SA_f (0<f<1) steht für SA-Heuristik mit Temperaturfolge $t_0=t$, $t_{n+1}=t_n*f$.

In dieser Arbeit geht es hauptsächlich, wenn nicht explizit Anderes erwähnt, um eine $SA_{0,5}$-Heuristik mit Abbruch bei t<0,00000001. Die Trotzwahrscheinlichkeit, also jene Wahrscheinlichkeit mit der in eine Verschlechterungsrichtung gegangen wird, ist dabei selbst für große Differenzen in den Funktionswerten bereits so klein, dass ab hier die Suche bereits beinahe exakt einer lokalen Suche entspricht.

2.5. Metrisch vs. Nichtmetrisch

2.5.1. Laufzeit

Nun wollen wir untersuchen, ob es sich überhaupt lohnt das Laufzeitverhalten der Heuristik für metrische bzw. nichtmetrische TSP getrennt zu untersuchen. Dazu wurden 8 Laufzeitstichproben S_1, S_2,…, S_8 der Größe N_i=100 (i=1,…,8) von metrischen bzw. nichtmetrischen TSP gezogen, die sich in Städteanzahl (n) und ihrer heuristischen Suchtiefe (r) unterscheiden.

Die Verteilungen scheinen sich auf den ersten Blick auf die Laufzeit-Histogramme weder für TSP(50):

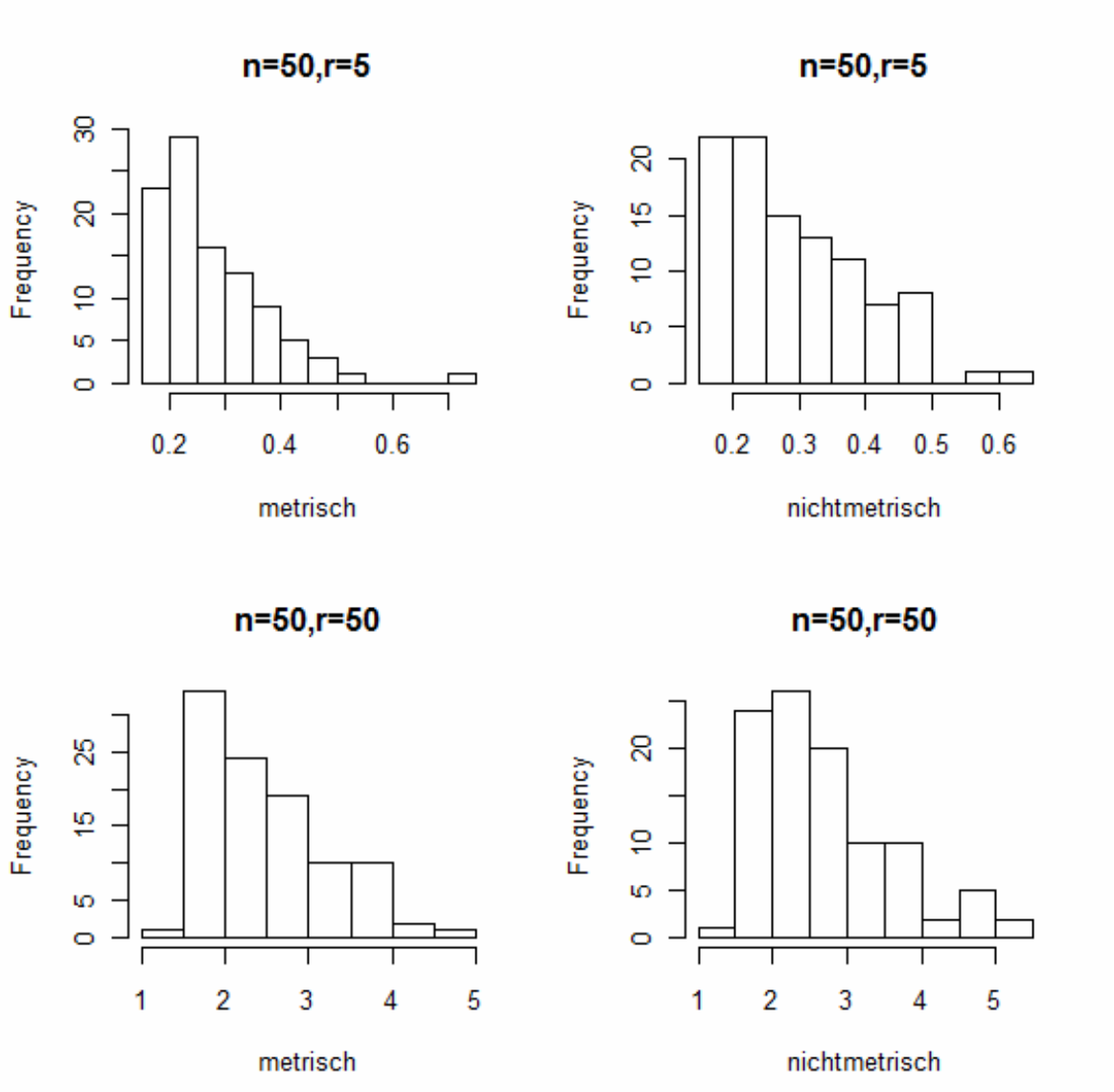

Abbildung 3: Histogramme der Laufzeiten für unterschiedliche r bei n=50

Noch in der Gruppe der TSP(80) stark zu unterscheiden:

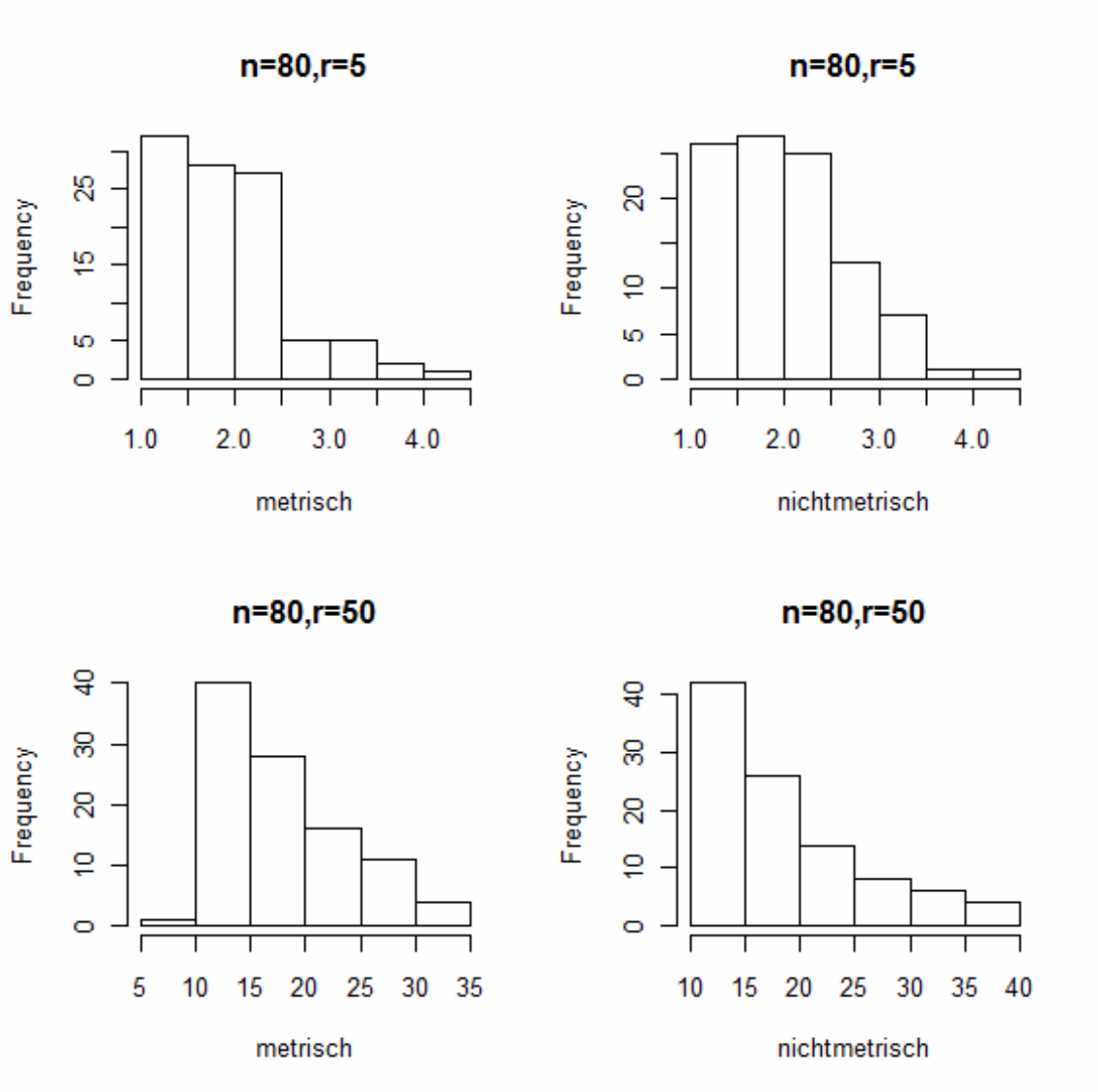

Abbildung 4: Histogramme der Laufzeiten für unterschiedliche r und n=80

Einzig die Rechtsschiefe ist unabhängig von n und r in allen Histogrammen deutlich zu erkennen. Außerdem weisen fast alle Parameterkonstellationen noch eine zusätzliche kleine Massenansammlung links neben dem einzigen Modus der Verteilung auf.
Auch bezüglich ihrer Lage und Streuung scheinen die einzelnen Stichproben sich bezüglich des Faktors „Metrik“ nicht stark voneinander zu unterscheiden. Der Mittelwert der Laufzeiten ist bei allen gesampelten Parameterkonstellationen in der nichtmetrischen Gruppe leicht, jedoch nicht signifikant größer:

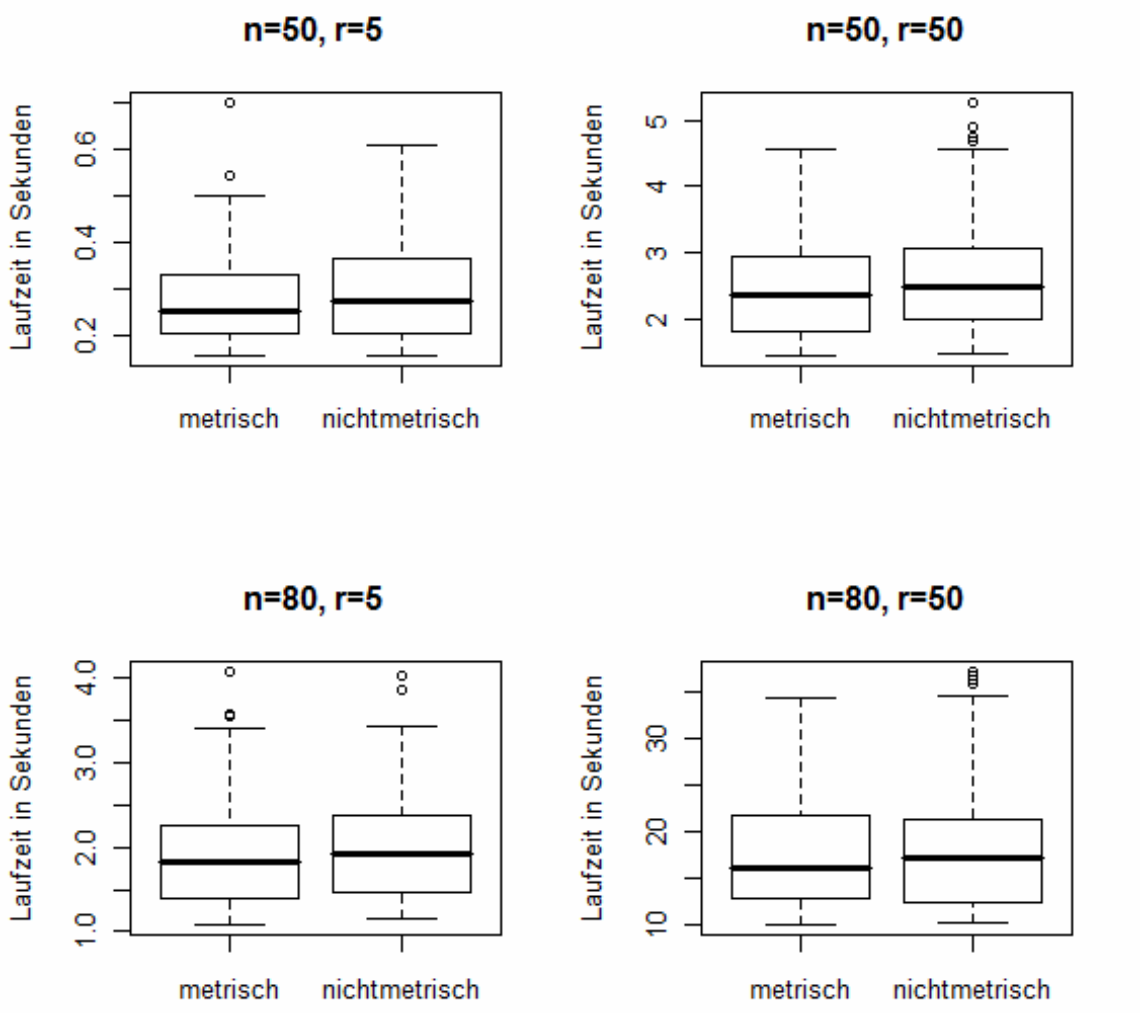

Abbildung 5: Boxplots der Laufzeiten

Diese Vermutung bestätigen auch der durchgeführte Zweistichproben t-Tests auf Gleichheit der Mittelwerte, der F-Tests auf Homogenität der Varianzen sowie der Kolmogorov-Smirnoff-Test (K-S-Test) auf Gleichheit der Verteilungen (jeweils bezüglich des Faktors „Metrik“), die den untersuchten Stichproben zugrunde liegen (Tabelle7)

	n=50, r=5	n=50,r=50	n=80,r=5	n=80,r=50
t-Test	0,146	0,076	0,238	0,4
F-Test	0,546	0,057	0,774	0,103
KS-Test	0,468	0,367	0,813	0,468

Tabelle 7: p-Werte der Tests auf Verteilungsgleichheit

Die Ergebnisse bezüglich der SA-Laufzeiten fielen bezüglich des Faktors „Metrik“ ebenfalls nichtsignifikant aus. Im Folgenden muss bezüglich der Laufzeiten dem Faktor „Metrik“ also keine Relevanz zugebilligt werden.

2.5.2. Lösungswerte

Betrachtet man nicht die Laufzeiten, sondern die approximativen Lösungswerte, die die beiden Heuristiken in Stichproben der Größe N=1000 zufälliger TSP-Instanzen ermittelt haben, so sieht man, dass der Faktor „Metrik“ Einfluss zu haben scheint. Eigentlich nicht weiter verwunderlich, da Orte in einem metrischen TSP die Dreiecksungleichung erfüllen müssen, was durch die Bauart der Distanzmatrizen (Abschnitt 2.1.3.) größere Distanzen „erzwingt“. Der Faktor „Metrik“ wirkt sich somit nicht auf die Performance der Heuristiken aus, er vergrößert lediglich die zufällig generierten Distanzen (siehe Abschnitt 2.1.3.)

	ILS	SA
nichtmetrisch	2,9821	9,2611
metrisch	7,2443	13,1661

Tabelle 8: Werte: nichtmetrisch vs. metrisch

Die Unterschiede in den Werten sind bezüglich t-Tests α=0,05-signifikant. Sind also die Funktionswerte von Bedeutung (z.B. beim Performancevergleich in Kapitel 5) ist der Faktor „Metrik“ von Relevanz. Allerdings scheint er auf die beiden Heuristiken einen konstanten Einfluss zu haben und ist somit wieder vernachlässigbar.

2.6. Symmetrisch vs. nichtsymmetrisch

2.6.1. Laufzeiten

Es wurden Stichproben der Größe N=1000 bei konstanter Parametrisierung (n=80, r=5, d=300, t=1) simuliert. Tabelle 9 zeigt die Laufzeitmittelwerte. Der Faktor „Symmetrie“ beeinflusst offensichtlich die Laufzeit der ILS Heuristik, während SA unabhängig von der Symmetrie zu arbeiten scheint.

	ILS	SA
symmetrisch	1,9800	0,0185
asymmetrisch	0,7766	0,0186

Tabelle 9: Laufzeiten: symmetrisch vs. nichtsymmetrisch

Der Unterschied bei ILS ist bezüglich t-Tests 0,05-signifikant während der Unterschied bei SA erwartungsgemäß nichtsignifikant ausfiel.

2.6.2. Lösungswerte

Stichproben derselben Bauart wie in 2.6.1. lieferten Lösungswertmittelwerte, die in Tabelle 10 dargestellt sind.

	2-opt	SA
symmetrisch	2,9821	9,2611
asymmetrisch	16,2460	9,2283

Tabelle 10: Werte: symmetrisch vs. nichtsymmetrisch

Diese sind wieder nur in der ILS Kategorie signifikant.

2.6.3. Zusammenfassung

Zusammenfassend kann über den Einfluss der beiden Faktoren „Metrik“ und „Symmetrie“ auf Laufzeit und Lösungswerthöhe der beiden Heuristiken gesagt werden:

		Metrik	Symmetrie
ILS	Laufzeiten	NEIN	JA
	Lösungswerte	JA	JA
SA	Laufzeiten	NEIN	NEIN
	Lösungswerte	JA	NEIN

Tabelle 11: Zusammenfassung des Einflusses von Metrik und Symmetrie

Wobei der Faktor „Metrik“ keinen echten Einfluss auf die Leistung der Heuristik hat, sondern lediglich durch Uniform([0;1])-verteilte x- und y-Koordinaten und die Dreiecksungleichung größere Tourlängen und somit auch größere optimale Tourlängen erzwingt!

3. AUSWERTUNG: ILS_r -HEURISTIK

3.1. ILS – Variante A

Die ILS-Variante A ist die in Abschnitt 2.3. beschriebene *Iterated Local Search* Heursitik mit Suchtiefeparameter r, der, wie beschrieben, einer stochastischen Anzahl an Iterationen entspricht, die vom bisherigen Erfolg der Heuristik abhängen.

3.1.0. Laufzeitverhalten von ILS_0

Bevor das Laufzeitverhalten der Heuristik untersucht wird, betrachten wir zunächst die Laufzeit ihres Kernelements, einer *einzelnen* lokalen Suche (siehe VND Abschnitt 1.4.2.). Sie kann als Spezialfall der ILS_r Heuristik mit r=0 betrachtet werden und wird als $X^{(n)}$ bezeichnet.

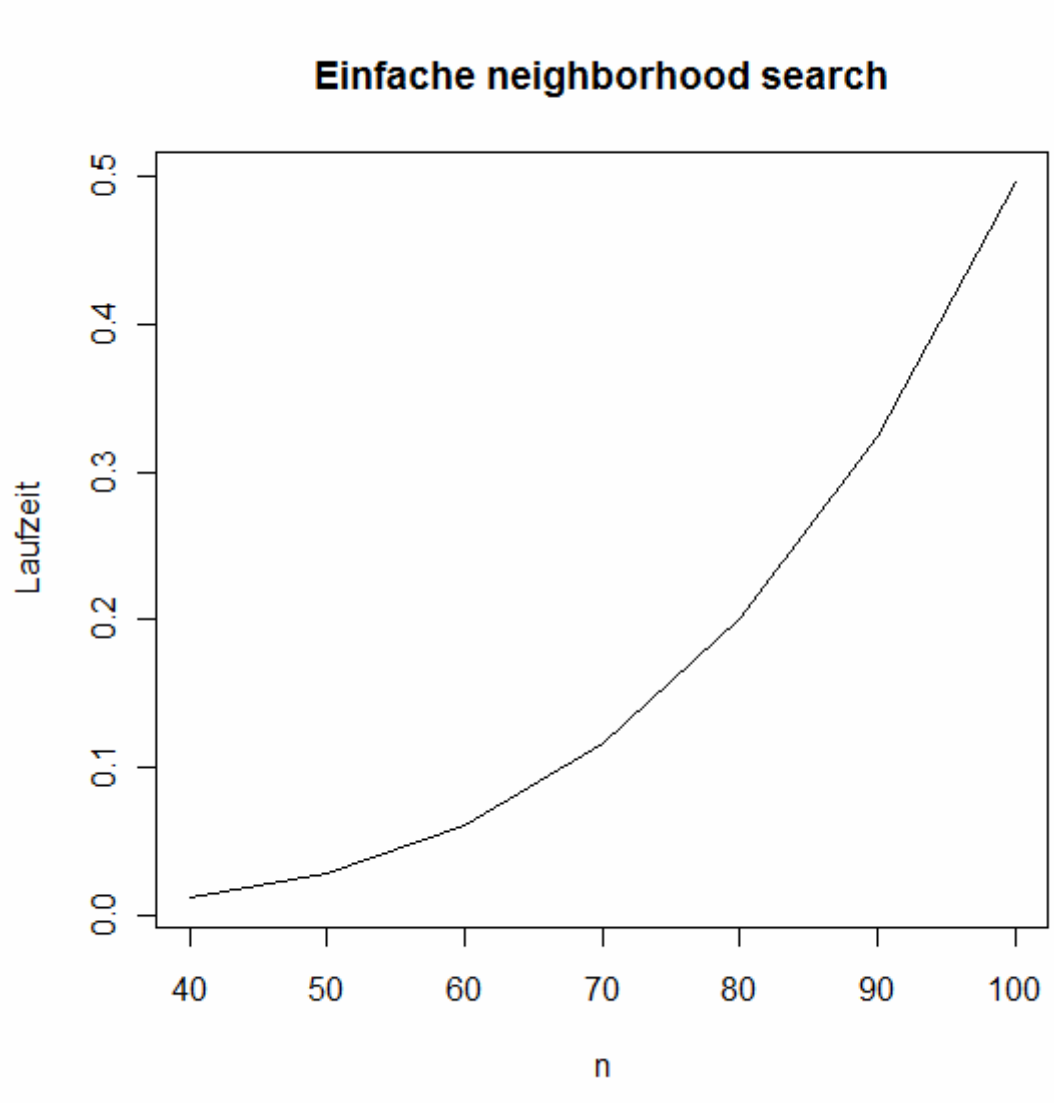

Abbildung 6: Laufzeit der einfachen VND (=ILS_0)

Die Unterstellung eines Modells, das die vierten Wurzeln der Laufzeiten linear mit der Instanzgröße n in Zusammenhang bringt liefert eine Anpassungsgüte von $R^2 = 0{,}9999747$. Die Modellparameter ergaben sich wie folgt:

$$\text{Laufzeit} = (-0{,}00998994 + 0{,}00851828 * n)^4$$

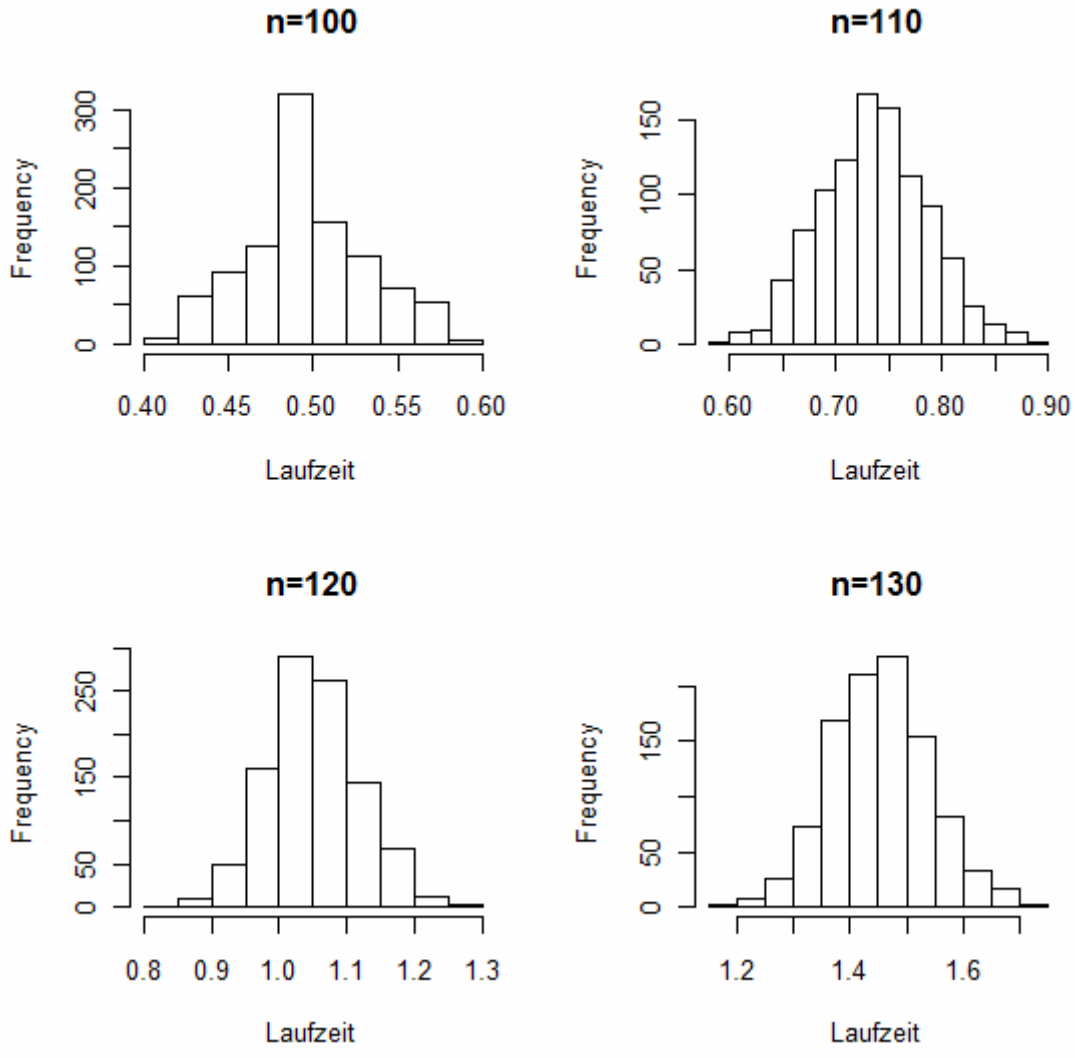

Abbildung 7: Histogramme der Laufzeiten von ILS_0

Die Laufzeiten scheinen dabei symmetrisch um diesen Mittelwert verteilt zu sein.

3.1.1. Die Verteilung der Laufzeiten der ILS_r Heuristik

Nach 2.5 ist es also nicht notwendig in Punkto Laufzeit der ILS_r Heuristik näher auf den Faktor „Metrik“ einzugehen. Obwohl sich die Laufzeiten bezüglich des Faktors „Symmetrie“ (nach Abschnitt 2.6.) in statistisch signifikanter Art und Weise unterscheiden, wird in diesem Kapitel bei der Modellierung der Laufzeiten nur auf symmetrische TSP-Instanzen eingegangen. Im Folgenden werden also ausschließlich Simulationen von metrischen, symmetrischen TSP zur Datenbeschaffung herangezogen.

Der Fokus dieses Unterkapitels liegt darauf die Verteilung der Laufzeiten der Heuristik zu betrachten und eine (möglicherweise bereits bekannte) Verteilungsfamilie zu finden, die die simulativ „gezogene“ Häufigkeitsverteilung möglichst gut trifft.

Sei $P^{(n)}$ die Zufallsvariable die die Anzahl der gestarteten lokalen Suchen bei TSP-Dimension n beschreibt. Dann kann die Laufzeit der Heuristik folgendermaßen betrachtet werden:

$$\text{Laufzeit} = P^{(n)} * X^{(n)}$$

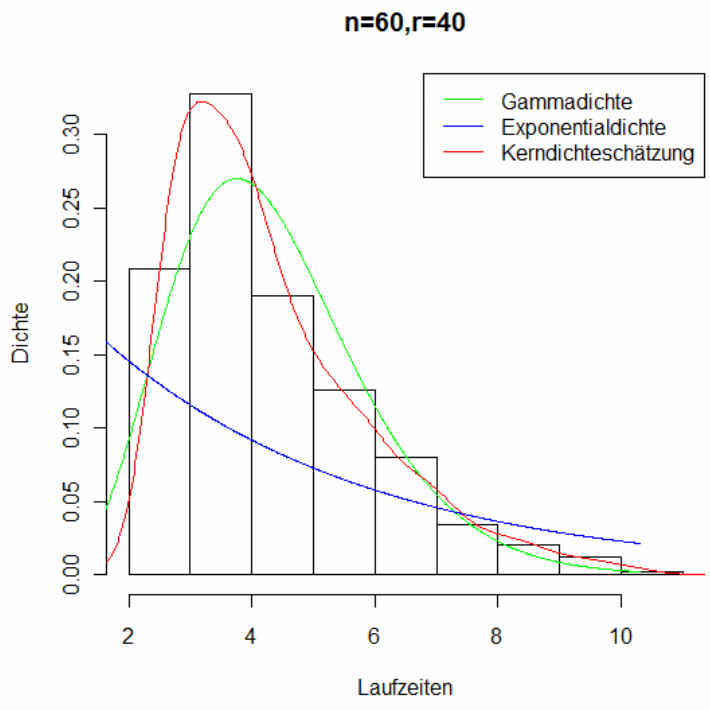

Abbildung 8: Histogramm der Laufzeiten und mögliche Dichten bei n=60

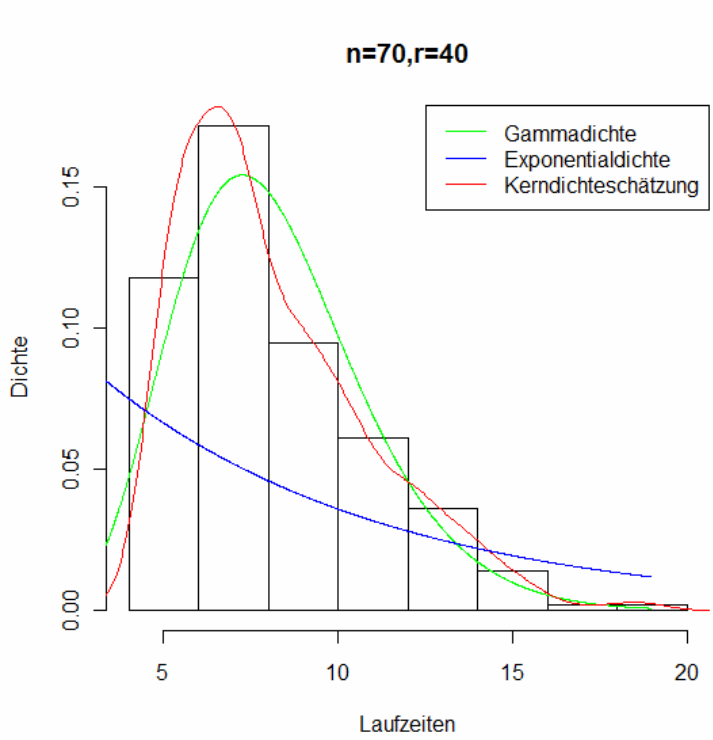

Abbildung 9: Histogramm der Laufzeiten und mögliche Dichten bei n=70

Abbildungen 8 und 9 zeigen die Verteilung der Laufzeiten für n=60,70 und einer heuristischen Suchtiefe von r=40 in beiden Fällen. Um die im Histogramm bereits relativ gut Erkennbare Form der Verteilung noch weiter zu verdeutlichen, zeigt die rote Linie die mittels Kerndichteschätzung geschätzte Dichte der Daten.
Diese unimodale, rechtsschiefe Struktur deutet auf eine Verteilung aus der Familie der Exponentialverteilungen hin. Die blaue und die grüne Linie sind die Dichtefunktionen der Exponential- bzw. Gammaverteilung mit aus den Daten geschätzten Parametern.

- ❖ Exponentialverteilung: $$f(x)=\begin{cases}\lambda e^{-\lambda x} & x\geq 0\\ 0 & x<0\end{cases}$$
- ❖ Gammaverteilung: $$f(x)=\begin{cases}\dfrac{b^{p}}{\Gamma(p)}x^{p-1}e^{-bx} & x\geq 0\\ 0 & x<0\end{cases}$$

$$\text{wobei}\quad \Gamma(x)=\int_{0}^{\infty}t^{x-1}e^{-t}\,dt$$

An den beiden Dichtefunktionen ist zu erkennen, dass die Exponentialverteilung ein Spezialfall der Gammaverteilung ist, da Gamma(p=1,b=λ)=Exp(λ). Die Exponentialverteilung scheint keine für die Daten geeignete Verteilung zu sein, die Verallgemeinerung, also die Gammaverteilung scheint die Kerndichteschätzungen bzw. die Histogramme jeweils ganz gut zu treffen.
Um dies zu verifizieren wurden Laufzeit-Stichproben der Größe N=100 aus TSP(n=50,60,80,80) mit jeweiligen r=5,10,20,40. Jede dieser Stichproben ist nach einem unbekannten $F_{n,r}$ verteilt. Für jede dieser Stichproben wurden Mittelwert $\mu_{n,r}$ und Varianzen $\sigma_{n,r}^2$ geschätzt und ein Kolmogorov-Smirnoff-Test (K-S-Test) auf Gamma($b=\mu_{n,r}^2/\sigma_{n,r}^2$, $p=\mu_{n,r}/\sigma_{n,r}^2$)-Verteilungen durchgeführt. Die Folgende Tabelle zeigt die Ergebnisse für die Hypothesen

$$H_0: F_{n,r} = \text{Gamma}(b=\mu_{n,r}^2/\sigma_{n,r}^2,\ p=\mu_{n,r}/\sigma_{n,r}^2)$$
$$H_1: F_{n,r} \neq \text{Gamma}(b=\mu_{n,r}^2/\sigma_{n,r}^2,\ p=\mu_{n,r}/\sigma_{n,r}^2)$$

n=50	r=5	r=10	r=20	r=40
ks-Statistik	0,095	0,097	0,089	0,103
kritischer Wert	0,136	0,136	0,136	0,136
H_0 Verwerfen?	Nein	Nein	Nein	Nein

n=60	r=5	r=10	r=20	r=40
ks-Statistik	0,121	0,096	0,065	0,101
kritischer Wert	0,136	0,136	0,136	0,136
H_0 Verwerfen?	Nein	Nein	Nein	Nein

n=70	r=5	r=10	r=20	r=40
ks-Statistik	0,098	0,086	0,092	0,095
kritischer Wert	0,136	0,136	0,136	0,136
H_0 Verwerfen?	Nein	Nein	Nein	Nein

n=80	r=5	r=10	r=20	r=40
ks-Statistik	0,084	0,095	0,084	0,132
kritischer Wert	0,136	0,136	0,136	0,136
H_0 Verwerfen?	Nein	Nein	Nein	Nein

Tabelle 12: Ergebnisse der Anpassungstests

3.1.2. Abhängigkeit der mittleren Laufzeit von n und r

Dieses Unterkapitel befasst sich mit der Modellierung der mittleren Laufzeit l durch die Dimension n des TSP und die Suchtiefe r der Heuristik. Dazu werden zunächst die univariaten Abhängigkeiten zwischen l und n bzw. zwischen l und r betrachtet. Wie aus der TSP-Theorie bekannt (siehe Kapitel 1) gehört das TSP zu den NP-vollständigen Problemen [Garey & Johnson, 1979], das heißt also, dass kein Algorithmus bekannt ist, der das TSP *exakt* in polynomialer Zeit löst.

3.1.2.1. Abhängigkeit der mittleren Laufzeit von n bei festem r

Betrachten wir nun das Laufzeitverhalten der ILS_r Heuristik für die fixe Suchtiefe r=5:

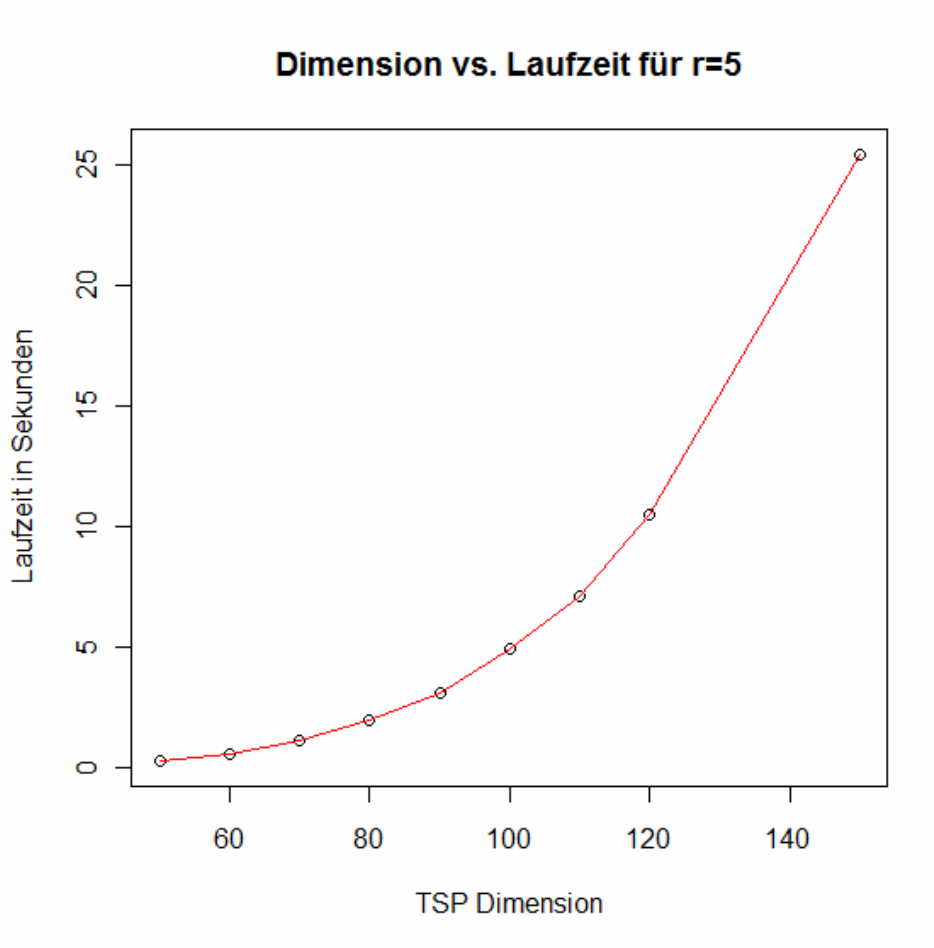

Abbildung 10: Abhängigkeit der Laufzeit von n

n	50	60	70	80	90	100	110	120	150
l	0,263	0,570	1,102	1,974	3,137	4,914	7,146	10,490	25,480
ln(l)	-1,334	-0,561	0,097	0,680	1,143	1,592	1,967	2,350	3,238

Tabelle 13: Laufzeiten in Abhängigkeit von n

Die hier dargestellten Laufzeiten sind arithmetische Mittelwerte aus metrischen Stichproben der Größe N=100 von TSP-Instanzen der Dimensionen 50,60,70,80,90,100,110,120 und 150. Das Resultat zeigt einen ziemlich „glatten“ beinahe perfekten, möglicherweise exponentiellen oder polynomialen Zusammenhang zwischen der TSP-Dimension und der mittleren Laufzeit der Heuristik mit Suchtiefe r=5.
Dies lässt auf folgenden 1. Modellierungsansatz (exponentiell) schließen:

$$l = a * e^{n*b}$$

(wobei l=Laufzeit, n=TSP-Dimension und a,b ϵ R unbekannte Konstanten)

Logarithmiert ergibt dies:

$$ln(l) = ln(a) + n*b$$

also ein lineares Modell, bei dem die abhängige Größe ln(l) linear in n von ln(a) und b abhängt.

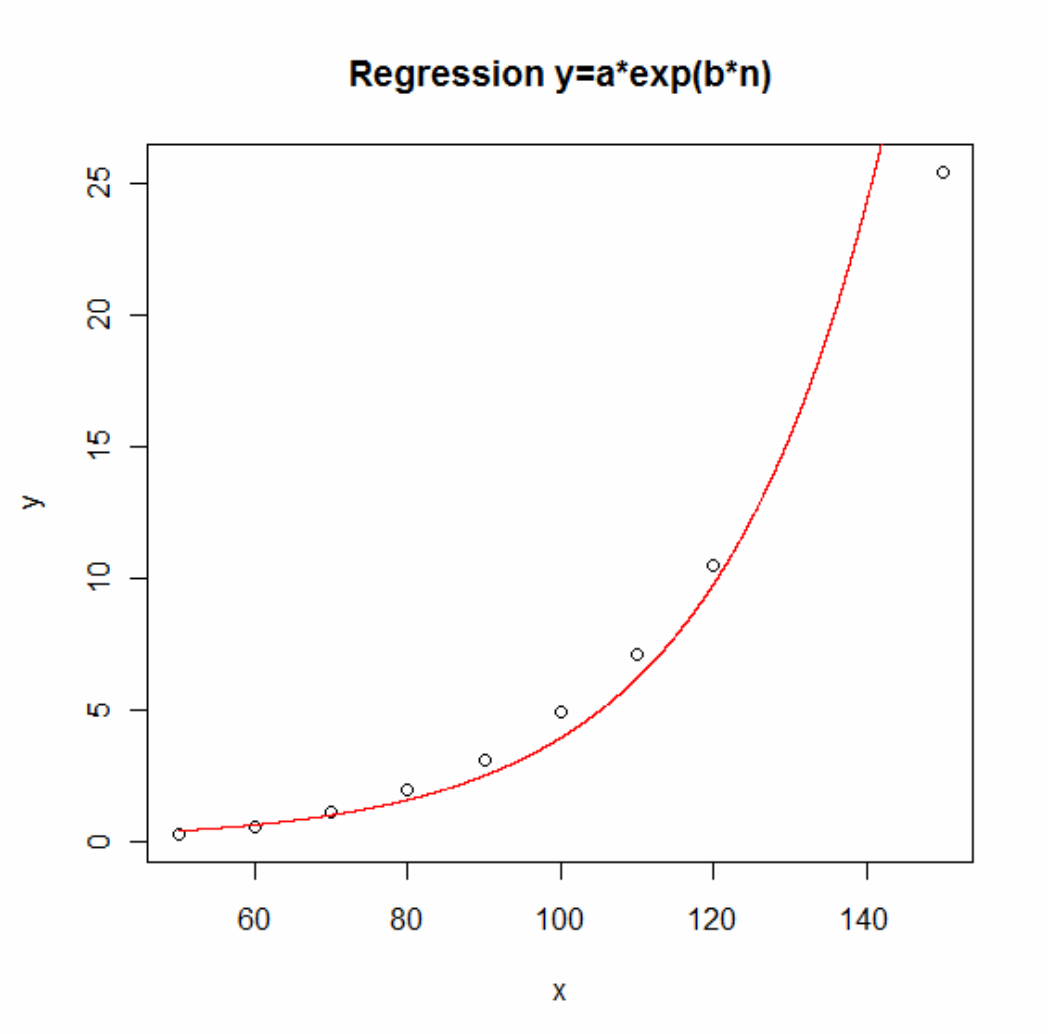

Abbildung 11: Fit des Exponentialmodells

Zusammenfassung der Regression:

Modell1	Wert	Std.Fehler
ln($\hat{a}$)=	-3,1667	
$\hat{a}$ =	0,0421	0,3032
$\hat{b}$ =	0,0454	0,0031
R^2=	0,9678	

Tabelle 14: Modellparameter des Exponentialmodells

Bis n=120 scheint der Fit noch durch die exponentielle Regression ganz passabel zu funktionieren. Die mittlere Laufzeit bei n=150 wirkt allerdings ein wenig außenstehend. Die Annahme eines exponentiellen Anstiegs der Laufzeit mit der Instanzgröße n scheint zwar ganz gut zu passen, allerdings gibt sie offenbar einen zu steilen Trend vor. Ein Blick auf die Residuen gibt möglichen Aufschluss auf mehr Systematik in den Fehlern:

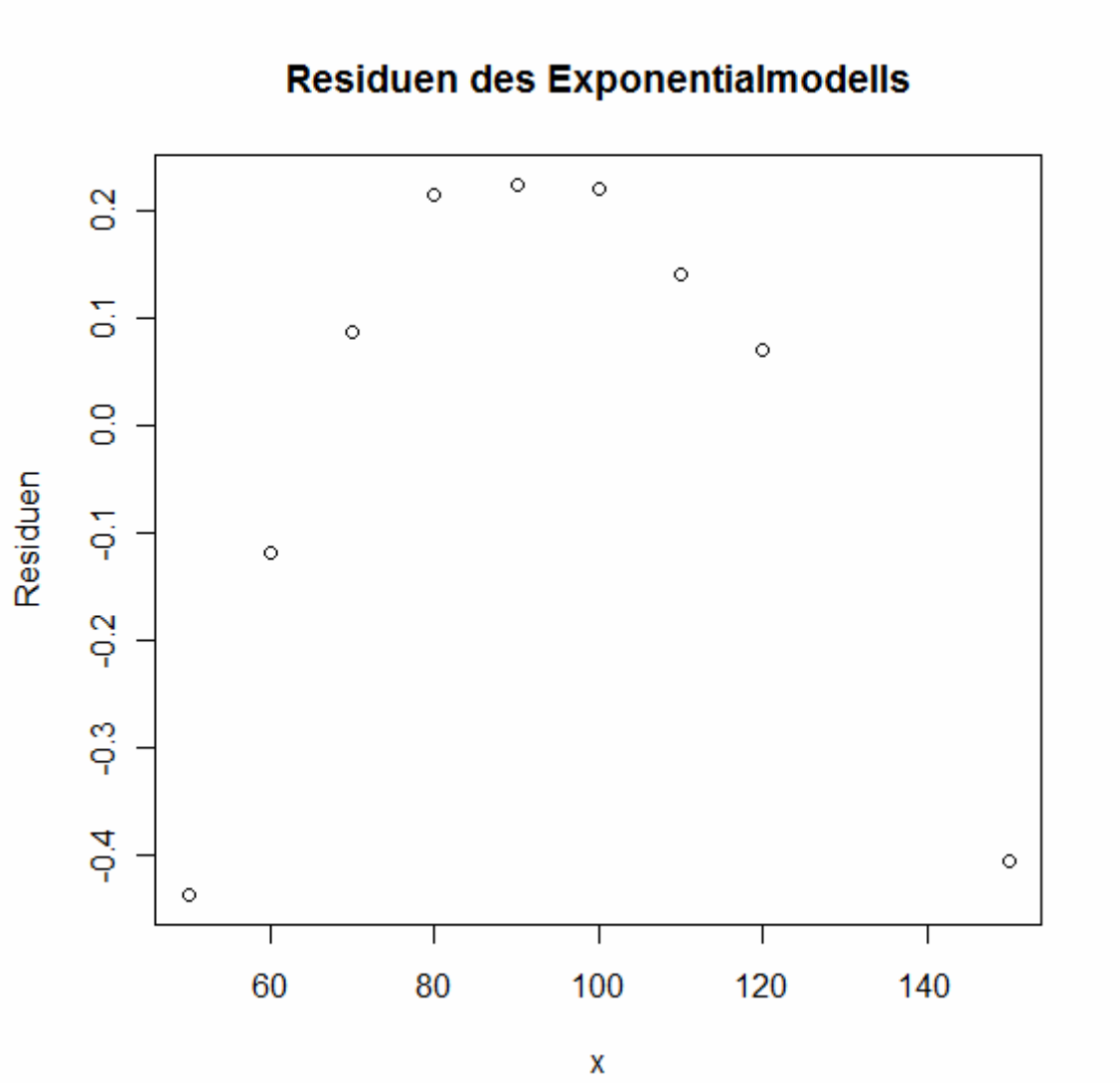

Abbildung 12: Residuen des Exponentialmodells

Der Residuenplot bietet einen für das statistisch geschulte Auge fast schauderhaften Anblick. Eine fast perfekte Parabel lässt vermuten, dass die Annahme des exponentiellen Wachstums die Daten systematisch „verschätzt“. Für die Schätzung der Laufzeit der Heuristik für TSP Instanzen bis n=150 scheint das Modell brauchbar, darüber hinaus ist allerdings eher Vorsicht geboten und anzunehmen, dass der geschätzte Trend wesentlich schneller ansteigt, als der den Daten zugrunde Liegende.

Auch das augenscheinlich gute Bestimmtheitsmaß R^2=0,9678 trügt, da es natürlich nur den vorhandenen Datenbereich abdeckt, nicht aber Information über den Fit über die n=150 Grenze hinaus liefert.

Die Tatsache, dass exponentielles Wachstum das Tatsächliche überschätzt ist aus sicht des 1. Modellierungsansatzes natürlich schlecht, aus rein praktischer Sicht

jedoch zufrieden stellend (exponentielle Algorithmen die das TSP exakt lösen sind bekannt, polynomiale jedoch nicht!) da eine exponentielle Heuristik, die ohnehin keine exakten, sondern eben nur „gute“ Lösungen liefert, uninteressant wäre. Dieser Gedanke führt zu einem 2. Modellierungsansatz (quadratisch):

$$l = (a + b*n)^2$$

$$\leftrightarrow$$

$$\sqrt{y} = a + b*n$$

Ähnliche Arithmetik wie bei Modell 1 führt das quadratische Modell auf ein Lineares zurück, in dem nun die Quadratwurzeln von l linear mit den TSP-Dimensionen n in Zusammenhang gebracht werden (Tabelle 15 und Abbildung 13):

n	50	60	70	80	90	100	110	120	150
l	0,263	0,570	1,102	1,974	3,137	4,914	7,146	10,490	25,480
sqrt(l)	0,513	0,755	1,050	1,405	1,771	2,217	2,673	3,239	5,048

Tabelle 15: Laufzeiten und deren Quadratwurzeln in Abhängigkeit von n

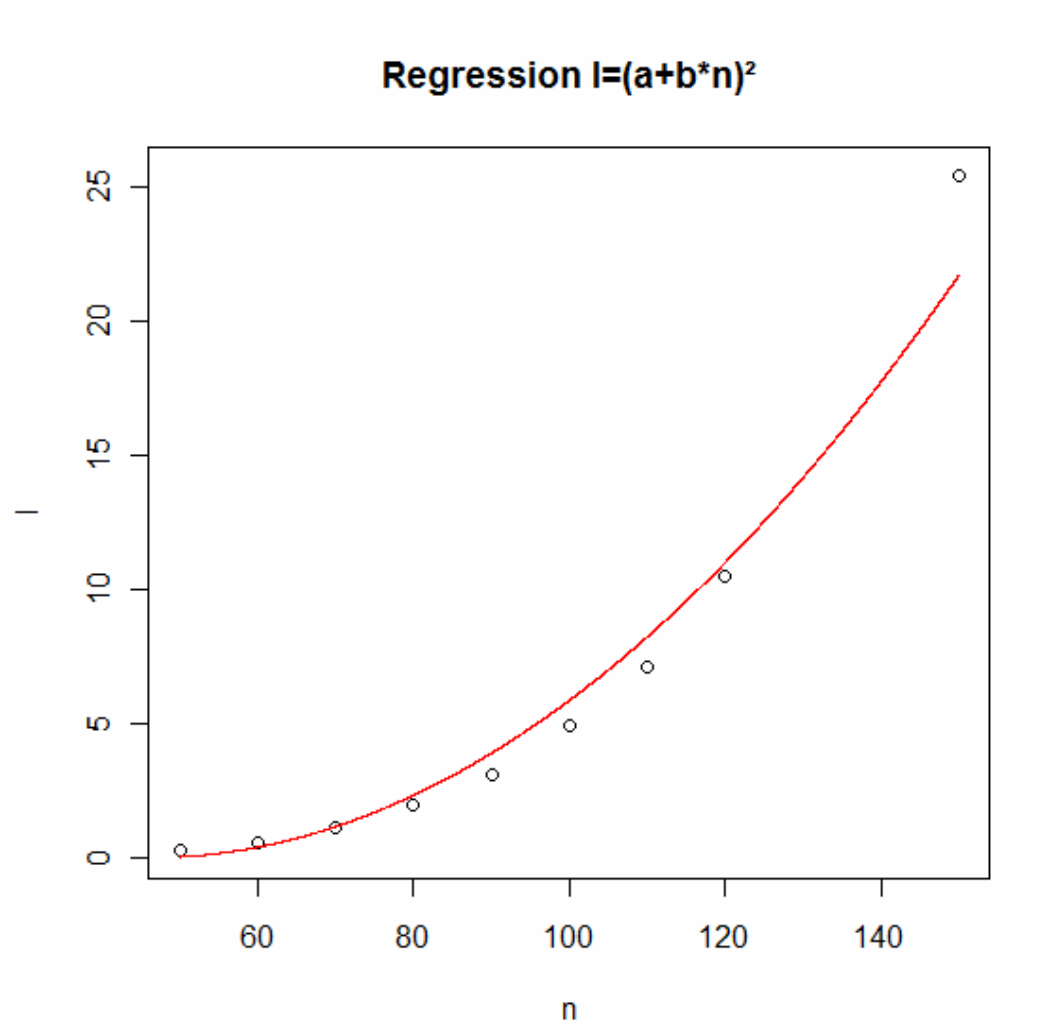

Abbildung 13: Fit des quadratischen Modells

Zusammenfassung der Regression:

Modell 2	Wert	Std.Fehler
$\hat{a}$ =	-2,0514	0,2654
$\hat{b}$ =	0,0447	0,0027
R^2=	0,9744	

Tabelle 16: Modellparameter des quadratischen Modells

Hier scheint der Fit auf den ersten Blick noch besser zu passen, auch das R^2=0,9744 spricht eine deutliche Sprache. Allerdings wirkt auch der quadratische Trend noch nicht ganz optimal: Angefangen von einer Unterschätzung im Bereich n<60, folgt eine Überschätzung bis n=120 und daraufhin wieder (und dies vermutlich für n>150) eine Unterschätzung. Der quadratische Trend dürfte also für den tatsächlich

zugrunde Liegenden zu schwach ansteigen. Ein Blick auf den Residuenplot zeigt ähnliches wie in Modell 1:

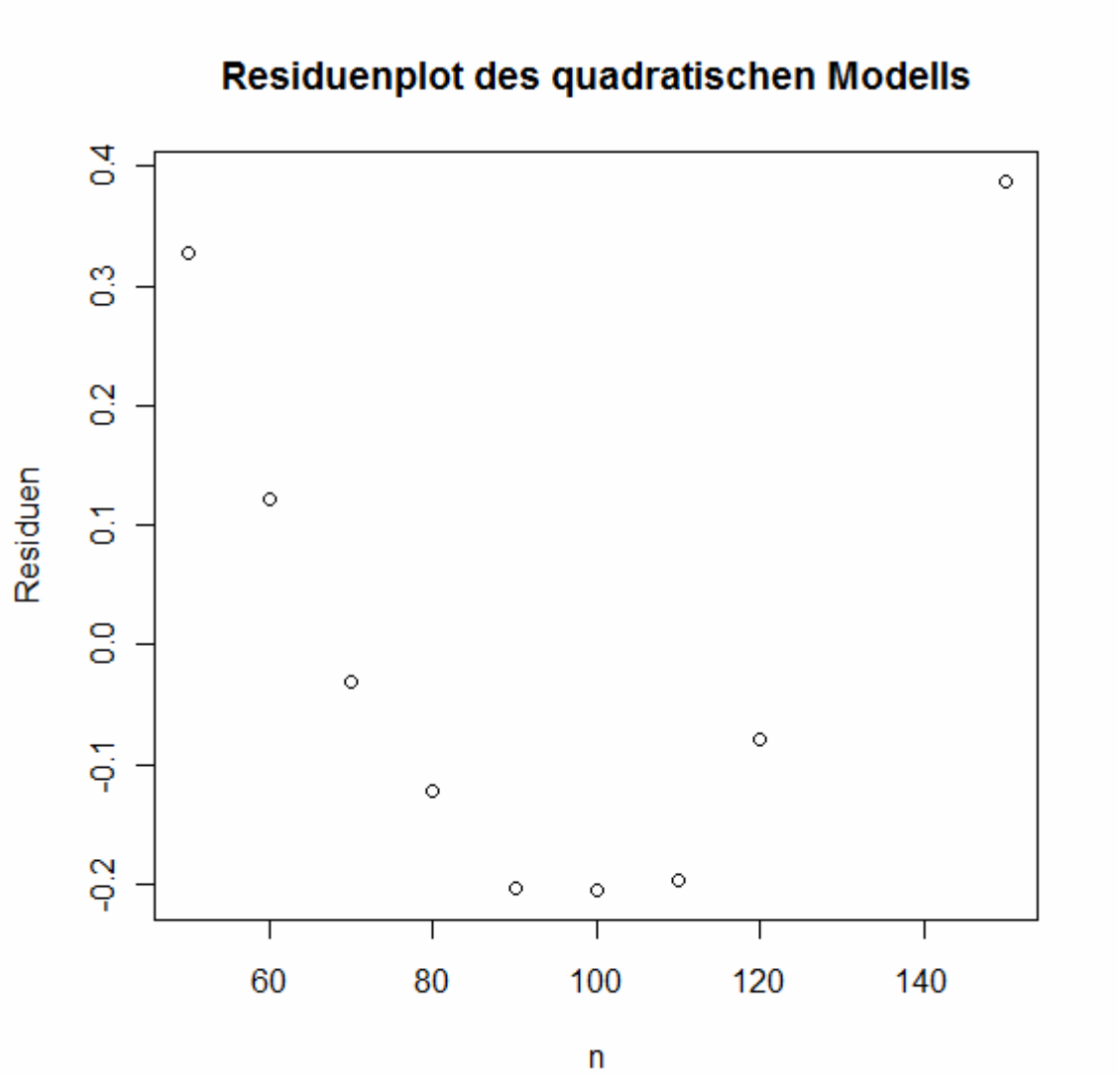

Abbildung 14: Residuen des quadratischen Modells

Der Unterschied der zweiten Modellierungsvariante zur Ersten besteht also darin, dass kein exponentieller, sondern ein polynomialer, im Speziellen quadratischer Zusammenhang unterstellt wird. Auch der quadratische Zusammenhang scheint noch nicht perfekt zu passen, er unterschätzt die Steile des Zusammenhangs im Gegensatz zum exponentiellen Modell. Nun lässt sich das quadratische Modell noch weiter auf ein polynomiales Modell verallgemeinern, das nicht nur quadratische, kubische oder höhere ganzzahlige Potenzen zulässt, sondern beliebig reelwertige:

$$I = (a+b*n)^k \quad k \epsilon R^+$$

Abbildung 15 veranschaulicht den Fit der unterschiedlichen Modelle:

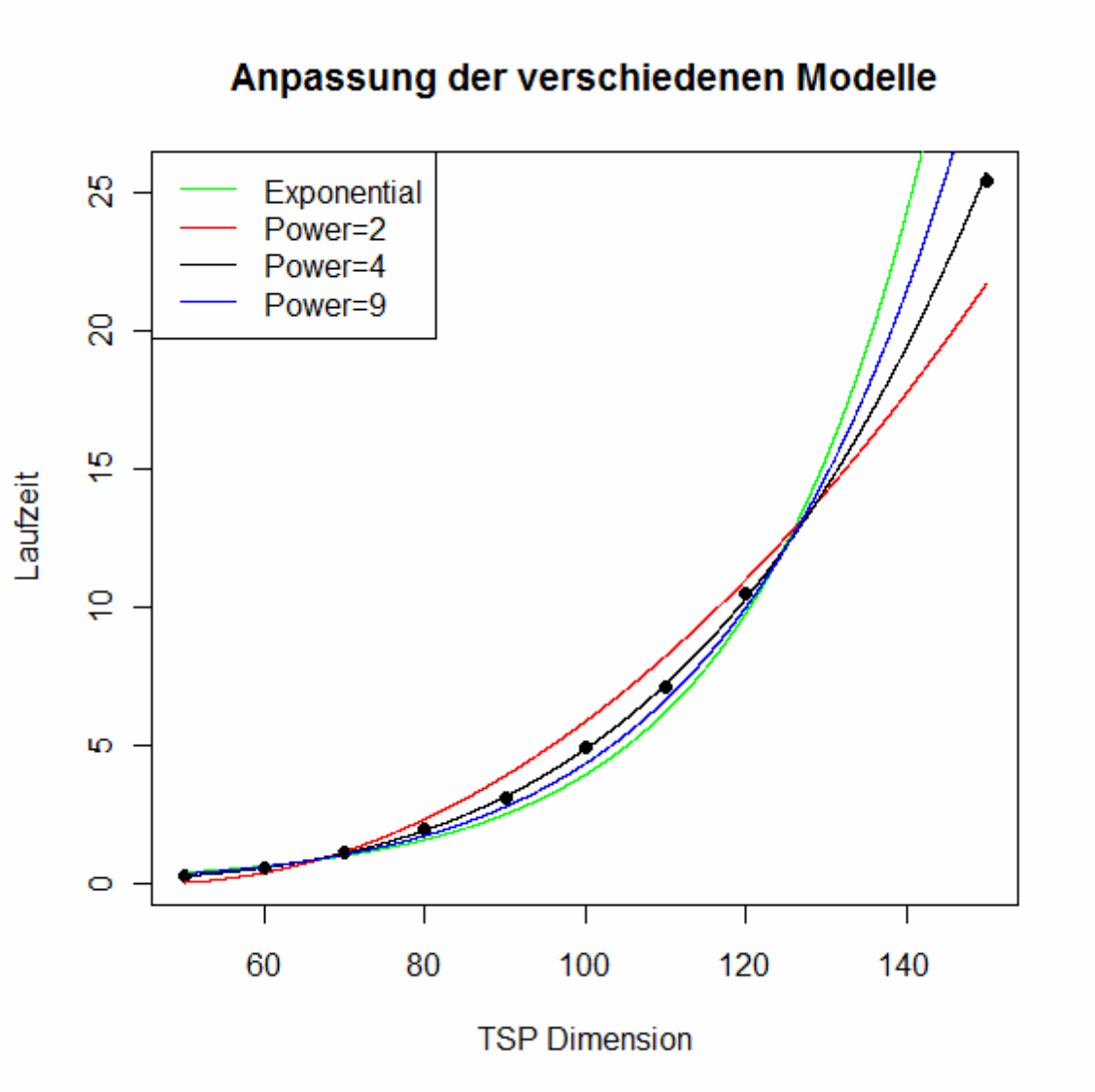

Abbildung 15: Vergleich der postulierten Modelle

Je nach Wahl von k, liefert die Verallgemeinerung von Modell 2 unterschiedlich gute Anpassung:

Modell	R^2
exp	0,9678038
k=1	0,833296
k=2	0,9744285
k=3	0,9972467
k=4	0,9999078
k=5	0,9983416
k=6	0,9959382
k=7	0,993552
k=8	0,991397
k=9	0,9895058

Tabelle 17: Bestimmtheitsmaße der Modelle mit unterschiedlichen Potenzen

Fasst man R^2: $R^+ \rightarrow [0;1]$ als Funktion auf, die k nach [0;1] abbildet, besitzt R^2 einen lokalen (=globalen) Maximierer $k_{opt} \in [3;5]$.
Die Funktion R^2 ist allerdings nicht explizit darstellbar, somit kann das exakte Optimum nicht im herkömmlich, analytischen Sinne berechnet werden. Abbildung 16 zeigt den Funktionsverlauf für k=1 bis k=50 in 0,01er Schritten.

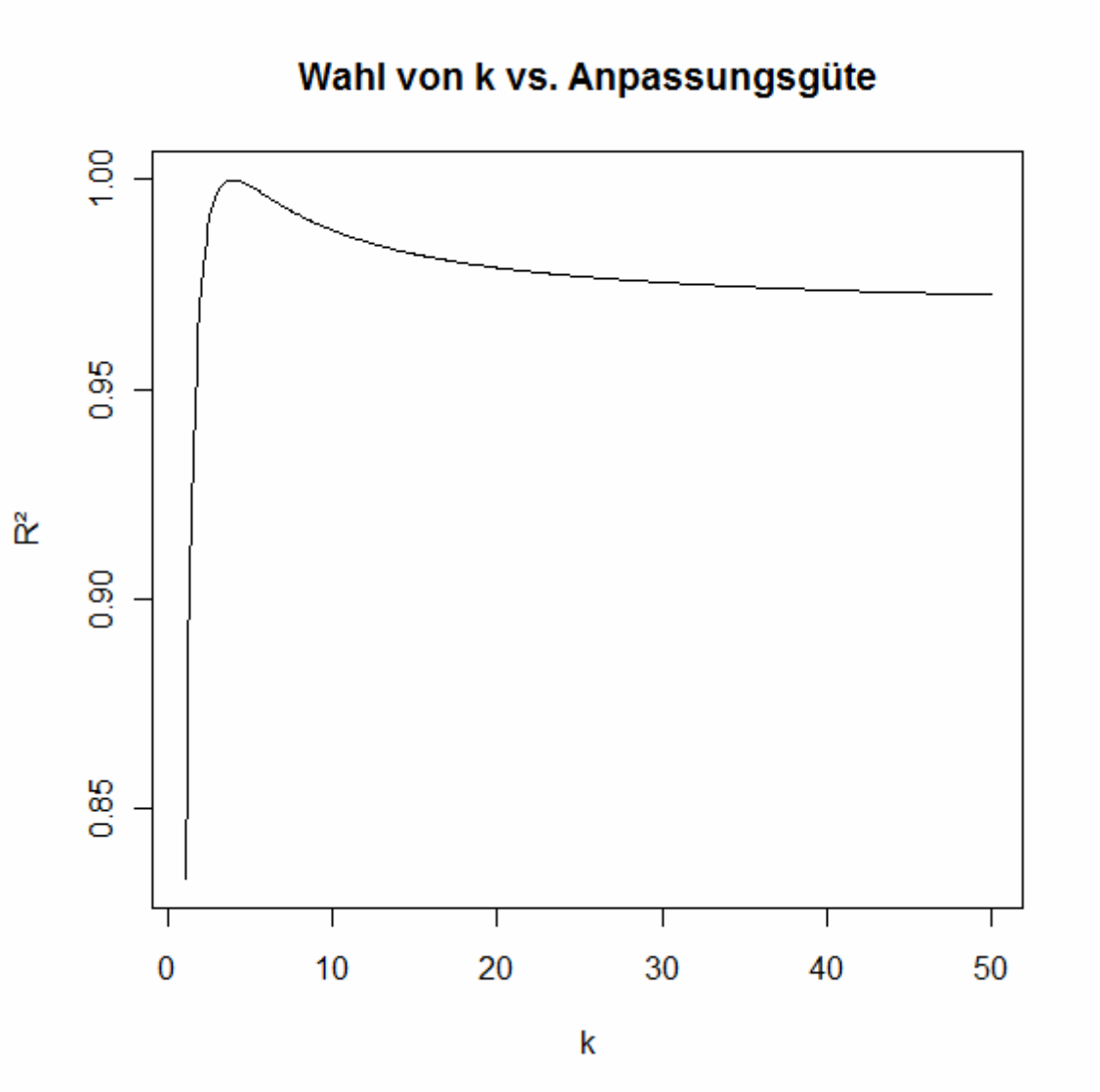

Abbildung 16: Anpassungsgüte in Abhängigkeit der Potenz

Wie bereits in der Funktionswertetabelle ersichtlich, liegt der gesuchte Maximierer k_{opt}, der die beste Anpassung liefert, offensichtlich im Intervall [3;5].
Die Ermittlung von k_{opt} erfolgt durch ein grid-search[1]-Verfahren:

[1] Programmcode siehe Kapitel 6.3.1

1. Sei F die Zielfunktion, D die Definitionsmenge, p die gewünschte Suchpräzision und kϵN beliebig
2. Finde x_u und x_o so, dass gilt $x_u<x_{opt}<x_o$ mit $F(x_{opt})>F(x)$ für alle xϵD
3. Teile $[x_u;x_o]$ in k gleich große Intervalle $[x_u=x_0;x_1]$, $]x_1;x_2]\ldots]x_{k-1};x_k=x_o]$
4. Werte F an den Intervalleckpunkten x_i (i=0,…,k) aus und finde x_b bϵ{0,1,…,k} sodass $F(x_b)>F(x_j)$ für alle jϵ{0,1,…,k}\b
5. Setze $x_u:=x_{b-1}$ und $x_o:=x_{b+1}$. Falls $x_o-x_u>p$ gehe zu 3.
6. Der Minimierer ist $(x_u/2-x_o/2)$

Für das Laufzeitenmodell ergibt sich ein optimales k von 3,884371≈4, welches ein Bestimmtheitsmaß von 0,9999343 ergibt. In Worten bedeutet das, dass 99,99343% der Gesamtvarianz der Laufzeiten durch das Modell erklärt werden. Optisch ist zwischen dem optimalen k=3,884371 und dem gerundeten k=4 der Unterschied mit freiem Auge fast nicht unterscheidbar, wie Abbildung 17 zeigt.

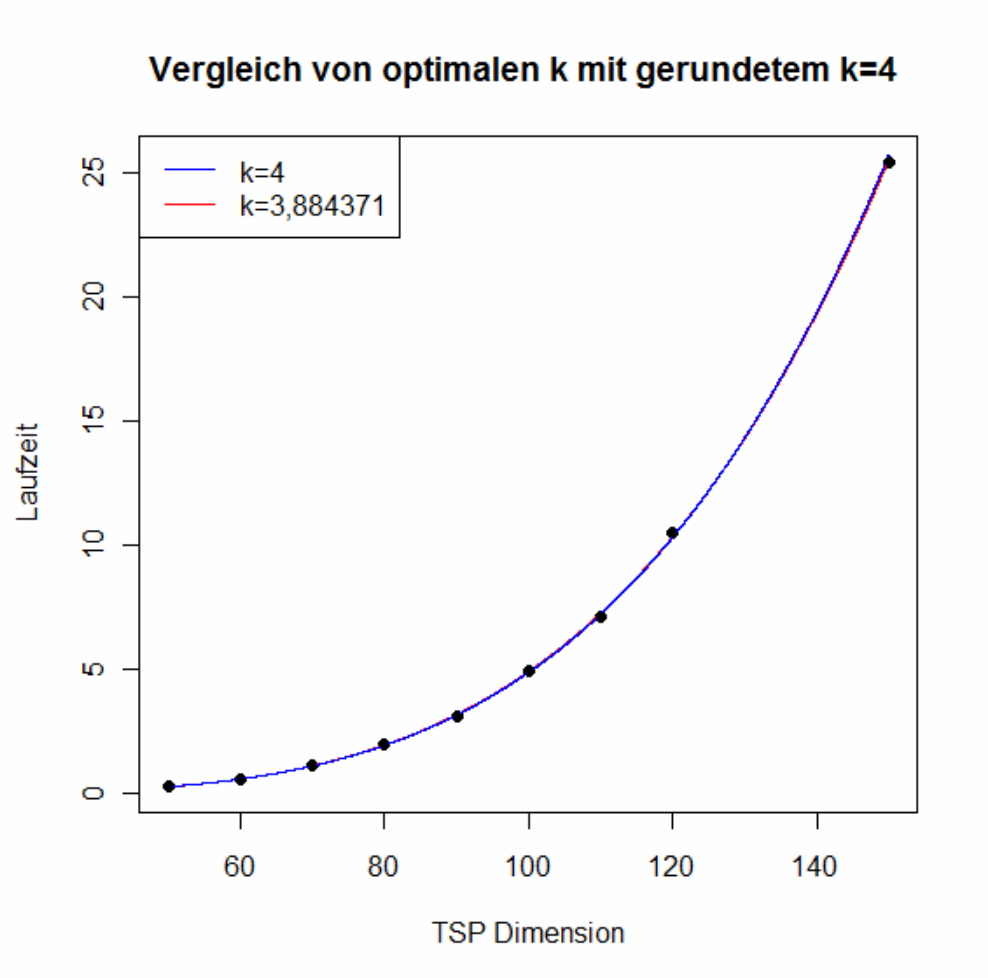

Abbildung 17: Fit des optimalen und des gerundeten Modells

Einen fast perfekten Fit liefert also das Modell:

$$l = (-0.08960 + 0.01595 * n)^{3,884371}$$

$$(\text{bzw.} \quad l = (-0.08960 + 0.01595 * n)^{4})$$

für die Laufzeiten l der Heuristik in Abhängigkeit von der TSP-Dimension n bei einer fixen Suchtiefe von r=5.

Im Folgenden beobachten wir nun, wie sich die Laufzeit in Abhängigkeit von r bei festem n verhält.

3.1.2.2. Abhängigkeit der mittleren Laufzeit von r bei festem n

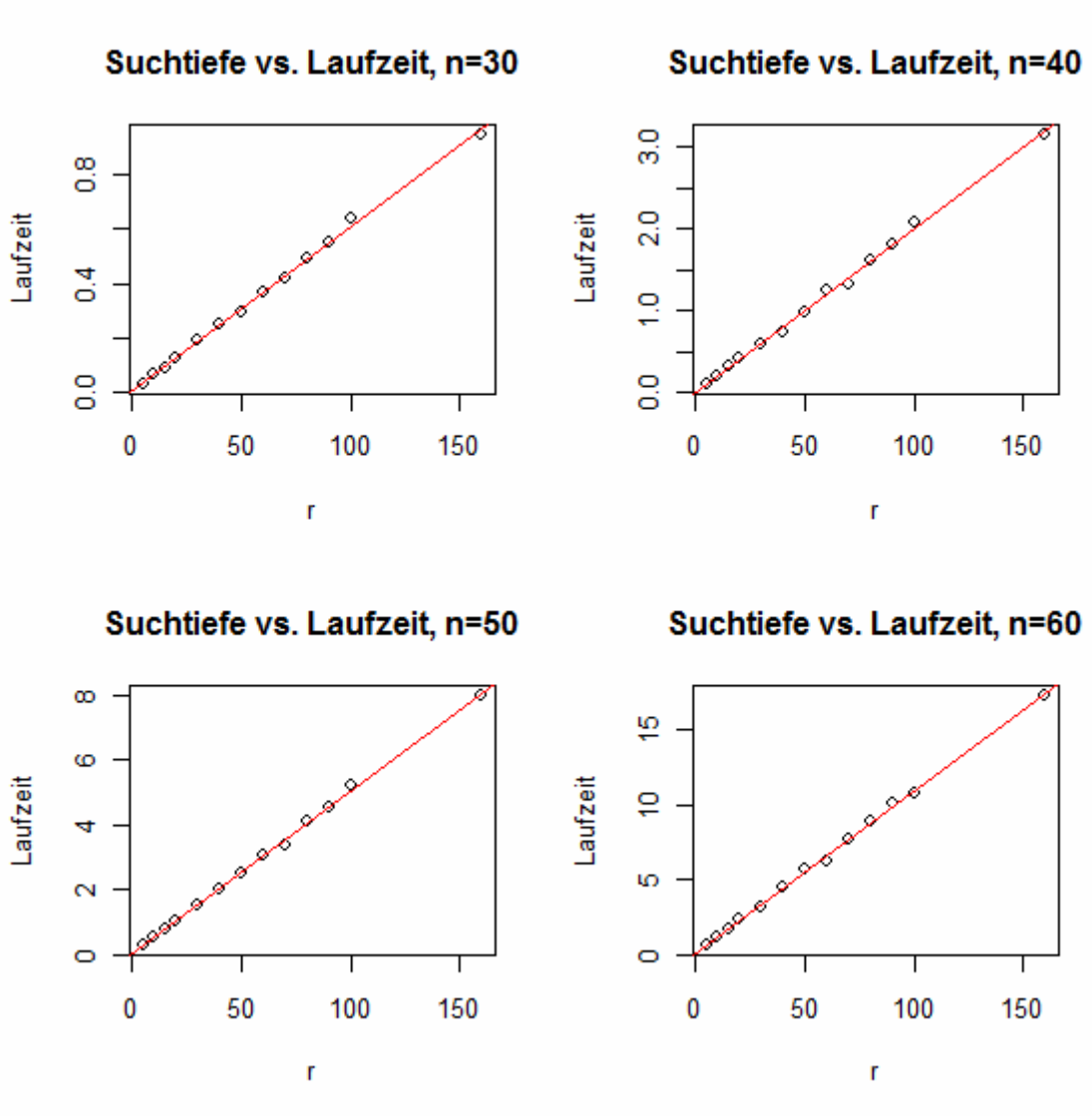

Abbildung 18: Abhängigkeit der Laufzeit von r bei festem n

Die Laufzeit scheint, wie Abbildung 18 zeigt, also linear von der Suchtiefe der Heuristik abhängen. Die Bestimmtheitsmaße für die linearen Modelle, die Aufschluss darüber geben, wie streng dieser lineare Zusammenhang ist, sind in Tabelle 18 zusammengefasst:

n	R^2
30	0.9978698
40	0.997955
50	0.9985712
60	0.998689
70	0.9972286
80	0.9985321

Tabelle 18: Bestimmtheitsmaße für linearen Zusammenhang

Unabhängig von n besteht also offenbar ein linearer Zusammenhang zwischen der Suchtiefe und der Laufzeit. Die nächste Frage von Interesse ist, ob dieser Zusammenhang auch unabhängig von n immer derselbe ist, oder ob er möglicherweise von n abhängt, also für größere n anders ist, als für kleinere. Sollte dies der Fall sein, ist die Form dieses Zusammenhangs (z.B. linear, polynomial, exponentiell…) wesentlich für die Erstellung eines Prognosemodells.
Um diese Abhängigkeit zu prüfen, betrachten wir in den Gruppen n=30,40,50,60 die Faktoren f zwischen den Laufzeiten für die einzelnen Suchtiefestufen:

	n	r=5	r=10	r=15	r=20	r=30	r=40	r=50	r=60	r=70	r=80	r=90	r=100	r=160
Werte:	30	0,03	0,07	0,09	0,13	0,19	0,25	0,29	0,37	0,42	0,50	0,55	0,64	0,95
	40	0,11	0,21	0,32	0,42	0,60	0,75	0,99	1,24	1,34	1,63	1,82	2,08	3,16
	50	0,26	0,53	0,75	1,02	1,51	2,02	2,48	3,08	3,36	4,11	4,52	5,22	7,95
	60	0,57	1,09	1,65	2,28	3,16	4,43	5,64	6,19	7,62	8,94	10,12	10,78	17,26
Faktoren:	30		2,059	1,334	1,359	1,519	1,308	1,161	1,263	1,143	1,171	1,111	1,162	1,482
	40		1,885	1,497	1,341	1,425	1,241	1,325	1,256	1,075	1,218	1,119	1,142	1,517
	50		1,996	1,426	1,361	1,482	1,334	1,230	1,239	1,091	1,224	1,099	1,156	1,523
	60		1,916	1,508	1,381	1,390	1,400	1,274	1,098	1,231	1,172	1,132	1,066	1,600

Tabelle 19: Laufzeiten und Faktoren für Suchtiefestufen r

6 paarweise, gepaarte zweistichproben t-Tests geben nun Aufschluss, ob sich die Differenzen der Faktoren $d_{x,y}$ für beide Gruppen signifikant von 0 unterscheiden. Tun sie dies nicht, kann davon ausgegangen werden, dass die Faktoren in beiden Gruppen die selben, also unabhängig von der TSP-Dimension n sind! Dies wäre wiederum ein Hinweis darauf, dass der Zusammenhang zwischen r und l in den beiden Gruppen derselbe ist, also nicht von n abhängt. Ein und dasselbe Wachstumsgesetz für l in Abhängigkeit von r könnte also für verschiedene n verwendet werden! Die Hypothesen des Tests lauten:

$$H_0: d_{x,y} = 0$$
$$H_1: d_{x,y} \neq 0$$

Der Test fällt bei einem p-Wert von 0,9865 und einem Signifikanzniveau von 0,05 (Bonferroni-adapatiert auf 0,0083 bei 6 paarweisen Tests) nicht signifikant zu Gunsten von H_0 aus.

Im nächsten Schritt gilt es, das vermutlich lineare Wachstumsgesetz heraus zu finden. Im Folgenden sind die mittleren Laufzeiten für unterschiedliche

Suchtiefestufen bei n=30,40,50,60 tabelliert. Diese lassen darauf schließen, dass eine Verdoppelung der Suchtiefe eine Verdoppelung der Laufzeit zur Folge hat:

	n	r=5	r=10	r=20	r=40	r=80	r=160
Werte:	30	0,034	0,070	0,127	0,252	0,495	0,948
	40	0,112	0,211	0,423	0,748	1,629	3,159
	50	0,263	0,526	1,021	2,018	4,111	7,954
	60	0,570	1,093	2,276	4,429	8,938	17,257
Faktoren:	30		2,0588	1,813	1,9865	1,9631	1,9151
	40		1,8847	2,0084	1,768	2,1781	1,9395
	50		1,996	1,9419	1,976	2,037	1,935
	60		1,9155	2,083	1,946	2,0178	1,9308

Tabelle 20: Laufzeiten und Faktoren für Verdoppelung der Suchtiefestufen

Und tatsächlich sind die Faktoren, mit denen man die Laufzeiten multiplizieren muss, um auf die Laufzeit bei doppelter Suchtiefe zu kommen unabhängig von der TSP-Dimension ungefähr 2. Dieser Verdoppelungsfaktor wird in Folge als q bezeichnet.

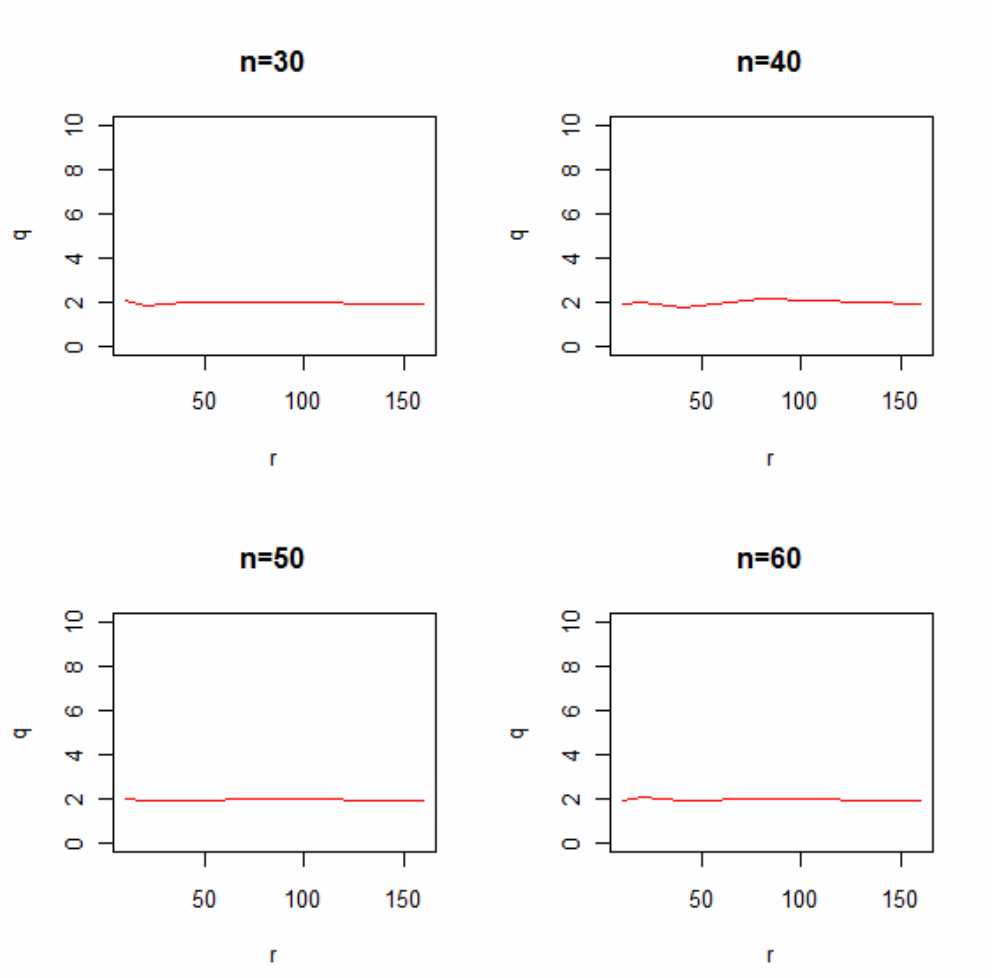

Abbildung 19: Verlauf des Verdoppelungsfaktors in Abhängigkeit von r

Die Schätzung von q durch das arithmetische Mittel der beobachteten Faktoren liegt bei 1,96.
Der t-Test der Hypothese (H_0: q=2, H_1: q≠2) fällt mit einem p-Wert von 0,09818 bei einem Signifikanzniveau von 0,05 nicht signifikant aus. Das 95%-Konfidenzintervall für q ist:

$$P(q\epsilon[1{,}922;\ 2{,}008]) = 0.95$$

Sei n fix. Dann ist $l : N \rightarrow R^+$ die Laufzeitfunktion abhängig von r.
Die inhaltlich durchaus sinnvolle Hypothese q=2 kann von den beobachteten Daten nicht widerlegt werden. Es macht also Sinn anzunehmen:

$$\mathbf{k*l(r=a) = l(r=a*k)}$$

3.1.3. Abhängigkeiten der Varianz der Laufzeit von n und r

3.1.3.1. Abhängigkeit der Laufzeitvarianz von n bei festem r

Die unterstellte Gammaverteilung ist durch ihren Mittelwert und ihre Varianz (die üblichen Darstellung durch rate und shape sind lediglich alternative Parametrisierungen, stehen aber in einem 1:1 Verhältnis zu Mittelwert und Varianz) bereits gegeben. Es gilt:

$$X \sim \gamma(p;b)$$

$$\rightarrow E(X) = \frac{p}{b}$$

$$\rightarrow VAR(X) = \frac{p}{b^2}$$

Gelingt es also in ähnlicher Art und Weise ein Prognosemodell für die Varianzen der Laufzeiten, zunächst in Abhängigkeit von n bei festem r=5, zu erstellen, wären nicht nur die mittleren Laufzeiten in Abhängigkeit von n und r schätzbar, sondern die ganze Laufzeitverteilung und somit auch Wahrscheinlichkeiten, Konfidenzbereiche etc.
Im Folgenden sind die Laufzeitvarianzen aus Stichproben der Größe N=100 für n=50,60,70,80,90,100,110,120,150 bei festem r=5 tabelliert:

n	50	60	70	80	90	100	110	120	150
Varianz	0,006	0,039	0,107	0,443	1,253	2,290	6,408	14,652	73,477

Tabelle 21: Laufzeitvarianzen in Abhängigkeit von n

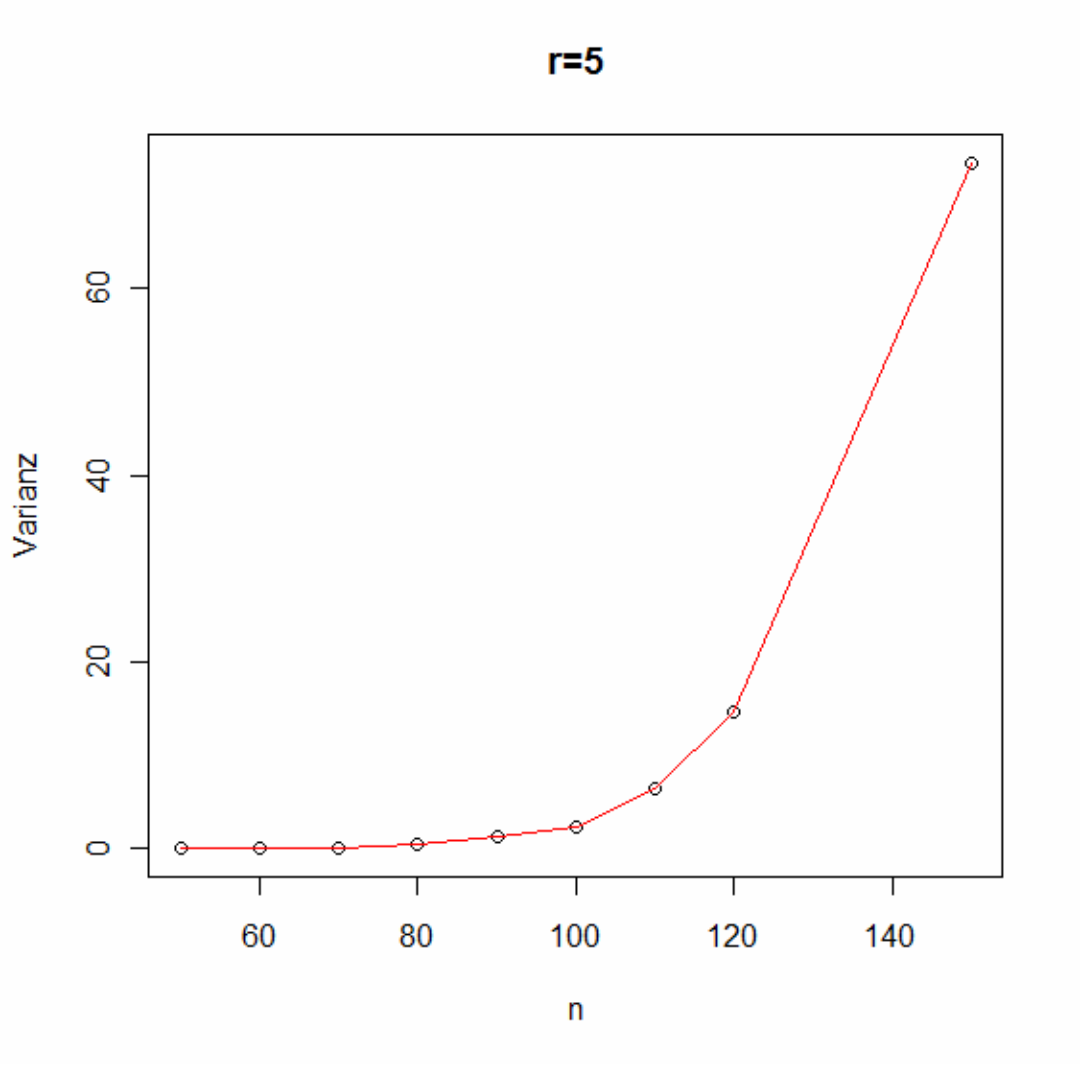

Abbildung 20:Laufzeitvarianz in Abhängigkeit von n

Wie im Fall der Mittelwerte scheint ein exponentieller bzw. polynomialer Zusammenhang plausibel.

Analog zur Vorgehensweise beim Modell der mittleren Laufzeiten fitten wir in Abbildung 21, um einen ersten optischen Eindruck zu vermitteln, exponentielle bzw. polynomial_k (k=2,4,9) Trends.

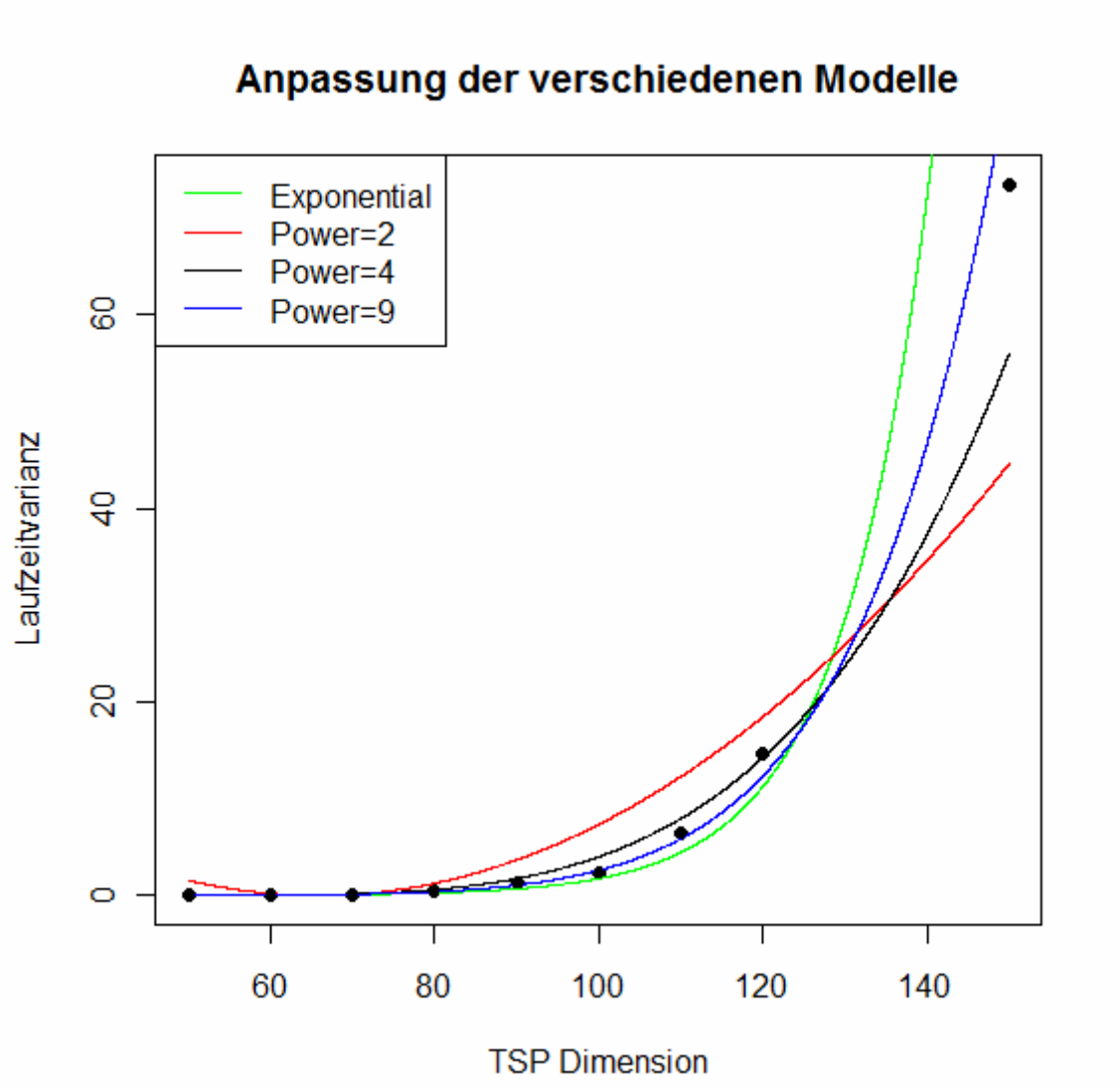

Abbildung 21: Vergleich der unterschiedlichen Varianzmodelle

Die bestmögliche Anpassung liegt offenbar zwischen Grad 4 und 9.

Die Bestimmtheitsmaße der Modelle ergaben sich zu:

Modell	exponentiell	poly(Grad=1)	poly(2)	poly(3)	poly(4)	poly(5)	poly(6)	poly(7)	poly(8)	poly(9)
R^2	0,963	0,625	0,844	0,939	0,976	0,990	0,996	0,998	0,998	0,997

Tabelle 22: Anpassungsgüten der Varianzmodelle

Das in Abschnitt 3.1.2.1. vorgestellte grid-search Verfahren liefert ein Optimum von R^2=0.9979521 für einen Grad k=7,446092.

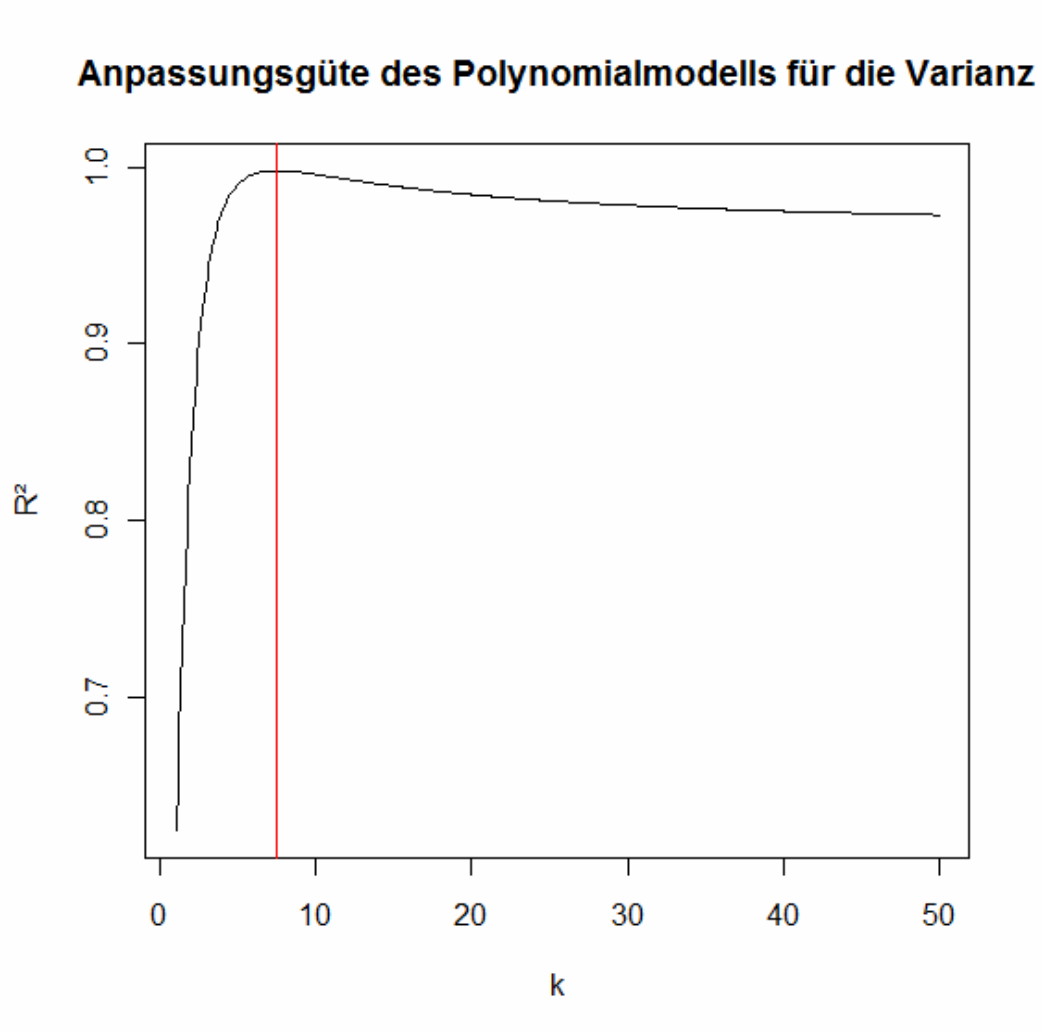

Abbildung 22: R^2 für die Varianzmodelle in Abhängigkeit der Potenz

Der Einfachheit wegen abgerundet auf k=7 liefert das Polynom immer noch eine Anpassung von R^2=0.9978173. Auffällig ist dabei, dass somit der optimale Grad die Standardabweichung, also die Quadratwurzel der Varianz zu prognostizieren 7,446092/2 ist, was in etwa dem optimalen Laufzeitmittelwert-Prognosegrad entspricht. Optisch ist, wie in Abbildung 23 zu sehen, zwischen dem optimalen und dem gerundeten Modell beinahe kein Unterschied auszunehmen:

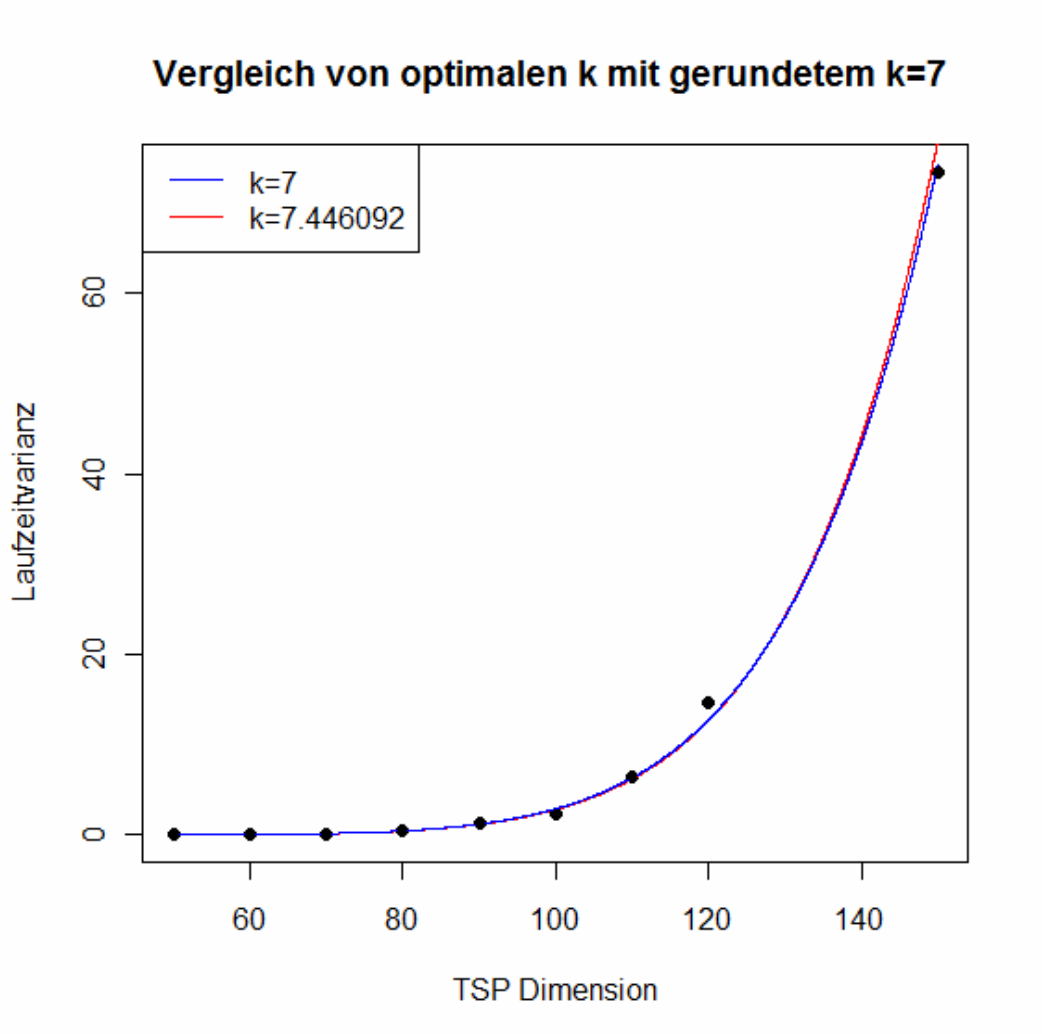

Abbildung 23: Vergleich des optimalen mit dem gerundeten Modell

Für festes r=5 erhalten wir also folgendes Laufzeitvarianzen-Prognosemodell:

$$\mathbf{Var_l = (-0.21003900 + 0.01373599*n)^7}$$

(bzw. $sd_l = (-0.21003900 + 0.01373599*n)^{7/2}$)

3.1.3.2. Abhängigkeit der Laufzeitvarianz von r bei festem n

Die Standardabweichungen, sd = $\sqrt{Var}$, der Laufzeiten sind für jeweils festes n=30,40,50,60 offensichtlich linear in Zusammenhang.

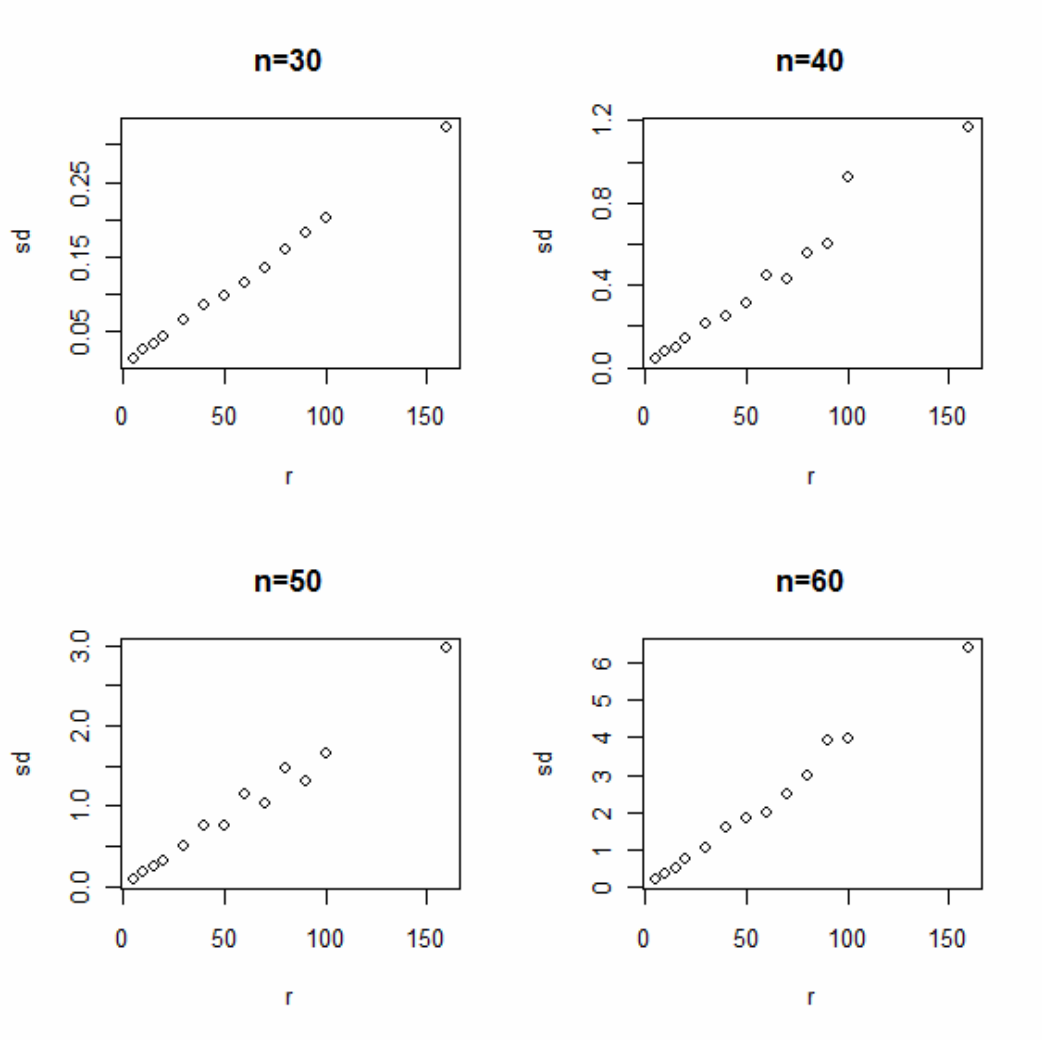

Abbildung 24: Laufzeitvarianz in Abhängigkeit von r

Der Fit scheint nicht für alle hier vorgestellten n perfekt zu funktionieren, betrachtet man jedoch die R^2 aus den jeweiligen linearen Modellen $sd = a_n + b_n * r$, kann man durchaus von passender Modellierung sprechen:

n=	30	40	50	60
R^2=	0,998	0,963	0,978	0,990

Tabelle 23: R^2 für lineare Modelle

Nun gilt es wieder zu testen, ob dieses (lineare) Fortpflanzungsgesetz über die TSP-Dimensionen n hinweg den gleichen Bedingungen folgt. Zuerst betrachten wir wieder die Stufenweisen sd sowie die Faktoren um von einer Stufe in die nächste zu gelangen (Tabelle 24).

	n	r=5	r=10	r=15	r=20	r=30	r=40	r=50	r=60	r=70	r=80	r=90	r=100	r=160
Werte:	30	0,014	0,026	0,033	0,044	0,066	0,086	0,099	0,114	0,135	0,161	0,183	0,202	0,324
	40	0,044	0,074	0,097	0,143	0,215	0,248	0,311	0,444	0,425	0,559	0,598	0,923	1,169
	50	0,081	0,174	0,239	0,324	0,500	0,756	0,748	1,137	1,023	1,467	1,317	1,654	2,958
	60	0,198	0,361	0,518	0,771	1,049	1,617	1,856	1,971	2,494	2,971	3,923	3,959	6,374
Faktoren:	30		1,808	1,283	1,339	1,488	1,299	1,15	1,156	1,184	1,187	1,14	1,102	1,606
	40		1,695	1,308	1,474	1,5	1,154	1,254	1,429	0,957	1,314	1,071	1,542	1,267
	50		2,155	1,377	1,355	1,543	1,513	0,989	1,521	0,899	1,434	0,898	1,256	1,789
	60		1,825	1,436	1,489	1,359	1,542	1,148	1,062	1,265	1,192	1,32	1,009	1,61

Tabelle 24: Sds der Laufzeiten und Faktoren für Suchtiefestufen

Die paarweisen, gepaarten t-Tests zu den Hypothesen:

$$H_0: d_{x,y} = 0 \qquad H_1: d_{x,y} \neq 0$$

p-Werte	n=30	n=40	n=50	n=60
n=30	X	0,771	0,214	0,237
n=40	X	X	0,429	0,775
n=50	X	X	X	0,634
n=60	X	X	X	X

Tabelle 25: Paarweise gepaarte-Tests auf Gleichheit der Faktoren

Keiner dieser paarweisen t-Tests fällt bei einem Bonferroni-adaptierten α=0,0083 signifikant aus. Die Hypothese, die Differenzen zwischen den Faktoren für die jeweiligen Stufen existiert nur Aufgrund von zufälligen Schwankungen, kann nicht verworfen werden. Inhaltlich bedeutet dies, dass sich diese Faktoren über n hinweg nicht verändern, was wiederum soviel heißt, dass das Zuwachsgesetz der Varianzen in Abhängigkeit von r nicht von n abhängt.
Um nun jedoch präzisere Aussagen über diese lineare Abhängigkeit machen zu können, betrachten wir wieder die Tabelle der Faktoren, diesmal jedoch nur für r=5,(2*5),2*(2*5),2*(2*2*5),…,2*(2*2*2*2*5):

	n	r=5	r=10	r=20	r=40	r=80	r=160
Werte:	30	0,014	0,026	0,044	0,086	0,161	0,324
	40	0,044	0,074	0,143	0,248	0,559	1,169
	50	0,081	0,174	0,324	0,756	1,467	2,958
	60	0,198	0,361	0,771	1,617	2,971	6,374
Faktoren:	30		1,8084	1,718	1,932	1,868	2,016
	40		1,6952	1,928	1,73	2,252	2,093
	50		2,155	1,866	2,334	1,939	2,017
	60		1,8254	2,138	2,096	1,838	2,145

Tabelle 26: Sds der Laufzeiten und Faktoren für Suchtiefestufen

Doppelte Suchtiefen haben offenbar wieder in etwa die doppelte Laufzeitstandardabweichung zur Folge. Auch diese Faktoren werden nun über die Gruppen n=30,40,50,60 hinweg, allerdings ungepaart, t-Tests unterzogen um die Hypothesen zu testen, ob sich die Laufzeitstandardabweichungsmittel zwischen den Gruppen unterscheiden:

p-Werte	n=30	n=40	n=50	n=60
n=30	X	0,569	0,09	0,159
n=40	X	X	0,391	0,61
n=50	X	X	X	0,639
n=60	X	X	X	X

Tabelle 27: Paarweise t-Tests auf Unterschiede der Faktoren

Auch hier fällt kein Test zum adaptierten α=0,0083 signifikant aus. Die Faktoren q für die Laufzeitstandardabweichungen bei jeweils verdoppeltem r ändern sich in ihrem Mittel also auch nicht über die Gruppen hinweg.
Wie auch bei den Laufzeitmitteln scheint hier der Faktor 2 die entscheidende Rolle zu spielen. Also testen wir folgende Hypothese für die gesamten Verdoppelungsfaktoren:

$$H_0: q=2$$
$$H_1: q \neq 2$$

Bei einem p-Wert von 0,466 kann H_0 (α=0,05) nicht verworfen werden. Weiters ist:

$$P(q \in [1.884774; 2.054796]) = 0,95$$

Sei n fix. Dann ist $sd : N \rightarrow R^+$ die Laufzeitstandardabweichungsfunktion abhängig von r.
Die inhaltlich durchaus sinnvolle Hypothese q=2 kann von den beobachteten Daten nicht widerlegt werden. Es macht also Sinn anzunehmen:

$$\mathbf{k*sd(r=a) = sd(r=a*k)}$$

3.1.4. Zusammenfassung

- Prognose der mittleren Laufzeit:

3.1.2.: → $l = (-0.08960 + 0.01595 * n)^{3,884371}$ bei r=5

3.1.2.: → unabhängiger Einfluss der Parameter n und r auf die Laufzeit

→ $l(r=a)*k = l(r=k*a)$

Insgesamt ergibt sich daher:

$$\mathbf{l(n,r) = r/5*(-0.08960 + 0.01595 * n)^{3,884371}}$$

(+regressionsbedingter Zufallsfehler)

- Prognose der Laufzeit Varianz:

3.1.3.: → $Var_l = (-0.21003900 + 0.01373599*n)^7$ bei r=5

→ $sd_l = (-0.21003900 + 0.01373599*n)^{7/2}$

3.1.3.: → unabhängiger Einfluss der Parameter n und r auf die Varianz und somit die Standardabweichung der Laufzeit.

→ $k*sd(r=a) = sd(r=a*k)$

Ingesamt ergibt sich daher:

$$\mathbf{sd_l(n,r) = r/5* (-0.21003900 + 0.01373599*n)^{7/2}}$$

- Prognose der Verteilung:

Die Gammaverteilung ist folgendermaßen parametrisiert:
Sei X~γ(b;p). Dann besitzt X die Dichte:

$$f(x) = \begin{cases} \frac{b^p}{\Gamma(p)} x^{p-1} e^{-bx} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

und es gilt:

$$\left.\begin{aligned} E(X) &= \frac{p}{b} \\ VAR(X) &= \frac{p}{b^2} \end{aligned}\right\} \Leftrightarrow \begin{cases} b = \dfrac{E(X)}{VAR(X)} \\ p = \dfrac{E(X)^2}{VAR(X)} \end{cases}$$

Die Zufallsvariable X bezeichnet nun die Laufzeit. Dann besagt das Modell:

$$E(X) \approx r/5*(-0.08960 + 0.01595 * n)^{3,884371} := \mu$$

$$sd(X) \approx r/5* (-0.21003900 + 0.01373599*n)^{7/2} := \sigma$$

dann können b und p folgendermaßen geschätzt werden:

$$b = \frac{\mu}{\sigma^2} \quad , \quad p = \frac{\mu^2}{\sigma^2}$$

Somit gilt weiters:

$$p(X \in [a,b]) = \int_a^b \frac{b^p}{\Gamma(p)} x^{p-1} e^{-bx} dx = \int_a^b \frac{\left(\frac{\mu}{\sigma^2}\right)^{\frac{\mu^2}{\sigma^2}}}{\Gamma\left(\frac{\mu^2}{\sigma^2}\right)} x^{\frac{\mu^2}{\sigma^2}} e^{-\frac{\mu}{\sigma^2}x} dx$$

3.1.5. Erfolgswahrscheinlichkeit der Heuristik

3.1.5.1. Empirischer Ansatz

Die ILS_r Heuristik liefert nicht zwangsläufig die optimale Lösung. Um die Wahrscheinlichkeit schätzen zu können, mit der das ILS_r Ergebnis tatsächlich das Optimale ist, bedienen wir uns erneut der Simulation. Für n<13 ist es mit alltäglicher Rechenleistung in akzeptabler Zeit möglich durch vollständige Enumeration relativ simpel TSP-Rundreisen zu optimieren. Die Laufzeiten verhalten sich wie folgt:

n	Laufzeit (sec)		
9	0,08		
10	0,73		
11	8,16		
12	103,58	1,726333333	Minuten
13	1430,79	23,8465	Minuten
14	20031,06	5,564183333	Stunden
15	300465,9	83,46275	Stunden
16	4807454,4	55,64183333	Tage
17	81726724,8	2,591537443	Jahre
18	1471081046	46,64767397	Jahre
19	27950539882	886,3058055	Jahre
20	5,59011E+11	17726,11611	Jahre
21	1,17392E+13	372248,4383	Jahre
22	2,58263E+14	8189465,643	Jahre
23	5,94005E+15	188357709,8	Jahre

Tabelle 28: Gemessene und berechnete Laufzeiten der Enumeration

Die Laufzeit für n≤12 wurde dabei gemessen, für n≥13 berechnet: l(n) = l(n-1)*n.
Ein handelsüblicher Rechner würde also 886 Jahre benötigen um ein TSP(19) per vollständiger Enumeration exakt zu lösen.
Der Grund, warum hiefür nach Alternativen gesucht wird liegt auf der Hand. Die ILS_r Heuristik liefert, im Gegensatz zur vollständigen Enumeration, zwar nicht unbedingt das tatsächliche Optimum, oftmals reicht allerdings bereits eine „gute“ Lösung in dafür vernünftiger Laufzeit völlig aus.
Dieses Unterkapitel widmet sich der Frage, wie wahrscheinlich es ist, durch die ILS_r Heuristik bei gegebener Suchtiefe eben dieses globale Optimum tatsächlich aufzufinden.
Da die Enumeration für n≥13 zu aufwändig wäre, betrachten wir zunächst die Fälle für n≤11 und r=1,…,10. Dazu wurden für alle Parameterkonstellationen 100 Instanzen simuliert und mittels vollständiger Enumeration exakt gelöst. Die heuristische Erfolgswahrscheinlichkeit wurde durch den Anteil der dabei erfolgreichen heuristischen Lösungsversuche geschätzt:

n	Erfolgswahrscheinlichkeiten			
	r=1	r=2	r=5	r=10
4	1	1	1	1
5	1	1	1	1
6	0,97	0,99	1	1
7	0,94	0,98	1	1
8	0,88	0,95	0,99	1
9	0,85	0,89	0,94	1
10	0,74	0,82	0,96	0,99
11	0,68	0,83	0,87	0,97
12	0,53	0,7	0,85	0,93

Tabelle 29: Geschätzte Erfolgswahrscheinlichkeiten der Heuristik

Analog zu Abschnitt 3.1.3.1 halten wir zunächst r=1 fest:

Aus den Daten wird nun ein Trend geschätzt, der es ermöglichen soll, für r=1 auch für Instanzen mit n>11 Aussage über die Erfolgswahrscheinlichkeit zu treffen.
Ein logistisches Regressionsmodell wird den Daten zugrunde gelegt um die Wahrscheinlichkeit zu schätzen.
Nach wahrscheinlichkeitstheoretischer Annahme ist ein Wahrscheinlichkeitsmaß p eine Funktion p: $\Re \rightarrow [0;1]$ wobei $\Re$ eine σ-Algebra auf Ω, der Grundmenge ist. p bildet also auf [0;1] ab. Dieses Intervall wird durch den Begriff der „Odds“ auf das Intervall [0;∞[expandiert:

$$\text{Odds}(p) = \frac{p}{1-p} \qquad \rightarrow \qquad \text{Odds} \in [0;\infty[$$

Die Odds werden durch die natürliche Logarithmusfunktion ln() auf ganz R transformiert:

$$\ln(\text{Odds}(p)) = \ln\left(\frac{p}{1-p}\right) \qquad \rightarrow \qquad \ln(\text{Odds}(p)) \in R$$

Die ln(Odds) der Erfolgswahrscheinlichkeit p der ILS Heuristik werden nun linear mit der Instanzgröße n in Zusammenhang gebracht:

$$\ln(\text{Odds}(p)) = a + b*n \ (+\text{Zufallsfehler})$$

Die Koeffizienten a und b dieses nun *linearen* Modells werden mittels Kleinstquadrate-Schätzung ermittelt und als $\hat{a}$ und $\hat{b}$ bezeichnet. Durch Umformung

ergeben sich aus den geschätzten Koeffizienten die für das Modell geschätzten Wahrscheinlichkeiten $\hat{p}$:

$$\ln\left(\frac{\hat{p}}{1-\hat{p}}\right) = \hat{a} + \hat{b} * n$$

$$\leftrightarrow \quad \frac{\hat{p}}{1-\hat{p}} = \exp(\hat{a} + \hat{b} * n)$$

$$\leftrightarrow \quad \hat{p} = \exp(\hat{a} + \hat{b} * n) - \exp(\hat{a} + \hat{b} * n) * \hat{p}$$

$$\leftrightarrow \quad \hat{p} + \exp(\hat{a} + \hat{b} * n) * \hat{p} = \exp(\hat{a} + \hat{b} * n)$$

$$\leftrightarrow \quad \hat{p} = \exp(\hat{a} + \hat{b} * n)/(1-\exp(\hat{a} + \hat{b} * n))$$

Das Ergebnis dieser logistischen Regression ist:

Parameter	Wert
$\hat{a}$ =	6,52096
$\hat{b}$ =	-0,53607
R^2=	0,9853

Tabelle 30: Modellparameter des Erfolgswahrscheinlichkeiten-Modells

Das Bestimmtheitsmaß $R^2 = 0.9853$ zeugt von brauchbarer Anpassung. Abbildung 25 zeigt den geschätzten Verlauf der Wahrscheinlichkeiten in Abhängigkeit von n und die Güte der Anpassung des logistischen Regressionsmodells an die geschätzten Erfolgswahrscheinlichkeiten:

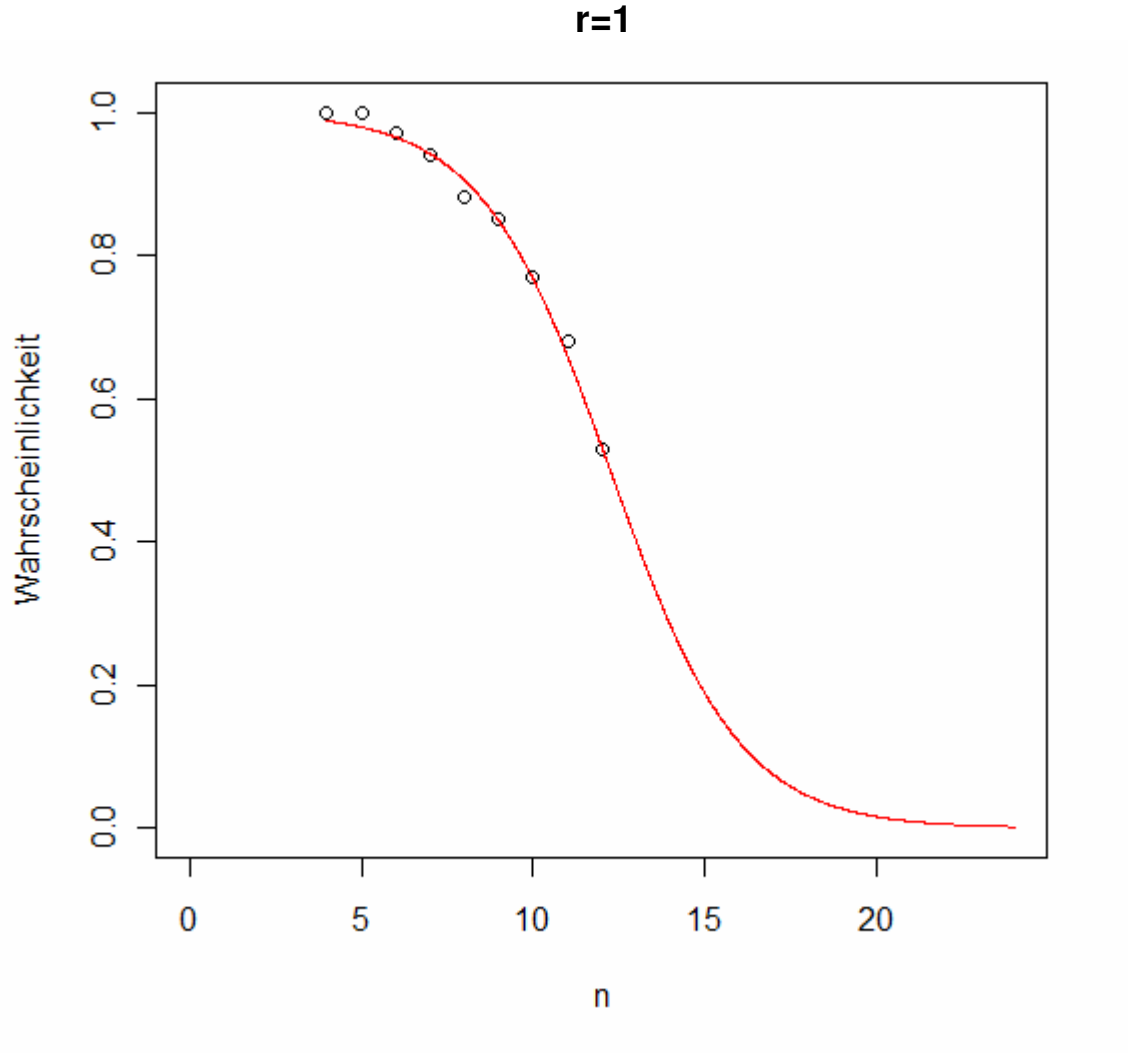

Abbildung 25: Modellfit des Erfolgswahrscheinlichkeiten-Modells für r=1

Tabelle 31 zeigt in übertriebener Präzision, die durch das Modell geschätzten Erfolgswahrscheinlichkeiten für r=1 in Abhängigkeit von n:

n	p
1	0,997431086465947000000000000000
2	0,995642240727185000000000000000
5	0,978971420325220000000000000000
10	0,766610478887239000000000000000
20	0,016088519358586600000000000000
30	0,000081393804705818800000000000
40	0,000000405222148573787000000000
50	0,000000002017251285357180000000
80	0,000000000000000248862783401519
100	0,000000000000000000006167274172
150	0,000000000000000000000000000000
200	0,000000000000000000000000000000

Tabelle 31: Geschätzte Erfolgswahrscheinlichkeiten für größere n

Im Fall r=10 ergibt die logistische Regression:

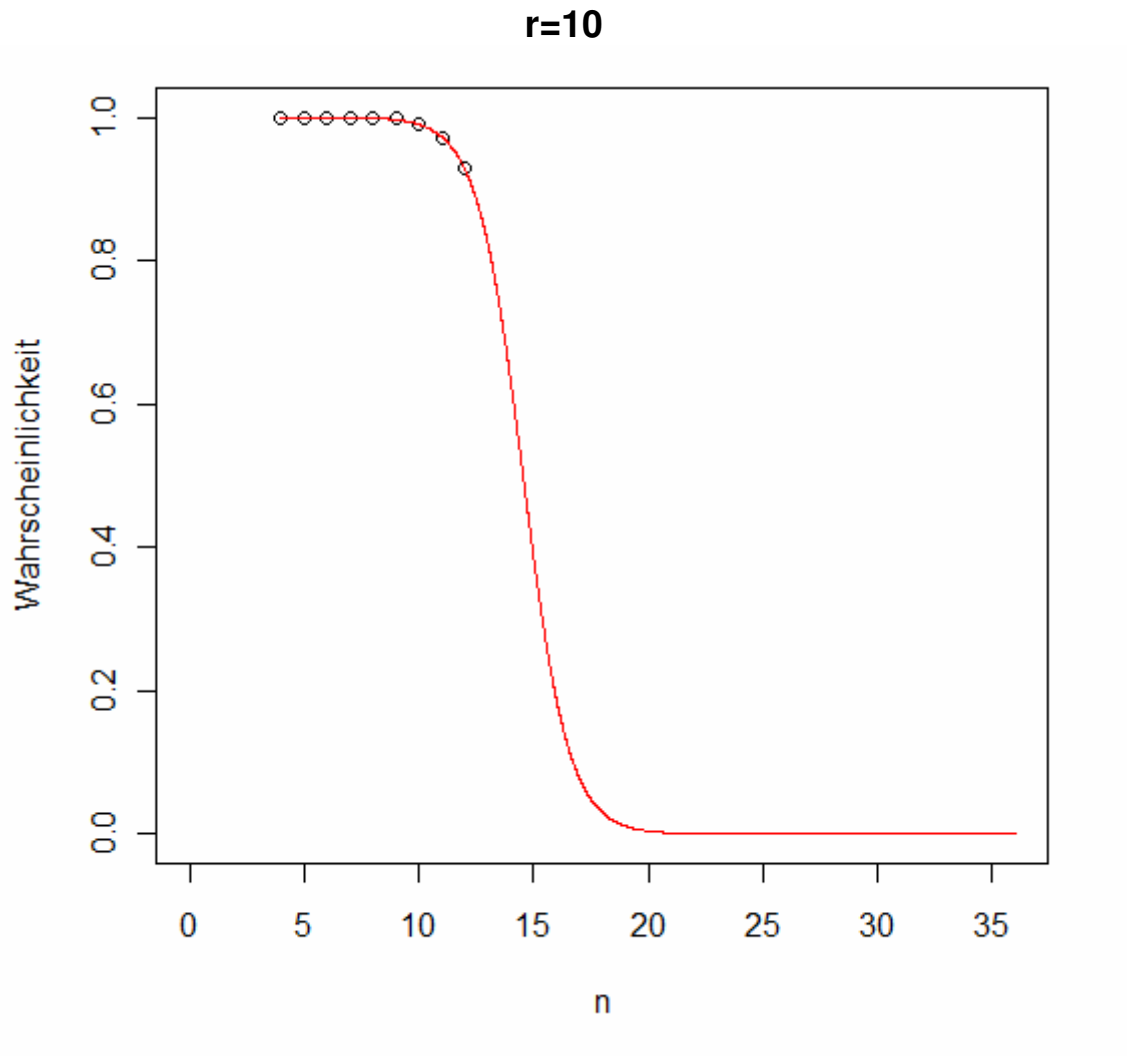

Abbildung 26: Modellfit des Erfolgswahrscheinlichkeiten-Modells für r=10

Parameter	Wert
$\hat{a}$ =	14.59900
$\hat{b}$ =	-1.00422
R^2=	0.9957

Tabelle 32: Modellparameter

Die Prognosen für r=1 bzw. r=10 zeigen, dass die Wahrscheinlichkeiten dass die Heuristik wirklich das Optimum trifft offenbar bereits für Instanzen um n=20 sehr klein werden.

Die ILS_r Heuristik mit variabler Suchtiefe scheint also für größere Instanzen nicht geeignet, tatsächlich optimale Lösungen zu finden.

3.1.5.2. Wahrscheinlichkeitstheoretischer Ansatz

Für den empirischen Ansatz muss also für jedes r ein eigenes Modell geschätzt werden. In Abschnitt 3.1.5.1. wurde dies für r=1 und r=10 berechnet. Um diese Wahrscheinlichkeiten in einer noch allgemeineren Art und Weise schätzen zu können betrachten wir zunächst die geschätzten Erfolgswahrscheinlichkeiten für die ILS_r Heuristik mit r=0, also einer lokalen Suche die unabhängig von Erfolg oder Misserfolg genau *einmal* gestartet wird (entspricht VND):

n	Erfolgswahrsch
4	1
5	0,991
6	0,928
7	0,827
8	0,742
9	0,622
10	0,395

Tabelle 33: Erfolgswahrscheinlichekiten für r=0

Inhaltlich bedeutet dies, dass die einmalige lokale Suche für ein TSP(10) mit 39,5%iger Wahrscheinlichkeit auch tatsächlich zum globalen Optimum führt. Die Lösungsraumbeschaffenheit bei n=10 dürfte also so aussehen, dass ca. 40% der möglichen Permutationen im Einzugsbereich des globalen Minimums liegen.
Im Folgenden werden wieder mittels logistischer Regression die Erfolgswahrscheinlichkeiten für n>10 geschätzt:

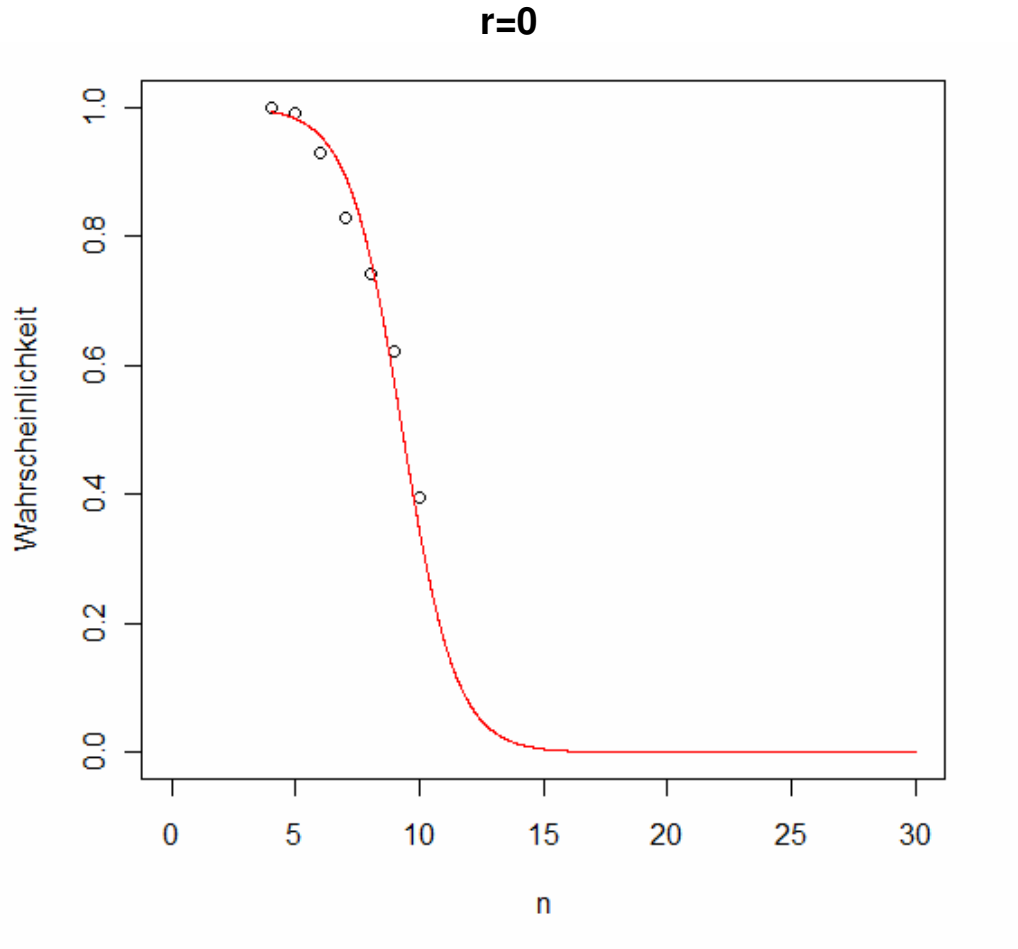

Abbildung 27: Modellfit für r=0

Parameter	Wert
$\hat{a}$ =	8,5846
$\hat{b}$ =	-0,9235
R^2=	0,9252

Tabelle 34: Modellparameter für r=0

Dieses Modell ermöglicht wieder, Wahrscheinlichkeiten p für n>10 zu prognostizieren

n	Erfolgswahrsch
11	0,171673
12	0,076047
13	0,031652
14	0,012814
15	0,005129
16	0,002043
17	0,000812
18	0,000323
19	0,000128
20	0,000050

Tabelle 35: Prognostizierte Erfolgswahrscheinlichekiten für r=0

Näherungsweise gilt Allgemein für r=0:

$$\hat{p} = \frac{\exp(8{,}5846 - 0{,}9252 * n)}{1 + \exp(8{,}5846 - 0{,}9252 * n)}$$

$\hat{p}$ ist also die (geschätzte) Wahrscheinlichkeit von einer zufälligen Startlösung aus mittels einfacher Neighborhood-Search das globale Optimum zu finden.
Nehmen wir an, wir starten zwei solche Suchläufe an zufälligen, voneinander unabhängigen Startpunkten. Die Zufallsvariable Z_2 beschreibt die Anzahl der dabei erzielten Erfolge. Es gilt:

$$P(Z_2>0) = 1 - (1- \hat{p})^2$$

Allgemeiner:

$$P(Z_n>0) = 1 - (1- \hat{p})^n$$

Sei außerdem X_r die Zufallsvariable, die die Anzahl der durchgeführten lokalen Suchen der ILS_r Heuristik beschreibt. Dann ist
X_r~negativ-binomial(1- $\hat{p}$;r), da sie unter der Unabhängigkeitsannahme für die zufälligen Startpunkte die Anzahl unabhängiger Bernoulliexperimente bis zum r-ten Misserfolg zählt. Somit gilt:

$$p(X_r = n) = \binom{n-1}{r-1} * \hat{p}^r * (1-\hat{p})^{n-r}$$

Die Erfolgswahrscheinlichkeit für ein bestimmtes r, p_r(Erfolg), lässt sich dann als Kombination von Z_n und X_r ausdrücken:

$$p_r(Erfolg) = \sum_{n=0}^{\infty} p(Z_n > 0) * p(X_r = n)$$

$$= \sum_{n=0}^{\infty} [1-(1-\hat{p})^n] * \binom{n-1}{r-1} * \hat{p}^r * (1-\hat{p})^{n-r}$$

3.2. ILS – Variante B

3.2.1. Laufzeiten

Variante B der ILS setzt Kenntnis der optimalen Tour voraus. Sei x_{opt} das globale Minimum der TSP-Touren, dann wird eine konstante Schranke c derart gewählt, dass $c = x_{opt}^* (1+p/100)$, wobei p ein beliebiger Fehlerprozentsatz ist, der der Heuristik als Toleranz gewährleistet wird. Die Heuristik iteriert die lokale Suche solange das zwischenzeitlich beste gefundene lokale Minimum x_{approx} größer ist als c.
Da vollständige Enumeration nur bei TSP(n) für n<13 in akzeptabler Zeit Lösungen liefert, werden zur Auswertung der Laufzeiten der Variante B der ILS bereits bekannte, gelöste TSP-Instanzen herangezogen. Eine Forschergruppe der kombinatorischen Optimierung der Ruprecht-Karls-Universität Heidelberg bietet auf ihrer Internetseite[1] (TSP-LIB) konkrete (asymmetrische) Instanzen mit zugehörigen bereits ermittelten Optima an. Wir bedienen uns nun Instanzen der Größen n=34,45,56,65 um die Laufzeit der ILS Heuristik (bezüglich Variante B) zu messen. 100 ILS-Durchläufe haben folgende mittlere Laufzeiten ergeben:

ILS	p=60	p=50	p=40	p=30	p=20
n=34	0,0000	0,0000	0,0022	0,0085	0,1065
n=45	0,0026	0,0043	0,0136	0,1219	7,7442
n=56	0,0123	0,0423	0,6194	14,2713	2405,3480
n=65	0,0379	0,3610	8,2137	547,5366	>24h

Tabelle 36: Laufzeiten ILS – Variante B

[1] siehe „http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/“

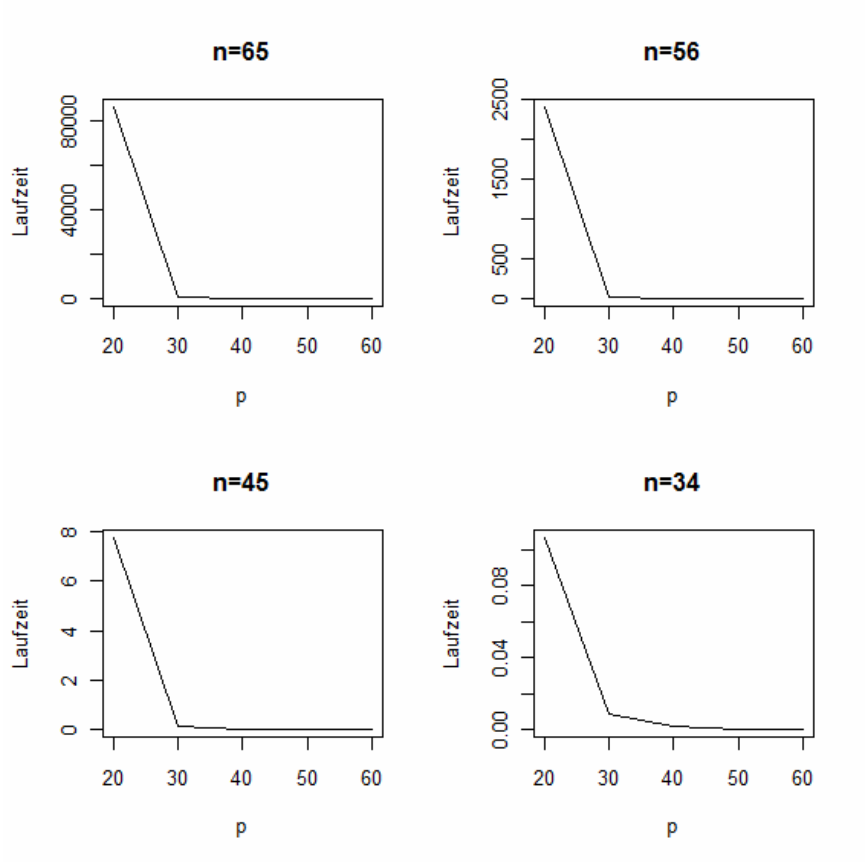

Abbildung 28: Laufzeiten ILS – Variante B

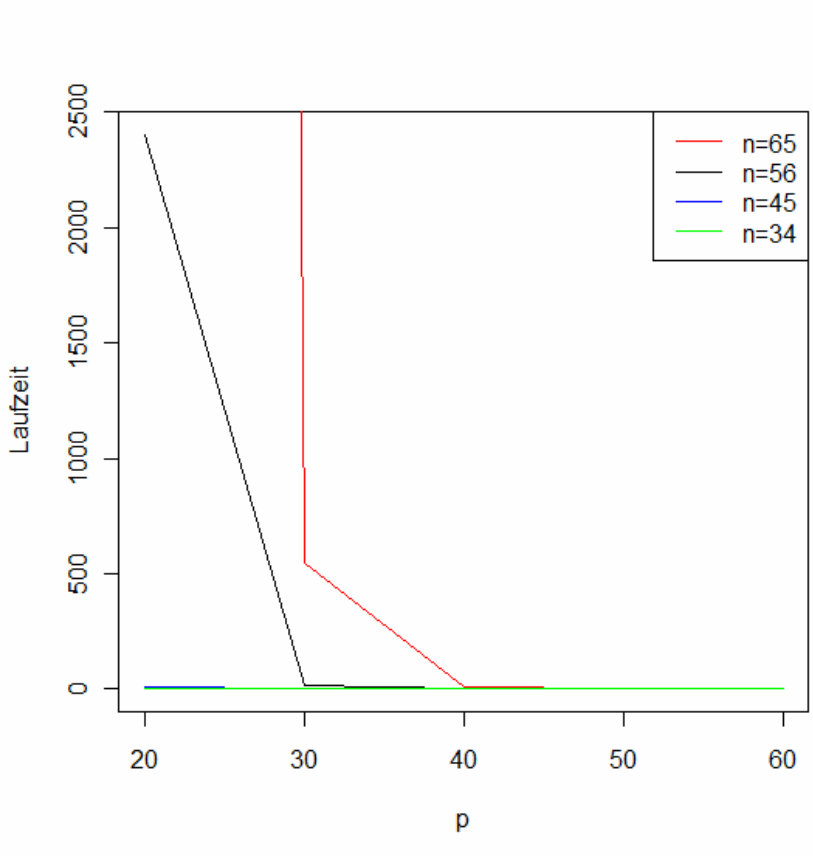

Abbildung 29: Vergleich der Laufzeiten ILS – Variante B

3.2.2. Die Verteilung der Laufzeiten

Es wurden Laufzeitstichproben der Größe N=1000 von ILS angewandt auf TSP(34) und TSP(45) simuliert. Die Histogramme weisen auf eine Dichte hin, die durch die Dichte der Exponentialverteilung gut gefittet zu werden scheint. Die über die Histogrammbalken gelegte Dichtefunktion ist jeweils eine Exponentialdichte mit aus den Daten geschätzten Mittelwerten.

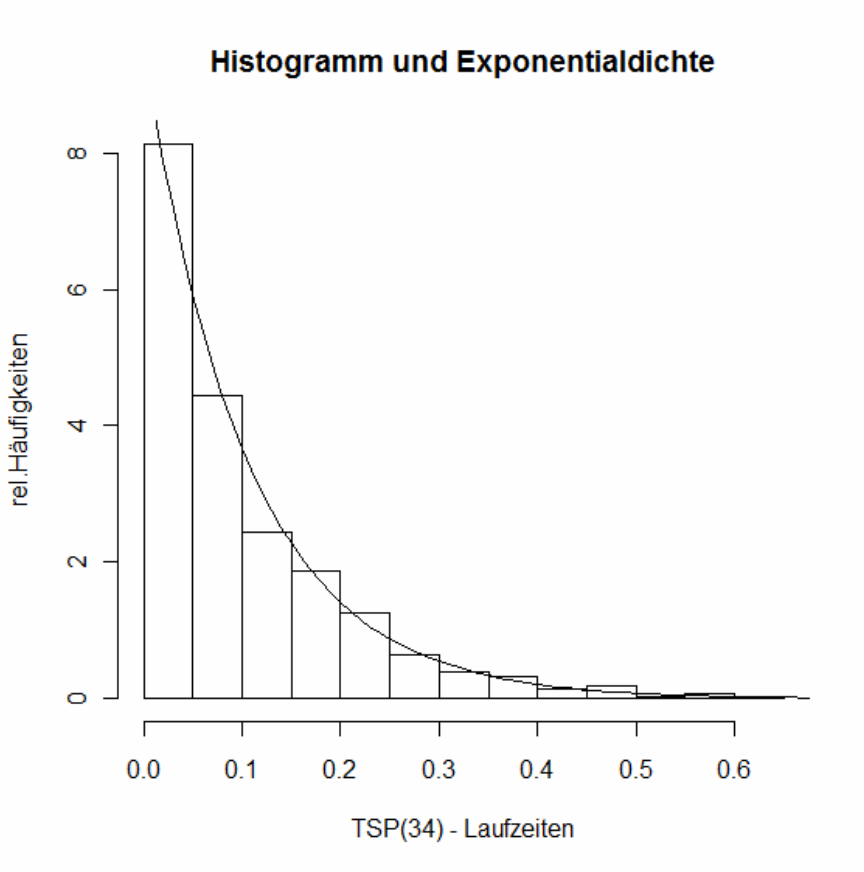

Abbildung 30: Fit der Exponentialdichte für n=34

Die Schätzung des Erwartungswerts durch das arithmetische Mittel beträgt 0,104805, die Varianzschätzung ergab 0,01112014 (die Varianz einer exponentialverteilten Zufallsvariable mit dem geschätzten Erwartungswert wäre 0,01098409)

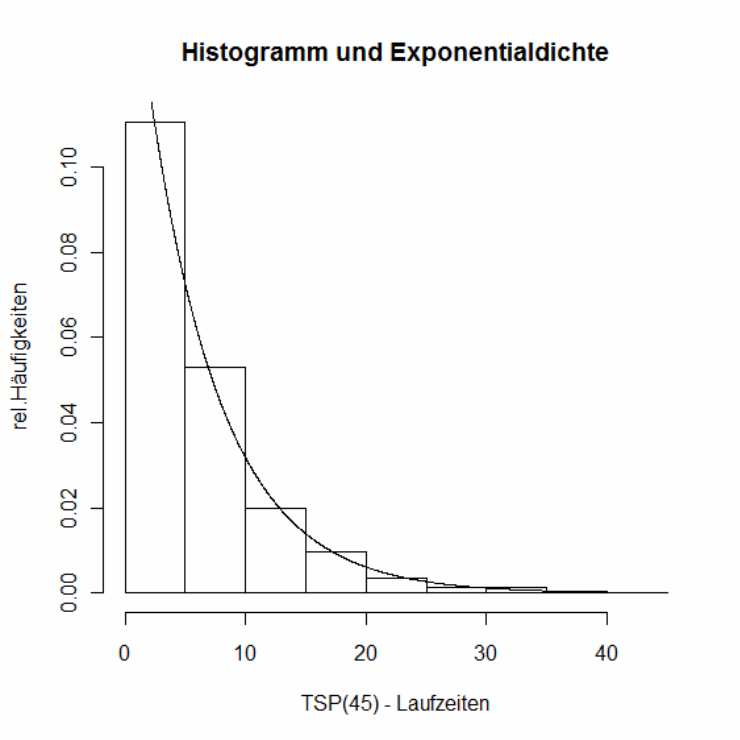

Abbildung 31: Fit der Exponentialdichte für n=45

Die Schätzung des Erwartungswerts durch das arithmetische Mittel beträgt 6,063, die Varianzschätzung ergab 35,68 (die Varianz einer exponentialverteilten Zufallsvariable mit dem geschätzten Erwartungswert wäre 36,76).
Der optische Eindruck wird auch aus statistischer Hinsicht in Form von K-S-Tests bestätigt. Es wurden 2 Anpassungstests durchgeführt, ein Einstichproben-Test, der testet ob die Stichprobe aus einer Exponentialverteilung mit aus der Stichprobe geschätztem Mittelwert stammt und ein Zweistichproben-Test, der eine zusätzliche Stichprobe aus einer Exponentialverteilung mit gleichem Mittelwert simuliert und testet, ob die beiden Stichproben aus der gleichen Verteilung stammen. Eine Bestätigung der Verteilungshypothese „Laufzeiten folgen einer Exponentialverteilung“ folgt aus dem nichtsignifikanten Testergebnis:

p-Werte	n=34	n=45
Einstichproben-Test	<0,05	0,594
Zweistichproben-Test	0,367	0,211

Tabelle 37: p-Werte der Anpassungstests

Die Laufzeiten der kleineren Instanz fallen also beim Einstichproben-Test noch durch, bei der größeren Instanz scheint sich die Lage derartig stabilisiert zu haben, dass keiner der beiden Tests die Hypothese auf Exponentialverteilung verwerfen kann.

4. AUSWERTUNG: SIMULATED ANNEALING

4.1. SA – Variante A

Für die Laufzeitanalysen und den Performancevergleich wird in den folgenden Kapiteln mit einer $SA_{0,5}$ –Heuristik gearbeitet.
Im Gegensatz zur Laufzeit der ILS_r Heuristik, die nach Kapitel 3 einerseits von der Instanzgröße n, andererseits vom heuristikspezifischen Suchtiefeparameter r abhängt, hängt die Laufzeit bei der auf Simulated Annealing basierenden Heuristik (SA) von drei Einflussgrößen ab:

- Von der TSP-Instanzgröße
- Von der Starttemperatur t
- Von der Temperaturaufenthaltsdauer d

Die Laufzeit von SA ist somit also beinahe deterministisch, da weder n, noch t, noch d zufällig sind. Die einzige stochastische Komponente, die die Laufzeit beeinflusst, ist, dass auch das Auffinden einer Verbesserungsrichtung bzw. das Update der Lösung durch die Toleranzwahrscheinlichkeit, eine Temperaturreduktion zur Folge hat und nicht nur das bloße Erreichen der Aufenthaltsdauer pro Temperaturstufe.
Fokus dieses Abschnitts wird es sein, eine Formel zu finden, die die Laufzeit l der SA Heuristik in Abhängigkeit dieser drei Größen möglichst gut prognostiziert. Das in Kapitel 2 vorgestellte Programm generierte dazu die Daten von Interesse aus simulierten symmetrischen, metrischen TSP-Instanzen. Nach Abschnitt 2.5 und 2.6 sind die Ergebnisse daraus auch für asymmetrische, nichtmetrische TSP gültig.

4.1.1. Die Abhängigkeit der Laufzeit von d, bei festem t und n

Sei vorerst n=50 bzw. 80 und t=100,200,400. So ergaben sich folgende mittlere Laufzeiten von Stichproben der Größe N=100:
n=50

t=100	d	100	200	300	400	500	600	700	800	900	1000	1600	3200
	Laufzeit	0,066	0,130	0,197	0,261	0,325	0,390	0,454	0,523	0,586	0,647	1,042	2,075
t=200	d	100	200	300	400	500	600	700	800	900	1000	1600	3200
	Laufzeit	0,135	0,268	0,403	0,534	0,668	0,805	0,939	1,077	1,200	1,343	2,137	4,283
t=400	d	100	200	300	400	500	600	700	800	900	1000	1600	3200
	Laufzeit	0,297	0,599	0,892	1,195	1,499	1,796	2,091	2,400	2,697	2,986	4,795	9,612

Tabelle 38: Laufzeiten n=50

n=80

t=100	d	100	200	300	400	500	600	700	800	900	1000	1600	3200
	Laufzeit	0,100	0,202	0,302	0,402	0,503	0,605	0,702	0,800	0,903	1,001	1,604	3,208
t=200	d	100	200	300	400	500	600	700	800	900	1000	1600	3200
	Laufzeit	0,208	0,420	0,623	0,823	1,031	1,244	1,445	1,654	1,855	2,062	3,304	6,598
t=400	d	100	200	300	400	500	600	700	800	900	1000	1600	3200
	Laufzeit	0,459	0,923	1,388	1,857	2,298	2,764	3,222	3,703	4,147	4,602	7,368	14,736

Tabelle 39: Laufzeiten n=80

Analog zur Vorgehensweise in Kapitel 3 betrachten wir zunächst wieder die Faktoren um bei jeweils festen n und t von einer Stufe d in die Nächste zu gelangen:

n=50	t=100	1,988	1,512	1,327	1,243	1,201	1,162	1,153	1,12	1,105	1,61	1,99
	t=200	1,985	1,506	1,324	1,251	1,204	1,167	1,148	1,114	1,119	1,591	2,004
	t=400	2,019	1,489	1,34	1,254	1,199	1,164	1,148	1,124	1,107	1,605	2,005

Tabelle 40: Faktoren für die nächste Aufenthaltsdauerstufe n=50

n=80	t=100	2,01	1,498	1,33	1,253	1,201	1,161	1,14	1,128	1,109	1,602	2
	t=200	2,025	1,482	1,322	1,252	1,207	1,162	1,144	1,122	1,112	1,602	1,997
	t=400	2,011	1,503	1,338	1,237	1,203	1,166	1,149	1,12	1,11	1,601	2

Tabelle 41: Faktoren für die nächste Aufenthaltsdauerstufe n=80

Diese so entstandenen sechs Messreihen werden nun wieder $\binom{6}{2}$=15 paarweisen, gepaarten t-Tests unterzogen, deren p-Werte sich in folgender Tabelle wieder finden:

		n=50			n=80		
		100	200	400	100	200	400
n=50	100	XXX	0,993	0,398	0,582	0,808	0,431
	200	XXX	XXX	0,419	0,599	0,808	0,498
	400	XXX	XXX	XXX	0,335	0,281	0,604
n=80	100	XXX	XXX	XXX	XXX	0,809	0,796
	200	XXX	XXX	XXX	XXX	XXX	0,724
	400	XXX	XXX	XXX	XXX	XXX	XXX

Tabelle 42: Paarweise gepaarte t-Tests

Da α=0,05 als obere Schranke für die Bonferroni-Adaptierung gesehen werden kann und sämtliche p-Werte >0,05 sind, muss das exakte Niveau für paarweise t-Tests gar nicht erst berechnet werden um aussagen zu können, dass der Test nie Nullhypothese „es gibt keine Unterschiede in den Faktoren über n und t hinweg betrachtet“ nicht verwerfen kann.

Somit ist also das wirken der Temperatur-Aufenthaltsdauer auf die Laufzeit unabhängig von n und t.
Eine kalkulativ „angenehme“ Eigenschaft, die beispielsweise auch beim Parameter r der ILS_r Heuristik auftrat.
In welcher Art und Weise sich d nun aber tatsächlich auf die Laufzeit auswirkt, soll im Folgenden untersucht werden. Streudiagramme lassen beinahe sicher vermuten, dass die Laufzeit linear in d wächst:

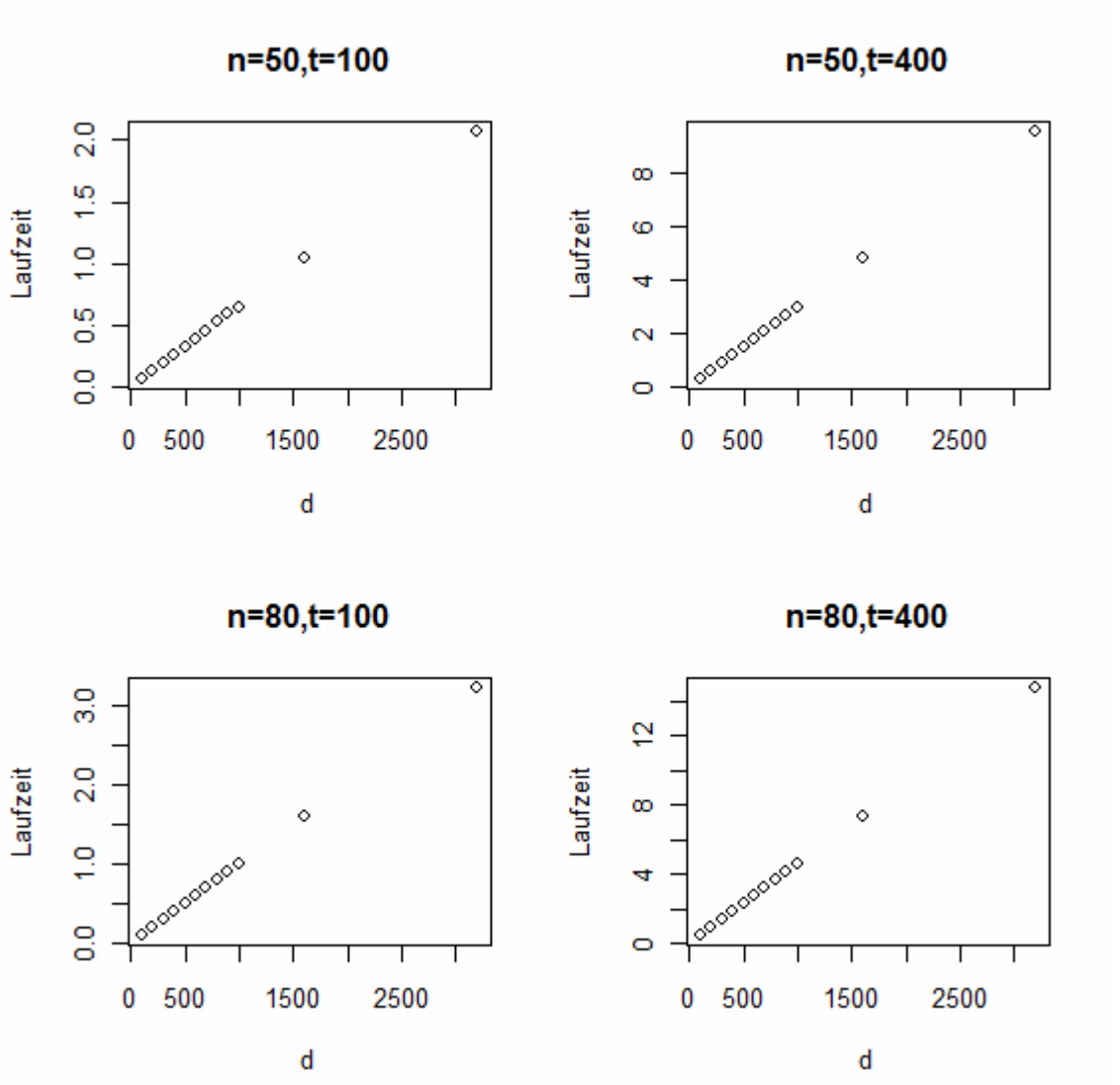

Abbildung 32: d vs. Laufzeit

Ein Blick auf diejenigen Faktoren q der Laufzeiten, die aus einer Verdoppelung von d resultieren, lässt erneut den Zusammenhang:

$$Laufzeit(k*d) = k*Laufzeit(d)$$

(Kurzschreibweise: $l_{d=D} * k = l_{d=D*k}$)

vermuten:

n=50	t=100	1,988	2,007	2,001	1,992	1,99
	t=200	1,985	1,994	2,017	1,984	2,004
	t=400	2,019	1,995	2,008	1,998	2,005

n=80	t=100	2,01	1,994	1,991	2,004	2
	t=200	2,025	1,959	2,008	1,998	1,997
	t=400	2,011	2,012	1,994	1,99	2

Tabelle 43: Verdoppelungsfaktoren n=50 und n=80

Der Mittelwert dieser Faktoren beträgt 1,999299. Da wir nun bereits wissen, dass das Wachstum der Laufzeit in Abhängigkeit von d unabhängig von n und t ist, wird dieser Faktor f bezüglich der Hypothese H_0: f=2, H_1: f≠2 nun wieder einem t-Test unterzogen, der mit einem p-Wert von 0,7642 H_0 nicht verwerfen kann.
Ein nicht unplausibles Resultat, da doppelte Verweildauer nichts anderes bedeutet, als doppelt so langer Aufenthalt in jeder Temperaturstufe. Dass die Laufzeiten sich dabei nicht exakt verdoppeln liegt an der Zufälligkeit, da der Wechsel auf ein niedrigeres Temperaturniveau nicht nur von der Verweildauer, sondern auch von den zufälligen Funktionswerten abhängt.

4.1.2. Die Abhängigkeit der Laufzeit von t und n

Für n=30,40,50,60,70,80 und festes d=100 wurden wieder SP der Größe N=100 gezogen um die Abhängigkeit der Laufzeit von t zu analysieren. Die Streudiagramme (hier nur für n=30,40,60,80) lassen wieder einen polynomialen$_k$ Zusammenhang vermuten.

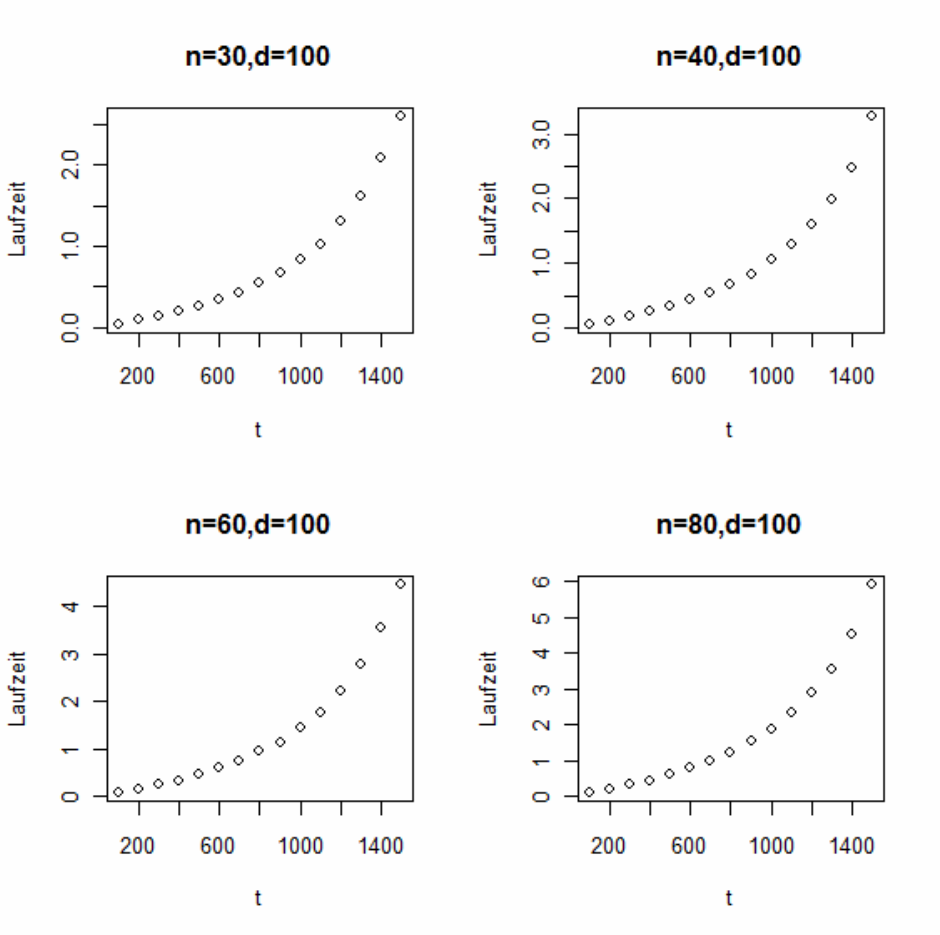

Abbildung 33: Starttemperatur vs. Laufzeit

Der Grad der optimalen Anpassung dieses Polynoms wird wieder mit dem bereits in Kapitel 3 verwendeten grid-search Verfahren ermittelt. Die jeweils optimale k bei bestimmten n ergeben sich dadurch wie folgt:

n	30	40	50	60	70	80
k	4,236	4,134	4,431	4,080	4,013	4,005

Tabelle 44: Optimale Potenzen

k≈4 und das (statistisch nicht explizit verifiziert, hier aber der Einfachheit angenommen) offenbar unabhängig von n.
Für n=30,40,50,60,70,80 werden nun lineare Modelle geschätzt:

$$laufzeit_n = (a_n + b_n * t)^4$$

Die Parameter a und b sind dabei erwartungsgemäß unterschiedlich bezüglich des jeweiligen n:

n	$\hat{a}_n$	$\hat{b}_n$
30	0,433	0,000539
40	0,460	0,000566
50	0,477	0,000592
60	0,502	0,000613
70	0,522	0,000636
80	0,538	0,000653

Tabelle 45: Modellparameter

Wollen wir nun betrachten, ob möglicherweise ein schätzbarer Zusammenhang zwischen n und a bzw. b besteht:

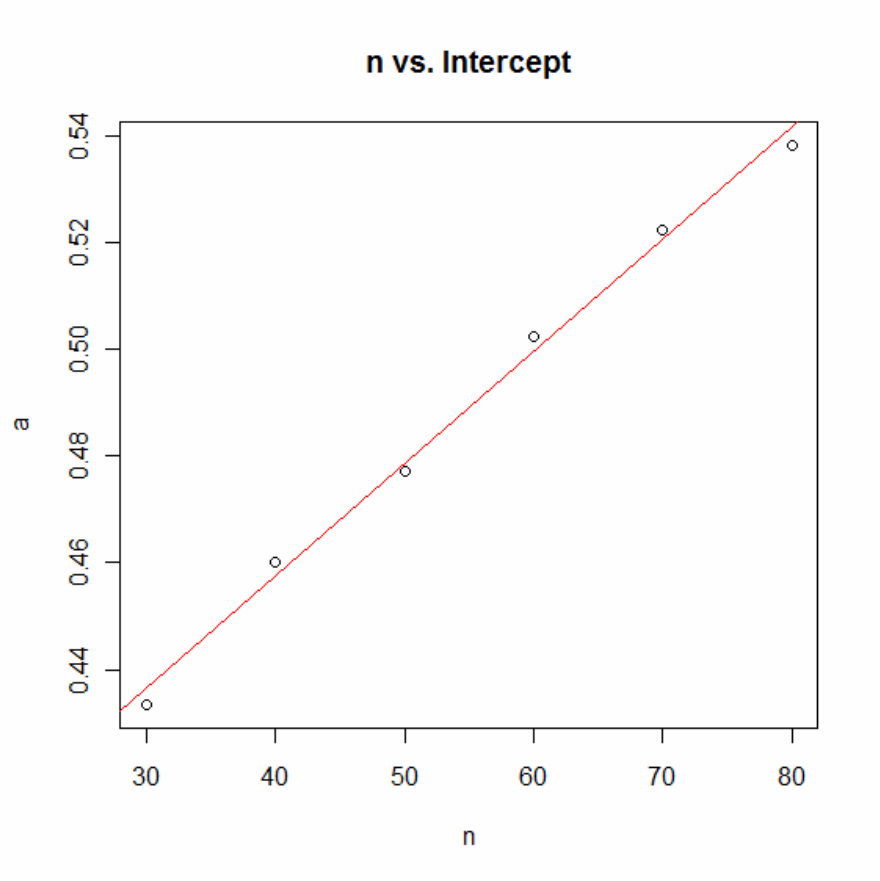

Abbildung 34: Prognose des Intercept

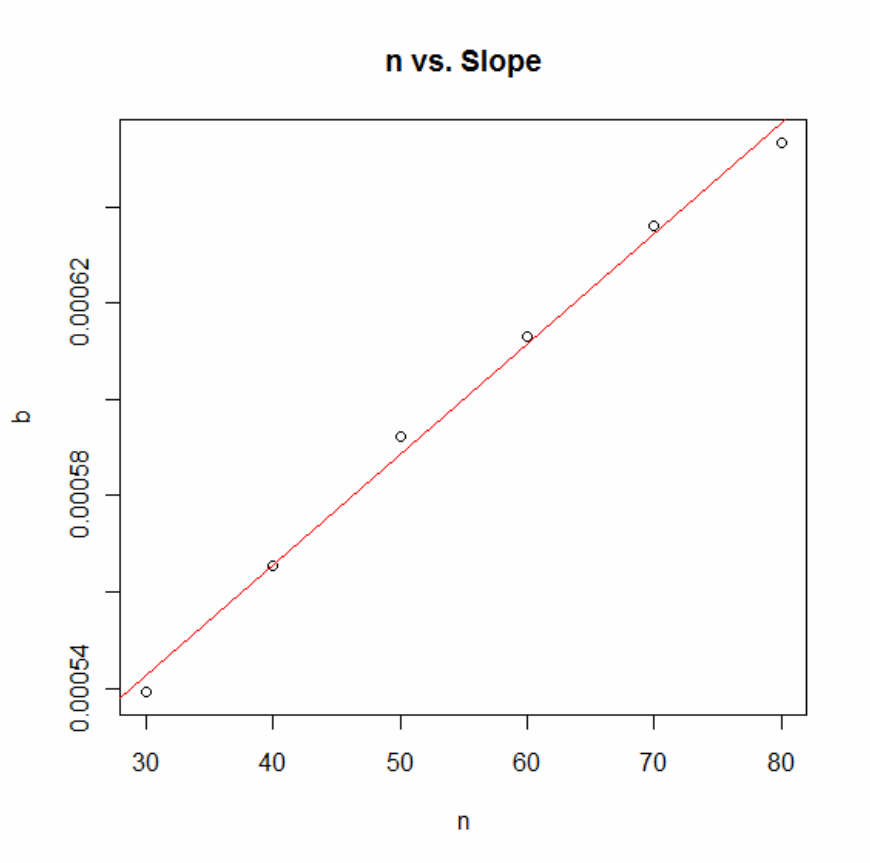

Abbildung 35: Prognose der Steigung

Die Modellparameter bzw. das Bestimmtheitsmaß für beide Modelle ergeben sich zu:

	$a=\alpha_a+\beta_a*n$	$b=\alpha_b+\beta_b*n$
$\alpha_{a,b}$	0,37324596	0,00047384
$\beta_{a,b}$	0,00210248	0,00000229
R^2	0,9948547	0,9949044

Tabelle 46: Modellparameter

Also:

$$a_n=0{,}37324596 + n * 0{,}00210248$$

$$b_n=0{,}00047384 + n * 0{,}00000229$$

$$\rightarrow l_{d=100} = (a_n + b_n*t)^4$$

$$\rightarrow l_{d=100} = [\ (0{,}37324596 + n * 0{,}00210248) + (0{,}00047384 + n * 0{,}00000229)*t\ ]^4$$

In Abschnitt 4.1.1. wurde für l in Abhängigkeit von d statistisch verifiziert, dass gilt

$$l_{d=100}*k = l_{d=100*k}$$

unabhängig von n und t.

Somit folgt insgesamt:

$$l = d/100 * [(0{,}37324596 + n * 0{,}00210248) + (0{,}00047384 + n * 0{,}00000229)*t)]^4$$

4.2. SA – Variante B

4.2.1. Laufzeiten

Analog zur Variante B bei ILS, ist auch die SA Variante B herkömmliches SA, das aber im Gegensatz zu Variante A nicht nach Erreichen einer bestimmten (tiefen) Temperatur abbricht, sondern wenn sich die approximative Lösung bis zu einem bestimmten Prozentsatz (Fehler) dem (bekannten) absoluten Minimum angenähert hat.

Sei also wiederum x_{opt} das globale Minimum der TSP-Touren, dann wird eine konstante Schranke c derart gewählt, dass $c = x_{opt}* (1+p/100)$, wobei p ein beliebiger Fehlerprozentsatz ist, der der Heuristik als Toleranz gewährleistet wird. SA wird beendet, sobald ein $x_{approx} < c$ gefunden wurde.

Dies setzt wiederum die Kenntnis des globalen Minimums voraus; wie in Abschnitt 3.2 wird dies durch die TSP-LIB Instanzen für n=34,45,56,65 gewährleistet.

Jeweils 100 SA-Durchläufe mit $t=10^{12}$, d=1000 und f=0.99999 wurden auf die Instanzen für p=20,30,40,50,60% angewandt:

SA	p=60	p=50	p=40	p=30	p=20
n=34	0,4031	0,4221	0,4431	0,4786	0,5589
n=45	0,4638	0,4892	0,5329	0,6010	0,8393
n=56	0,5145	0,5652	0,6579	0,9176	7,7187
n=65	0,5908	0,6750	0,8848	5,5510	25,0243

Tabelle 47: Laufzeiten SA - Variante B

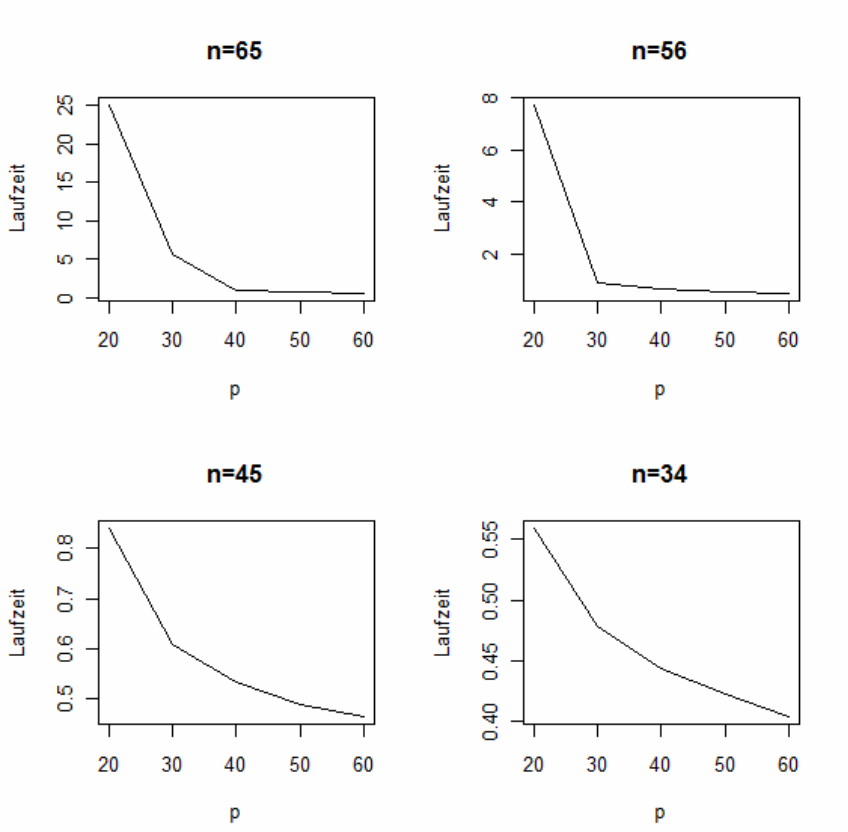

Abbildung 36:Laufzeiten SA-Variante B

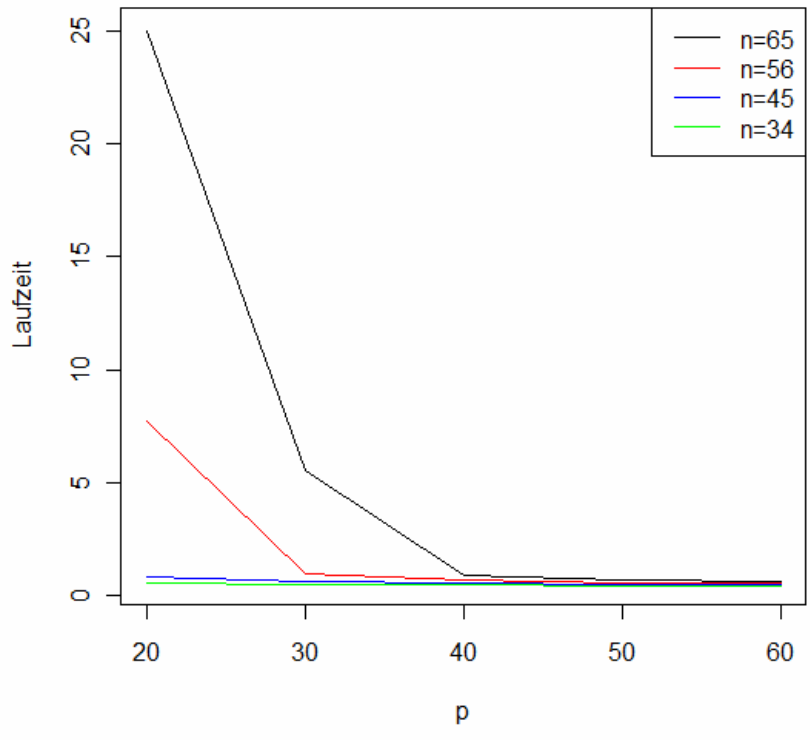

Abbildung 37: Vergleich der Laufzeiten SA-Variante B

Vergleich zu ILS:

ILS vs. SA	p=60	p=50	p=40	p=30	p=20
n=34	-	-	-	-	-
n=45	-	-	-	-	+
n=56	-	-	-	+	+
n=65	-	-	+	+	+

Tabelle 48: „+“ SA, „-„ILS

Ein „+“ in der Tabelle steht für besseres Abschneiden von SA im Gegensatz zu ILS. Es ist deutlich zu erkennen, dass SA vor allem dann effizienter wird, wenn es um die wirklich guten Lösungen geht. Dies ist wohl dadurch zu erklären, dass ILS in relativ kurzer Zeit erstmal irgendein lokales Minimum findet, das schon „relativ“ gut ist, während SA zu Beginn bei noch sehr hohen Temperaturen lediglich an der Oberfläche kratzt, nicht selten wegen zu diesem Zeitpunkt noch sehr hohen Trotzwahrscheinlichkeiten auch schlechtere Richtungen akzeptiert und erst bei niedrigeren Temperaturen beginnt wirklich in die Tiefe zu gehen.
Auffällig ist auch, dass der Übergangspunkt, an dem SA über ILS zu dominieren beginnt, mit steigender TSP-Dimension wächst.
Der wesentliche Effizienzvorteil liegt darin, dass bei ILS *alle* Nachbarn abgesucht werden, während SA *einen* zufälligen Nachbarn auswählt und gemäß der Zielfunktionswertdifferenz bzw. gegebenenfalls der Trotzwahrscheinlichkeit akzeptiert oder nicht.

4.2.2. Die Verteilung der Laufzeiten

Analog zu Abschnitt 3.2.2 wird nun auch die Verteilung der Laufzeiten der Variante B des SA untersucht. Dabei erweisen sich die Histogramme bezüglich einer ersten Verteilungsvermutung zunächst als weniger offensichtlich:

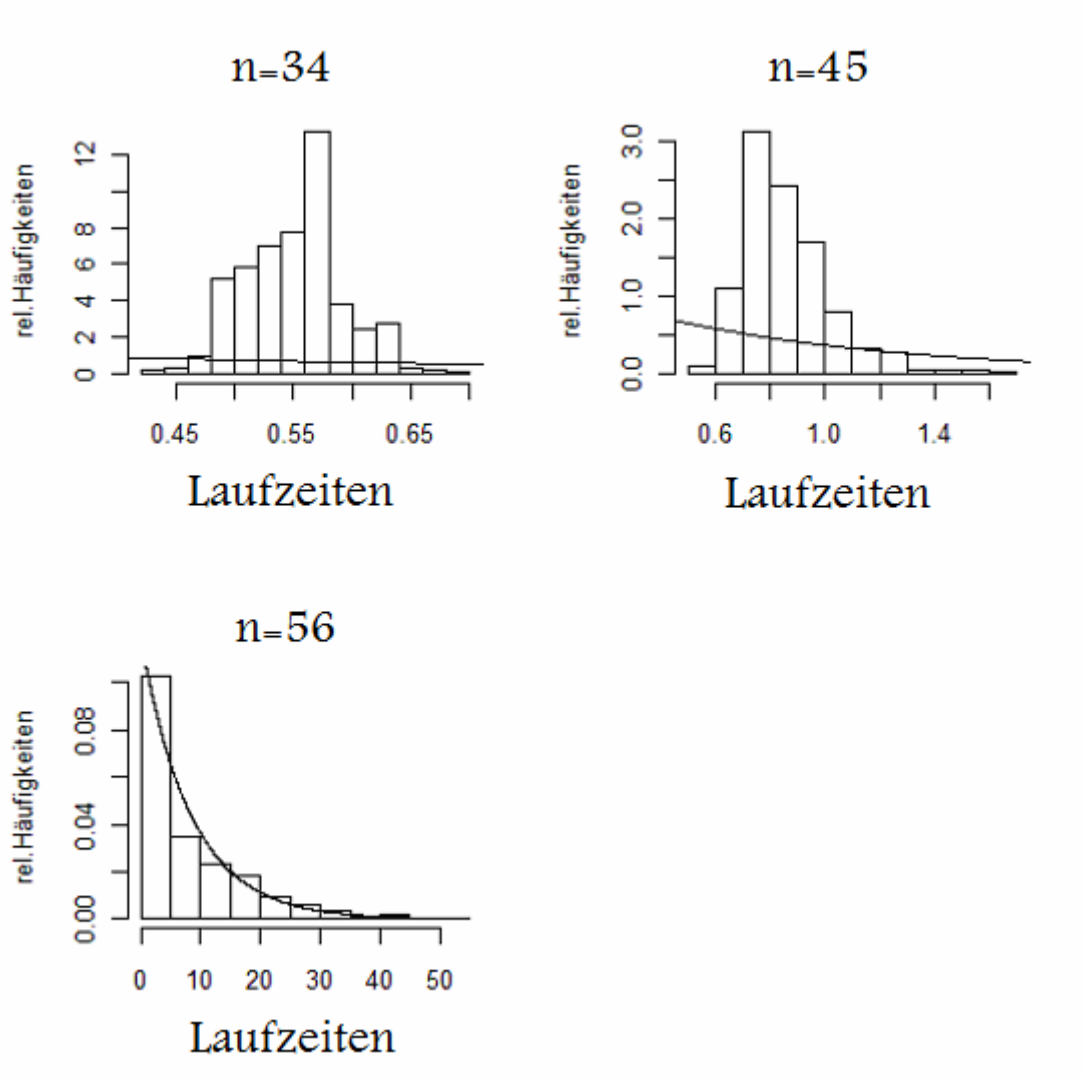

Abbildung 38: Histogramme mit vermuteter Exponentialdichte

Die Mittelwerte und Varianzen für n=34 / n=45 / n=56 ergaben sich zu: 0,5465 / 0,8561 / 7,7187 bzw. 0,001609 / 0,02611 / 59.56821 (die durch die vermutete Exponentialverteilung erwarteten Varianzen wären 0,2987 / 0,7329 / 55,952)

Die vermutete Exponentialverteilung scheint bei n=34 ganz und gar nicht zu passen. Mit wachsendem n gelingt der Fit bei n=45 schon besser, allerdings immer noch alles

andere als zufrieden stellend. Erst bei n=56 wird das aus dem ILS-Kapitel gewohnt gute Ergebnis erzielt. Die Verteilung scheint bei SA also erst für größere n gegen eine Exponentialverteilung zu streben. Inhaltlich macht das durchaus Sinn, dass die Masse links des Modus mit wachsendem n langsam verschwindet, da sie so etwas wie „das zufällige, schnelle Auffinden einer bereits hinreichend guten Lösung“ verkörpert. Dies ist bei kleineren Instanzen beim SA durchaus noch möglich und wahrscheinlich, da SA – Variante B *jederzeit* abbricht, sobald sich der momentane Lösungswert im Toleranzbereich befindet, hingegen ILS auf jeden Fall erst ein lokales Minimum findet und dann erst dessen mögliche Lage im Toleranzbereich überprüft, was die wesentlich frühere Verteilungsstabilisierung zur Folge hat.
Diese Hypothese wird auch quantitativ wieder durch die beiden bereits in Abschnitt 3.2.2. vorgestellten Kolmogorov-Smirnoff-Anpassungstests untermauert:

p-Werte	n=34	n=45	n=56
Einstichproben-Test	<0,05	<0,05	0,221
Zweistichproben-Test	<0,05	<0,05	0,281

Tabelle 49: p-Werte der Anpassungstests

Wie vermutet wird die Hypothese „Die Laufzeiten folgen einer Exponentialverteilung“ für n=34 und n=45 von beiden Tests glatt verworfen, während keiner der beiden Tests bei n=56 signifikant gegen diese Hypothese sprechen konnte.

5. VERGLEICH ILS_r UND $SA_{0,5}$

Während der Fokus in Kapitel 3 und 4 auf den Laufzeiten lag, sind nun die heuristisch ermittelten Funktionswerte von Relevanz. Der Faktor „Metrik“ hat zwar Einfluss auf die Höhe der Funktionswerte (Tabelle 11), dieser Einfluss scheint allerdings nur eine konstante Verschiebung (unabhängig von der Heuristik) der Lösungswerte zu sein und ist somit zu vernachlässigen. Der Faktor „Symmetrie“ wirkt sich unterschiedlich auf die ermittelten Lösungswerte der beiden Heuristiken aus und muss somit gesondert behandelt werden.

5.1. Relativer Performancevergleich

Beim relativen Performancevergleich werden die beiden Heuristiken auf dieselben Probleminstanzen angewandt und die Höhe der ermittelten Lösungswerte miteinander verglichen.

5.1.1. Symmetrische TSP

Tabelle 50 stellt die Mittelwerte von SP der Größe N=100 bei n=50, d=100 und k=5 dar. Es wurden also 100 zufällig generierte symmetrische TSP(50) Instanzen sowohl mit ILS_5 als auch mit $SA_{0,5}$ gelöst:

	ILS_5		$SA_{0,5}$	
	Zeit	Wert	Zeit	Wert
t=0,000000005	0,2830	2,3830	0,0003	10,0674
t=0,00000005	0,2633	2,3891	0,0005	9,9064
t=0,0000005	0,2776	2,4027	0,0006	8,4149
t=0,000005	0,2719	2,3583	0,0014	7,7150
t=0,00005	0,2829	2,4331	0,0014	7,0984
t=0,0005	0,2841	2,3625	0,0023	6,7958
t=0,005	0,2710	2,4002	0,0023	6,7488
t=0,5	0,2934	2,3971	0,0039	6,3023
t=1	0,2757	2,3874	0,0034	6,3020
t=10	0,2608	2,4549	0,0085	6,4638
t=25	0,2749	2,4166	0,0178	6,3977
t=50	0,2762	2,4501	0,0331	6,4316
t=100	0,2850	2,3510	0,0649	6,3240
t=500	0,2776	2,3867	0,4012	6,4797
t=1000	0,2648	2,3865	1,2367	6,3563
t=1500	0,2836	2,4110	3,8388	6,2857

Tabelle 50: ILS_5 vs. SA, n=50

Auffällig ist dabei nicht nur, dass die ILS_5 Heuristik für die meisten t in kürzerer Zeit einen besseren Lösungswert liefert, sondern auch, dass der letztendlich gefundene Wert offenbar erst für kleine t wirklich beginnt von t abzuhängen.
Dies liegt daran, dass eine hohe Starttemperatur Anfangs sehr hohe Wahrscheinlichkeiten zur Folge hat, was wiederum bedeutet, dass fast immer auch in Verschlechterungsrichtungen gegangen wird. Somit ist die Suche zu Beginn ein „planloses Umherirren“ unabhängig von den Differenzen der Funktionswerte. Erst wenn dann die Temperatur in Bereiche kommt, die halbwegs brauchbare Wahrscheinlichkeiten erzeugt (weit genug weg von 1), beginnt effizient gearbeitet zu werden, bis dann die fast einfrierende Temperatur Wahrscheinlichkeiten nahe bei 0 zur Folge hat und das Verfahren mehr oder wenig in eine gewöhnliche VND. Wann dieser „effiziente“ Bereich beginnt, hängt natürlich auch von der Differenz der Funktionswerte ab, die wiederum natürlich von der Instanzgröße n, als auch von der Beschaffenheit der Distanzmatrix an sich (sieh Kapitel 2) abhängt. Für den Fall n=50 und eine Distanzmatrix wie in Kapitel 2 beschrieben, beginnt dieser Bereich offenbar ca. bei t=0,0005.
Folgende Grafik zeigt die Funktion der Trotzwahrscheinlichkeiten, also die Wahrscheinlichkeit eine Verschlechterungsrichtung einzuschlagen, in Abhängigkeit von t für feste Differenzen:

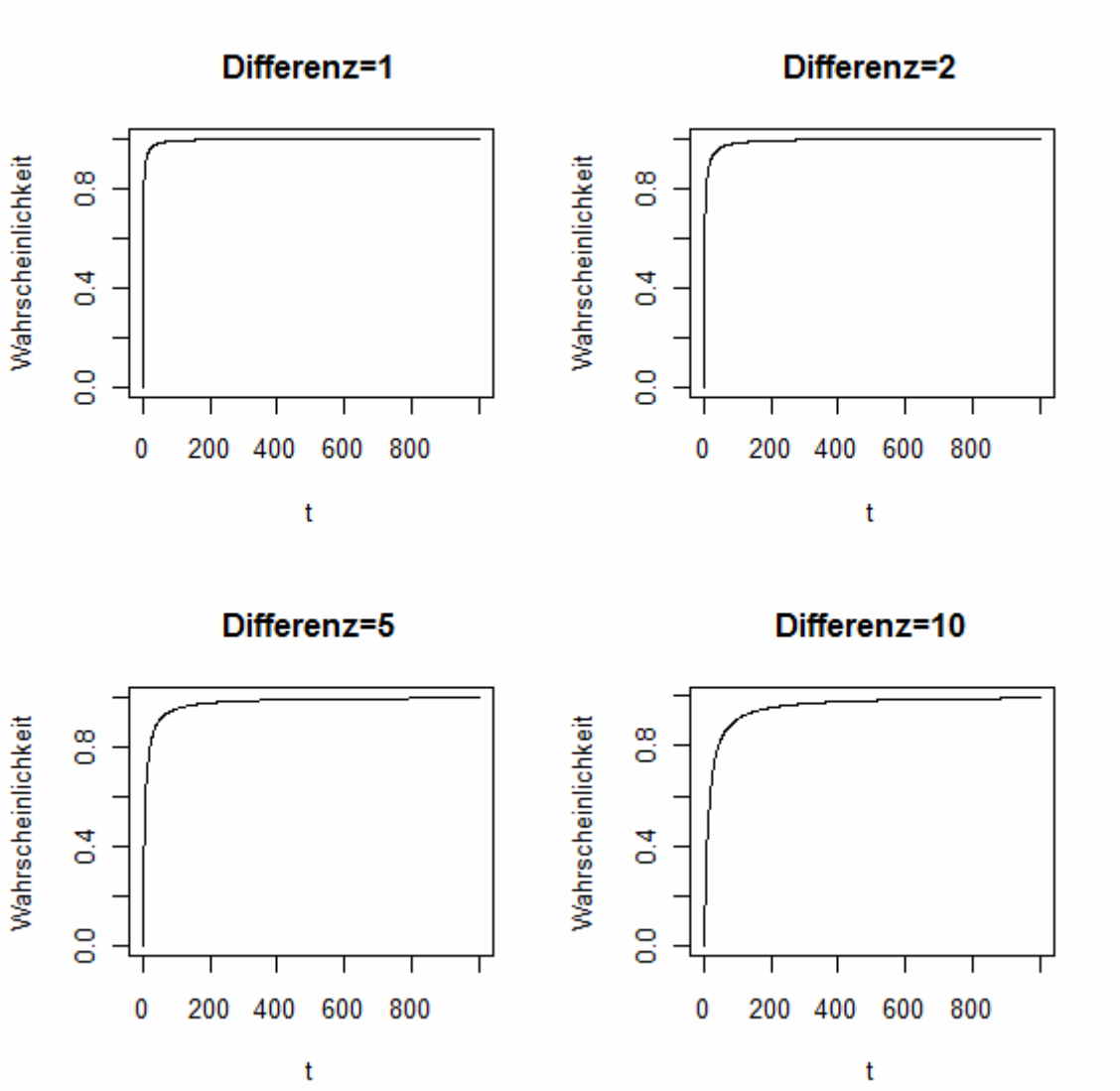

Abbildung 39: Trotzwahrscheinlichkeiten in Abhängigkeit von t

Durch Simulation[1] sind die Verteilungen der auftauchenden Differenzen zu veranschaulichen:

[1] Programmcode siehe Kapitel 7.3.2

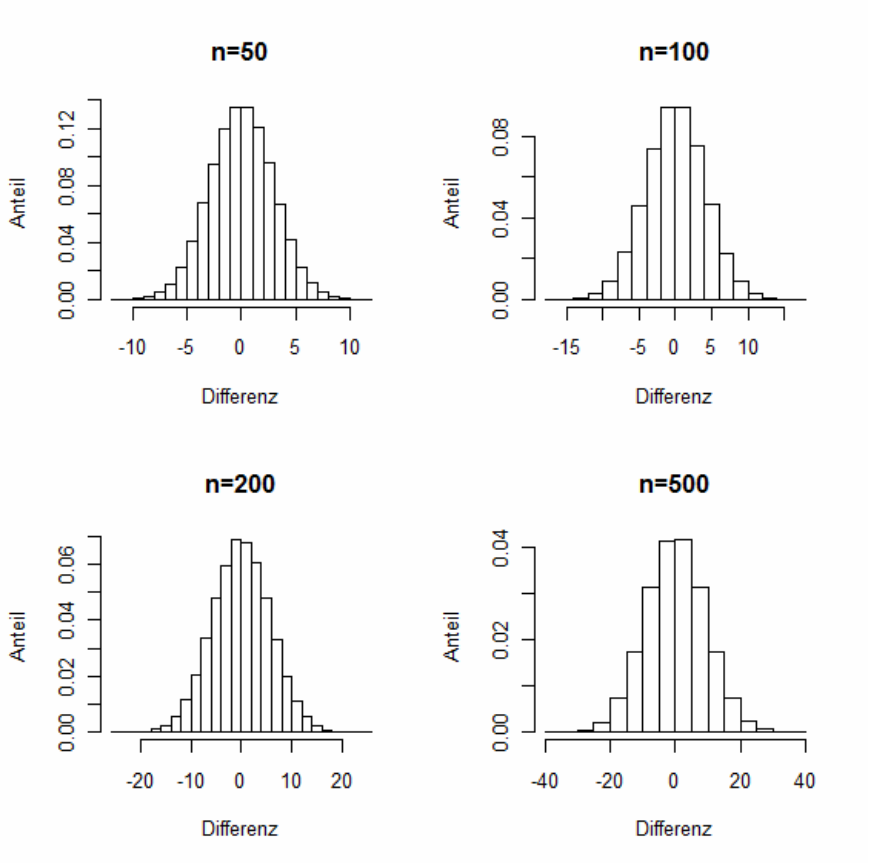

Abbildung 40: Histogramme der simulierten Differenzen

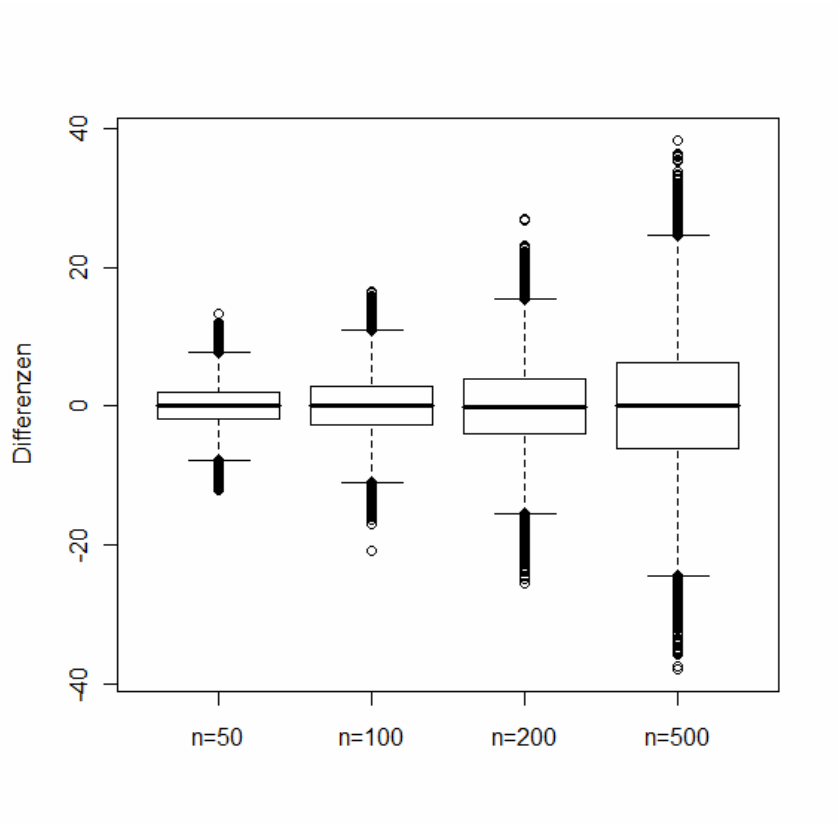

Abbildung 41: Boxplots der simulierten Differenzen

n	Min	0,25-Quantil	Median	Mittel	0,75-Quantil	Max
50	-11,870	-1,963	0,006	-0,007	1,942	12,250
100	-19,070	2,778	-0,011	-0,024	2,745	19,000
200	-24,810	-3,885	-0,031	-0,002	3,877	24,480
500	-42,290	-6,241	-0,006	-0,053	6,146	41,020

Tabelle 51: Deskriptive Statistiken der Funktionswertedifferenzen

Bei eben diesem t wird etwa durch $SA_{0,5}$ eine Lösung in ca. 1/100 des Zeitaufwands von ILS_5 gefunden, die allerdings ca. 2,5 mal so schlecht ist.
Hält man t=0,0005 nun fest und schraubt dann d ein wenig in die Höhe kann man eine ca. 1,5-mal so schlechte Lösung in beinahe derselben Zeit erzielen:

	2-opt		Annealing	
	Zeit	Wert	Zeit	Wert
d=10000	0,28118	2,3543	0,21383	4,4878
d=50000	0,26528	2,3707	1,0705	4,1511
d=100000	0,27878	2,3896	2,14053	4,0854

Tabelle 52: 2-opt vs. SA, t fest , n=50

Insgesamt scheint die $SA_{0,5}$ Heuristik allerdings zu mindest bei symmetrischen TSP(50) weniger gut abzuschneiden, als die ILS_5 Heuristik. Die Spanne der Lösungswerte für die ILS_5 Heuristik ist in folgender Grafik ersichtlich. Ein r=5 scheint angemessen. Mit wesentlich höherem r verlängert man die Laufzeit im Verhältnis zur Erfolgssteigerung zu sehr.

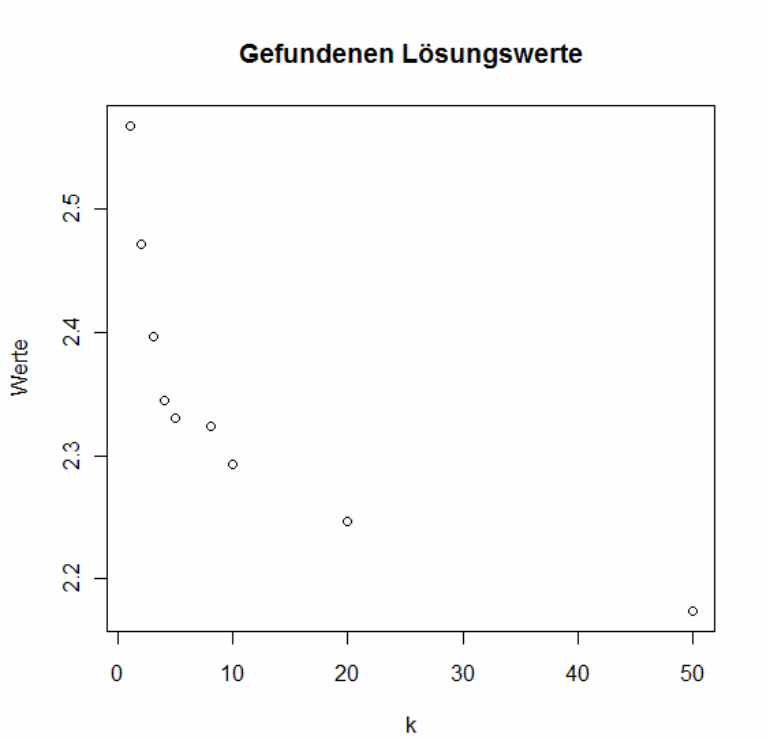

Abbildung 42: Lösungswerte in Abhängigkeit von r

Wieder wurden 100 TSP-Instanzen mit ILS_5 und $SA_{0,5}$ mit unterschiedlichen Starttemperaturen behandelt. Für n=80 ergaben sich die Laufzeiten und ermittelten Werte wie folgt:

	ILS		Annealing	
n=80	Zeit	Wert	Zeit	Wert
t=0,000000005	1,99908	2,7089	0,00015	22,9593
t=0,00000005	1,96987	2,6395	0,00078	17,7664
t=0,0000005	1,9458	2,6349	0,0017	14,7061
t=0,000005	1,9152	2,7344	0,0009	13,6855
t=0,00005	1,9269	2,6872	0,0023	12,2568
t=0,0005	1,9338	2,6323	0,0033	11,6831
t=0,005	1,9299	2,6806	0,0041	11,3378
t=0,05	1,9210	2,7065	0,0046	11,0313
t=0,5	1,9370	2,7037	0,0060	10,6827
t=1	1,97321	2,6731	0,00629	10,5701
t=10	1,92004	2,6424	0,01436	10,7103
t=25	1,92565	2,6897	0,02741	10,6548

Tabelle 53: ILS_r vs. SA, n=80

Für größeres n scheint der Punkt an dem die gewählte Starttemperatur kaum mehr Einfluss auf die ermittelten Werte zu haben scheint bei ca. 0,5 zu liegen. Bei festem t=0,5 kann durch Erhöhung der Verweildauer d die Effizienz (allerdings natürlich auch die Laufzeit) gesteigert werden:

	ILS		Annealing	
	Zeit	Wert	Zeit	Wert
d=1000	1,87306	2,6488	0,05104	7,941
d=10000	1,93119	2,6308	0,51335	6,9386
d=50000	1,86923	2,6736	2,56148	6,5592
d=100000	1,80534	2,6513	5,11975	6,3572
d=200000	1,87955	2,6863	10,24479	6,3119

Tabelle 54: ILS . Vs SA, t fest, n=80

Allerdings ist auch hier wieder zu sehen, dass eine Steigerung der Temperaturstufen-Aufenthaltsdauer ab einem gewissen Punkt den gefundenen Wert nur noch marginal verbessert. Im Bereich von d=50000 erreicht man so zu bei einer noch erträglichen, im Rahmen der ILS_5 liegenden Laufzeit einen ca. 2,5-mal so großen Wert. Im Folgenden bleiben wir daher bei einem t=1.

5.1.2. Asymmetrische TSP

Tabelle 55 zeigt den relativen Performancevergleich der beiden Heuristiken für asymmetrische TSP. 100 TSP wurden simuliert und von beiden Heuristiken näherungsweise gelöst. (k=5, t=1, d=1000) Auffällig ist hierbei, dass SA in kürzerer Zeit die besseren Werte liefert. Da die Laufzeit von ILS_r schneller in r wächst, als die Laufzeit von SA in d (Abbildung 43) und sogar die Wertequotienten über n hinweg zunehmen, macht das Rennen im Falle eines asymmetrischen TSP klar die $SA_{0,5}$-Heuristik.

	ILS		SA		
	Zeit	Wert	Zeit	Wert	Quotient
n=40	0,04918	7,7744	0,0321	4,0117	1,93793155
n=60	0,23541	11,8525	0,04445	5,9259	2,00011813
n=80	0,76071	16,0208	0,05643	7,957	2,01342214
n=100	1,82913	20,269	0,06782	9,8937	2,04867744

Tabelle 55: ILS vs. Annealing

Wie aus 2.6 bekannt ist der Faktor „Symmetrie“ für die SA Werte irrelevant, lediglich die ILS_5 Werte sind signifikant schlechter.

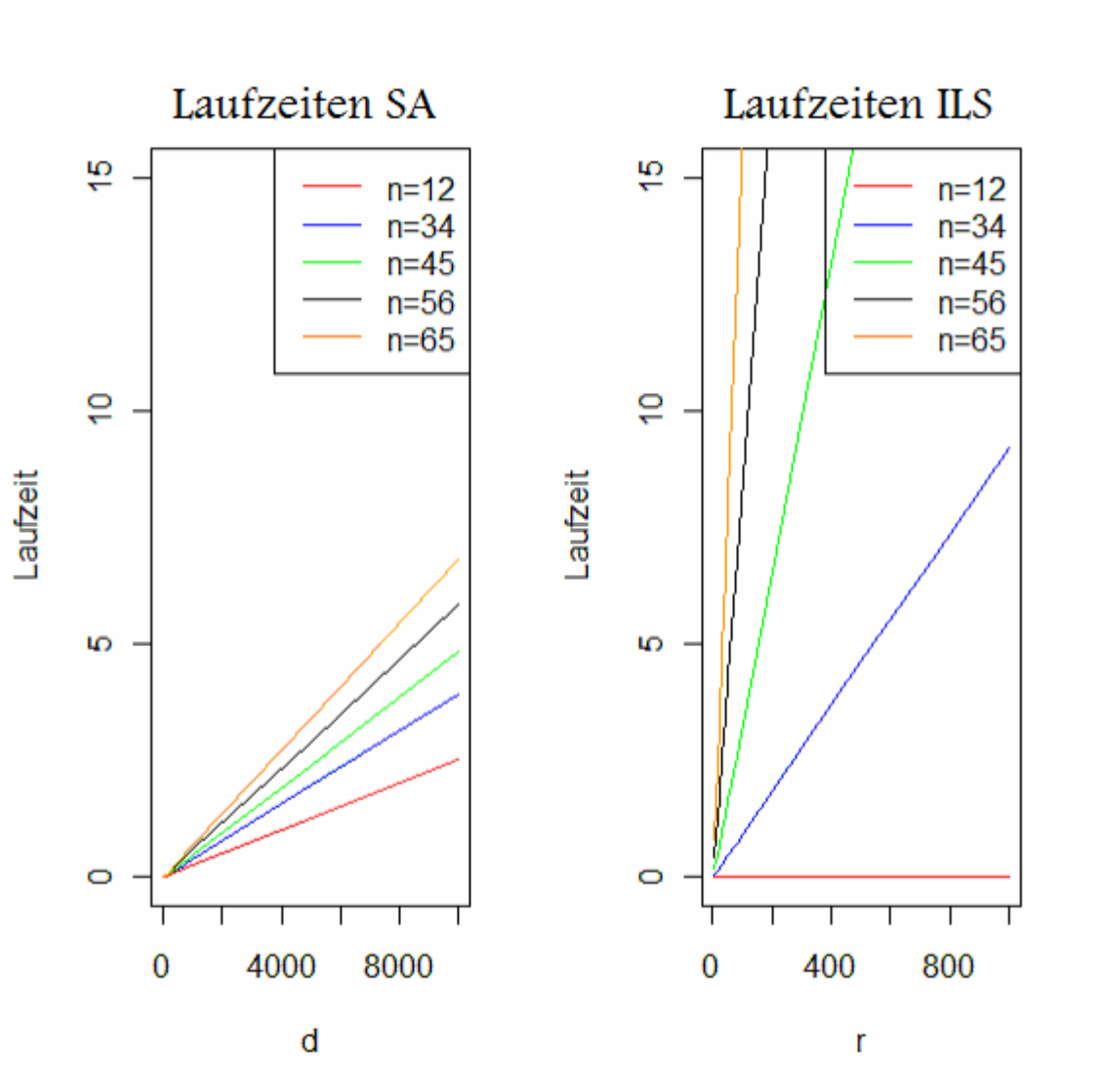

Abbildung 43: Laufzeitenwachstum ILS_r vs. $SA_{0,5}$

5.2. Absoluter Performancevergleich

Um den tatsächlichen, absoluten Erfolg der Heuristiken quantifizieren zu können, müsste man die Optima für die simulierten TSP-Instanzen kennen. Für n<13 ist durch vollständige Enumeration die Ermittlung des Optimums in akzeptabler Zeit möglich. Die Leistung der Heuristiken kann also für TSP(n) Instanzen mit n<13 leicht überprüft und durch $\frac{x_{heu}}{x_{opt}} - 1$ also als Anteil des enumerativ ermittelten Optimums, den der heuristische Lösungswert über eben diesem Optimum liegt. Dieser Anteil wird im Folgenden als Fehler bezeichnet.

Nun macht es keinen Sinn Instanzen mit n>12 wie bisher zufällig zu generieren, weil das Optimum nicht bekannt ist und auch in akzeptabler Zeit nicht ermittelt werden kann. Wir bedienen uns im Folgenden wieder der TSP-LIB Instanzen.

Die erste Auffälligkeit, die jene Instanzen betrifft ist, dass es sich im Gegensatz zu den in dieser Arbeit simulierten dabei um ganzzahlige Distanzmatrizen handelt. Wollen wir also zunächst visualisieren, welchen Einfluss der Größenbereich der Distanzen auf die heuristischen Fehler hat. Dazu wurden 3 Probleminstanzen zufällig generiert:

$$n12_{klein}=(u_{i,j})_{12x12}$$
$$n12_{mittel}=(v_{i,j})_{12x12}$$
$$n12_{groß}=(w_{i,j})_{12x12}$$

wobei $u_{i,j}$, $v_{i,j}$, $w_{i,j}$ (i,j=1,…,12; i≠j) Realisierungen von aufgerundeten Unif([0;10]), Unif([0;100]) bzw. Unif([0,1000]) Zufallsvariablen sind. Sowohl bei der ILS_r Heuristik, als auch beim SA scheint der Größenbereich auf den zum Optimum relativen Fehler keinen Einfluss zu haben:

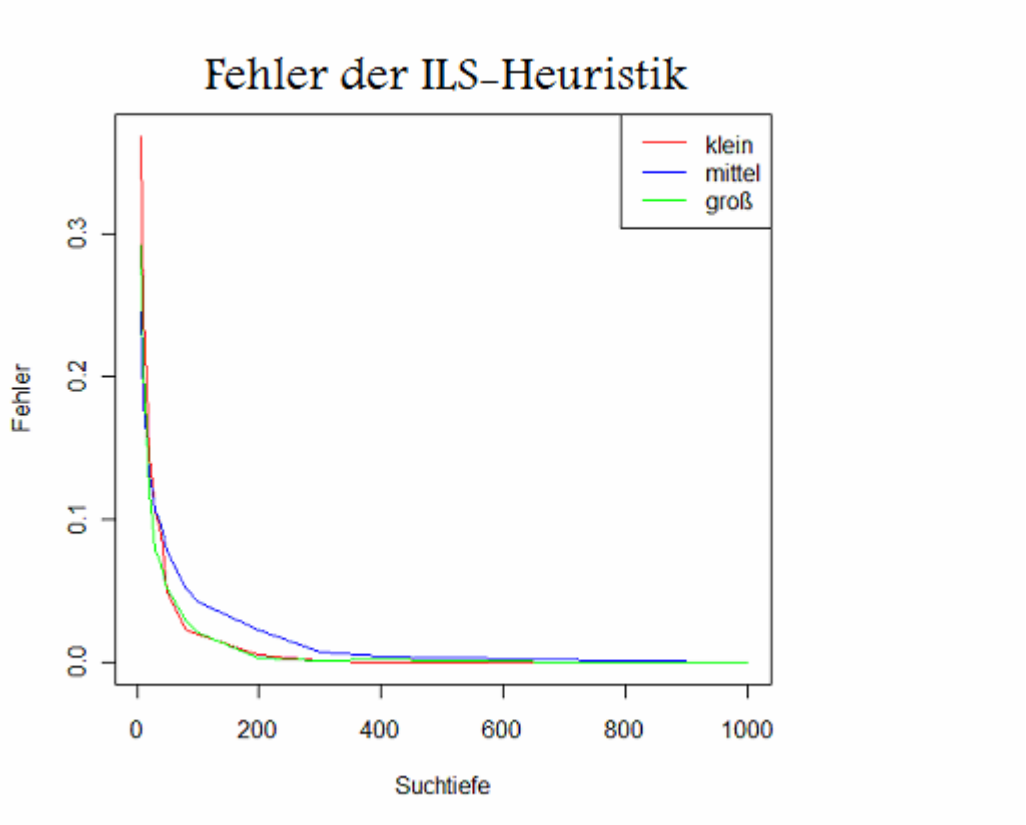

Abbildung 44: Fehler der ILS - Heuristik

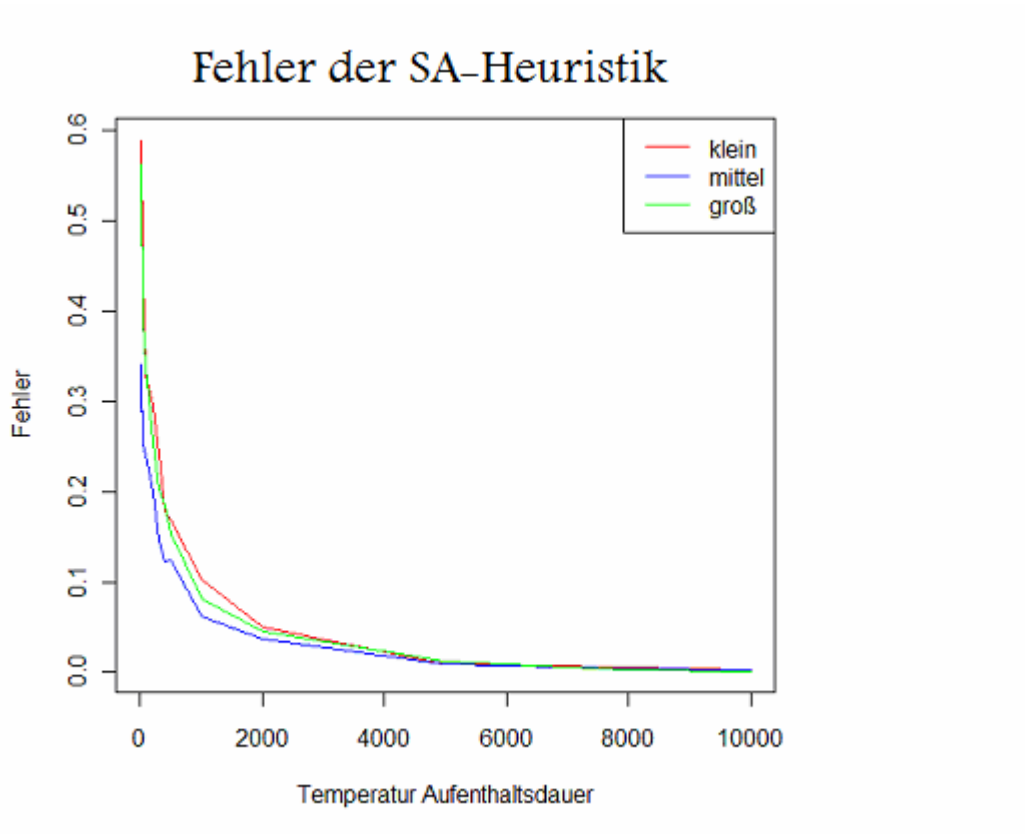

Abbildung 45: Fehler des SA

Im nächsten Schritt betrachten wir wie der Fehler von den Parametern der Heuristiken (also r bei ILS und d bei SA) abhängt. Dabei werden nun die exakt gelösten größeren Instanzen sowie die zufällig generierte n=12-Instanz aus dem Wertebereich 1,…10 verwendet. Die tabellierten Werte sind die arithmetischen Mittel der Fehler bei jeweils (für n=12,34,45,56,65) N=100 maligem Lösen von ein und der selben Probleminstanz:

ILS:

	r=5	r=10	r=20	r=30	r=40	r=50	r=80	r=100	r=200	r=300	r=400	r=500	r=800	r=1000
n=12	0,37	0,25	0,14	0,10	0,09	0,05	0,02	0,02	0,01	0,00	0,00	0,00	0,00	0,00
n=34	0,37	0,32	0,31	0,28	0,25	0,24	0,22	0,23	0,18	0,17	0,17	0,15	0,14	0,14
n=45	0,63	0,57	0,52	0,51	0,49	0,49	0,45	0,45	0,41	0,41	0,39	0,37	0,36	0,35
n=56	0,89	0,87	0,80	0,77	0,75	0,73	0,71	0,69	0,67	0,65	0,62	0,62	0,61	0,60
n=65	1,04	0,97	0,94	0,92	0,89	0,88	0,86	0,84	0,80	0,80	0,77	0,77	0,75	0,74

$SA_{0,5}$:

	d=10	d=20	d=30	d=40	d=50	d=100	d=500	d=1000	d=5000	d=10000
n=12	0,59	0,56	0,48	0,48	0,44	0,33	0,17	0,10	0,01	0,00
n=34	0,93	0,70	0,61	0,55	0,52	0,41	0,29	0,24	0,16	0,14
n=45	1,22	0,97	0,84	0,76	0,71	0,58	0,38	0,33	0,24	0,22
n=56	1,76	1,39	1,24	1,15	1,06	0,85	0,56	0,47	0,35	0,31
n=65	2,00	1,62	1,43	1,29	1,23	0,98	0,67	0,58	0,43	0,39

Tabelle 56: Fehler bei großen Instanzen

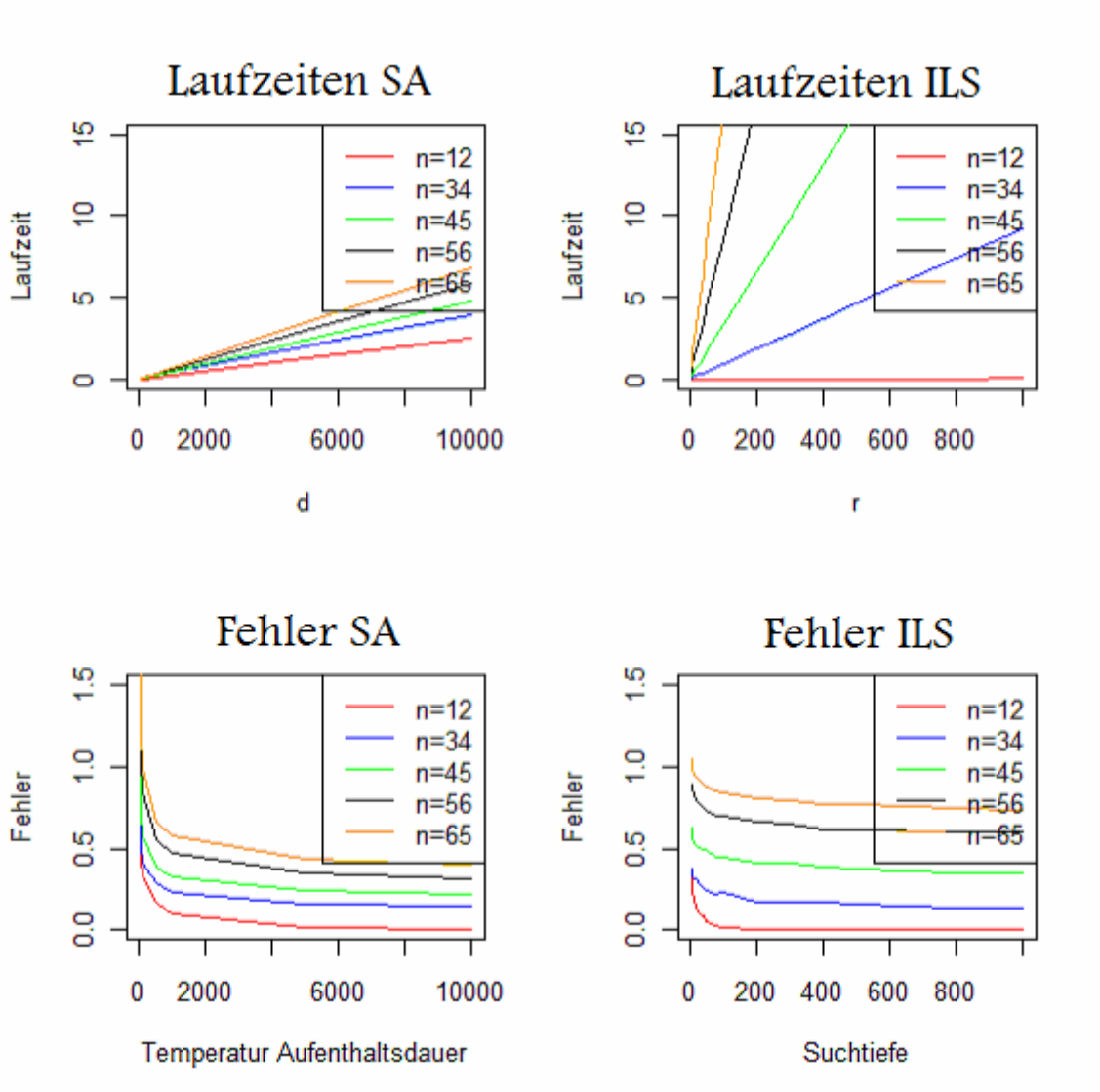

Abbildung 46: Fehler und Laufzeiten beider Heuristiken

Im Kontrast zur Vermutung aus Kapitel 5.1, in dem die beiden Heuristiken pro Stichprobe jeweils auf 100 zufällig generierte Instanzen angewandt wurden, schneidet nun $SA_{0,5}$ wie in Abbildung 46 zu sehen ist sowohl im Laufzeitvergleich als auch im Performancebereich offensichtlich besser ab, als ILS_r. Es darf dabei nicht außer Acht gelassen werden, dass es hierbei um asymmetrische TSP Instanzen (Abschnitt 5.1.2.) geht, die Aussage aus Abschnitt 5.1.1. aber für symmetrische TSP getroffen wurde und dass der Wertebereich der Distanzen nicht [0;1] ist, sondern den bereits exakt gelösten Beispiel-Distanzmatrizen entspricht. Dieser Wertebereich hat natürlich direkten Einfluss auf die Funktionswertdifferenzen (Abbildung 47) und somit auch auf die Trotzwahrscheinlichkeiten.

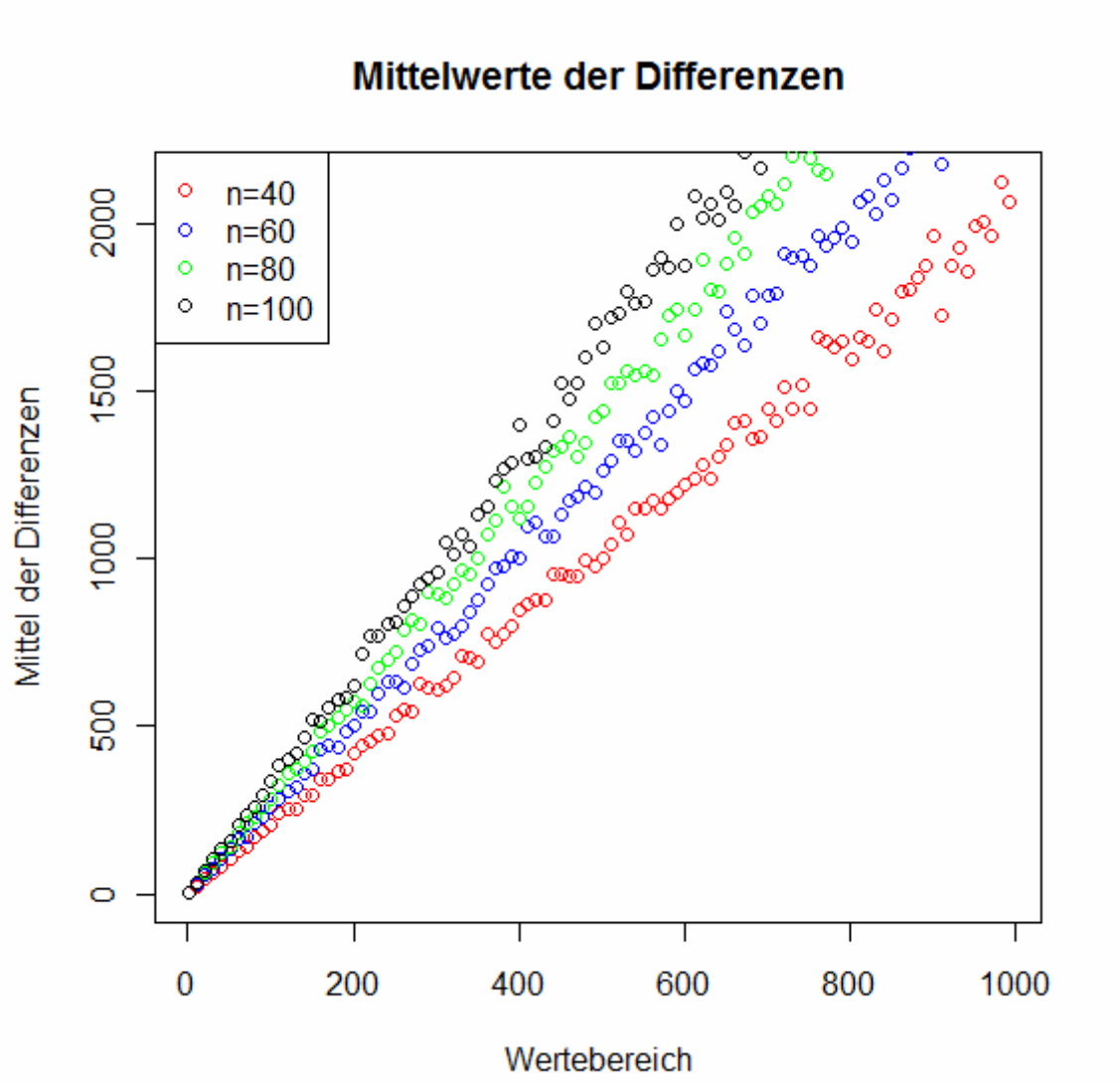

Abbildung 47: Simulierte Funktionswertedifferenzen-Mittelwerte

Der Einfluss des Wertebereichs auf die Trotzwahrscheinlichkeiten ist mathematisch unumstritten, ob sich dieser Einfluss nun tatsächlich auch auf die Performance des SA auswirkt wird wieder empirisch untersucht. Tabellen 57 (asymmetrisch) und 58 (symmetrisch) verdeutlichen diese Unterschiede (r=5, d=1000, t=1):

Bereich:	ILS		Annealing		
0-10	Zeit	Wert	Zeit	Wert	Quotient
n=40	0,04435	60,82	0,03276	22,62	2,688771
n=60	0,22376	94,07	0,04392	34,57	2,7211455
n=80	0,67693	129,03	0,0564	45,31	2,84771574

Bereich:	ILS		Annealing		
0-100	Zeit	Wert	Zeit	Wert	Quotient
n=40	0,04662	765	0,03479	400,51	1,91006467
n=60	0,12478	999,96	0,04534	602,6	1,65940923
n=80	0,78483	1607,04	0,05865	792,74	2,02719681

Bereich:	ILS		Annealing		
0-1000	Zeit	Wert	Zeit	Wert	Quotient
n=40	0,05038	7727,87	0,03261	4146,95	1,86350691
n=60	0,23334	12070,02	0,04403	6127,71	1,969744
n=80	0,72979	16469,37	0,05738	8223,35	2,00275678

Tabelle 57: Einfluss des Wertebereichs auf die Performance (asymmetrisch)

Bereich:	ILS		Annealing		
0-10	Zeit	Wert	Zeit	Wert	Quotient
n=40	0,08929	8,89	0,03233	23,05	0,3856833
n=60	0,42029	6,87	0,04398	34,18	0,20099473
n=80	1,30185	5,2	0,0565	45,61	0,11401009

Bereich:	ILS		Annealing		
0-100	Zeit	Wert	Zeit	Wert	Quotient
n=40	0,11149	226,63	0,03572	407,98	0,55549292
n=60	0,57973	249,05	0,04553	591,55	0,42101259
n=80	1,91296	267,58	0,05697	787,24	0,33989635

Bereich:	ILS		Annealing		
0-1000	Zeit	Wert	Zeit	Wert	Quotient
n=40	0,11238	2387,62	0,03226	4234,07	0,5639066
n=60	0,60076	2674,97	0,04401	6115,91	0,4373789
n=80	1,9927	2980,56	0,05713	8297,15	0,35922696

Tabelle 58: Einfluss des Wertebereichs auf die Performance (symmetrisch)

Offensichtlich schneidet SA bei asymmetrischen TSP besser ab, als ILS und umgekehrt. Der Wertebereich scheint keinen echten Einfluss auf die Performanceunterschiede zu haben. Er vergrößert zwar die Funktionswertdifferenzen (Abbildung 47) und verändert somit auch die Trotzwahrscheinlichkeiten, für die Arbeit der Heuristik scheint dies allerdings nicht von besonderer Relevanz. Die monoton fallende Temperaturfolge gleicht diesen Effekt offenbar wieder aus.

5.3. Effizienzoptimierung

5.3.1. Der 2-opt-Move

Wie in Abschnitt 5.1.2 zu sehen, ist bereits $SA_{0,5}$ der ILS_r Heuristik bei asymmetrischen TSP-Instanzen nicht nur Laufzeit- sondern auch Lösungsqualitätsmäßig überlegen. Dabei wurde der Decreasefaktor f mit bisher 0,5 noch sehr niedrig gewählt.
Der Laufzeitvorteil für SA bei symmetrischen TSP(n) liegt am so genannten 2-opt-Move[1] [Croes 1958], der wie folgt funktioniert:
Eine mögliche Lösung des TSP(n), x, ist eine Permutation der Zahlen 1, 2, …, n, also:

$$x = (a_1,\ldots, a_{i-1}, a_i, a_{i+1},\ldots, a_{j-1}, a_j, a_{j+1},\ldots,a_n)$$

Der 2-opt-Move, der den i-ten Ort mit dem j-ten Ort vertauscht, kann als Funktion $m_{i,j}: T \rightarrow T$ aufgefasst werden, sodass:

$$m_{i,j}(x) = (a_1,\ldots, a_{i-1}, a_j, a_{j-1},\ldots, a_{i+1}, a_i, a_{j+1},\ldots, a_n)$$

Im Gegensatz zur herkömmlichen Nachbarfunktion $n_{i,j} : T \rightarrow T$

$$n_{i,j}(x) = (a_1,\ldots, a_{i-1}, a_j, a_{i+1},\ldots, a_{j-1}, a_i, a_{j+1},\ldots, a_n)$$

müssen wegen $d_{i,j} = d_{j,i}$ (aufgrund der Symmetrie) durch den 2-opt-Move (Abbildung 48) halb so viele Rechenoperationen durchgeführt werden, da:

$$L(m_{i,j}(x)) = L(x) - d_{i-1,i} - d_{j,j+1} + d_{i-1,j} + d_{i,j+1}$$

und

$$L(n_{i,j}(x)) = L(x) - d_{i-1,i} - d_{i,i+1} - d_{j-1,j} - d_{j,j+1} + d_{i-1,j} + d_{j,i+1} + d_{j-1,j} + d_{i,j+1}$$

[1] Implementierung siehe 7.3.3.

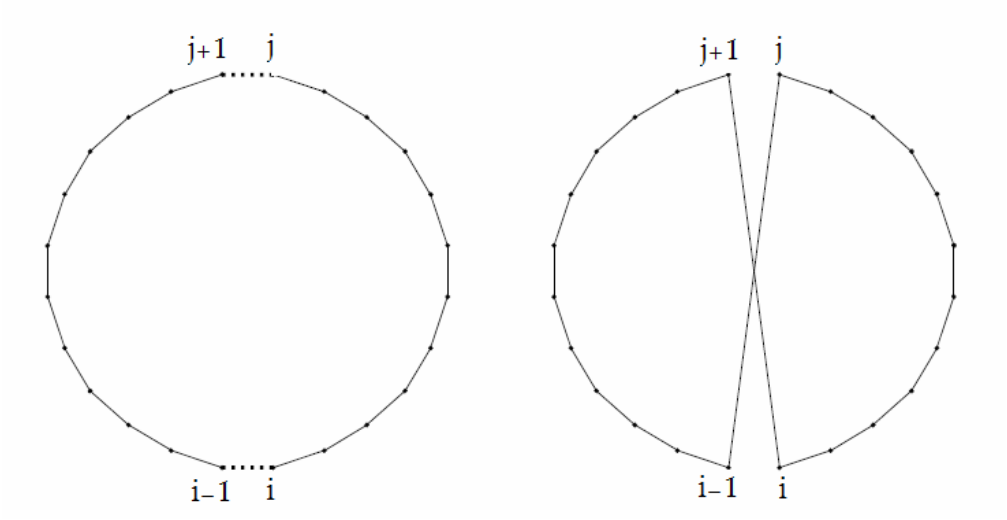

Abbildung 48: Der 2-opt-Move

5.3.2. Erhöhung des Decreasefaktors

Nun kann weiters der Decreasefaktor f von bisher 0,5 auf beispielsweise 0,999 erhöht werden. Dieser Faktor hat auf die Laufzeit von SA eine ähnliche Auswirkung wie die in Kapitel 4 betrachtete Temperaturaufenthaltsdauer. Er bestimmt das monotone Fallverhalten der Temperaturfolge, da $t_{n+1} = t_n * f$ gilt.
Eine effizientere Parametrisierung (also die Wahl eines „besseren“ t, d und f) der SA Heuristik kann aus diesem Laufzeitpolster, den SA nach Abschnitten 5.1.1. und 5.1.2. auf ILS_r hat, nun auch für symmetrische (im Falle der Asymmetrie war SA ohnehin effizienter) für eine Verbesserung der ermittelten Lösungswerte opfern. Dies wird vor allem für größere TSP (n>200) von besonderer Bedeutung sein, da die Laufzeit in Abhängigkeit von n bei der ILS_r wesentlich schneller wächst als bei SA. Die folgende Tabelle zeigt Lösungswerte und Laufzeiten (Mittelwerte, N=10) von ILS_5 und SA (t=100,d=100,f=0.999)

	ILS		SA	
	Zeit	Wert	Zeit	Wert
n=50	0,119	9,754	0,100	4,537
n=80	0,749	16,351	0,959	6,709
n=100	1,524	20,939	0,959	8,559
n=150	10,895	31,275	0,972	13,183
n=200	26,001	42,711	0,983	18,280

Tabelle 59: ILS vs. SA: Laufzeit und Werte (symmetrisch)

6. ANHANG

6.1. Zusammenfassung

Das Traveling Salesman Problem (TSP) ist aufgrund seiner Struktur ein *kombinatorisches Optimierungsproblem* und stammt aus dem Bereich des Operations Research. n Orte deren paarweise Distanzen zueinander bekannt sind, sind hintereinander so zu besuchen, dass die Länge des dabei zurückgelegten Gesamtwegs (Tour) möglichst kurz ist. Die kombinatorische Beschaffenheit garantiert einen endlichen Lösungsraum, somit kann die optimale Lösung durch Durchprobieren aller möglichen Touren ermittelt werden (vollständige Enumeration). In der Praxis ist dies jedoch aufgrund des schnellen Wachstums der Anzahl der möglichen Touren in n für große n nicht zu bewerkstelligen. Es ist kein exakter, polynomialer Lösungsalgorithmus für das TSP bekannt, daher bedient man sich diverser Heuristiken, die gute, jedoch nicht zwangsläufig optimale Lösungen liefern.
In dieser Arbeit wurde das Laufzeitverhalten von zwei ausgewählten Heuristiken (Iterated Local Search [ILS] und Simulated Annealing[SA]) in zwei Varianten analysiert. Hauptaugenmerk wurde dabei auf die Prognose der Laufzeit sowie deren statistische Verteilung gelegt.
Kapitel 1 stellt eine kleine Einführung in die Thematik dar und bietet Deklarationen für fortlaufend verwendete Buchstaben und Symbole.
Kapitel 2 beschreibt die verwendeten Ressourcen und erklärt die geschriebenen Prozeduren. Außerdem wird der Einfluss der beiden TSP-Charakteristika „Metrik“ und „Symmetrie“ auf die Laufzeiten und die ermittelten Lösungswerte beider Heuristiken untersucht um in den Laufzeit-Kapiteln 3 und 4 sowie im Funktionswerte-relevanten Kapitel 5 auf die wichtigen Faktoren einzugehen.
Während der Faktor „Metrik“ auf die Laufzeiten beider Heuristiken keinen Einfluss hat, wirkt er sich auf die Höhe der Funktionswerte klarerweise aus, da unter der verwendeten Instanz-Konstruktionsvorschrift (Abschnitt 2.1.3.) die Dreiecksungleichung die Distanz |AB| zwischen zwei Orten A und B zu einer unteren Schranke für die Summe der Distanzen |AC| und |CD| für jeden Ort C macht. Diese Beschränkung wird im nichtmetrischen Fall nicht erzwungen, was kürzere Tourlängen ermöglicht.
Der Faktor „Symmetrie“ wirkt sich bei SA weder auf die Laufzeit noch auf die Höhe der ermittelten Lösungswerte aus, bei ILS_r hingegen sowohl als auch.

In Kapitel 3 wurde dann das Laufzeitverhalten von ILS (beide Varianten) untersucht. Diese bricht entweder nach r Misserfolgen ab (Variante A) oder sobald der approximative Lösungswert in einem bestimmten Toleranzbereich um den tatsächlichen Lösungswert liegt (Variante B). Für Variante A wurde ein Modell aufgestellt, das die Laufzeit und deren Variabilität in Abhängigkeit von n und r prognostiziert, Variante B bot keinerlei Modellierungsbasis. Hypothesen für Anpassungstests auf Gammaverteilung (Variante A) und Exponentialverteilung (Variante B) konnten nicht verworfen werden. Schließlich wurde noch die Erfolgswahrscheinlichkeit der Heuristik (Variante A) in Abhängigkeit von n und r sowohl empirisch als auch wahrscheinlichkeitstheoretisch modelliert. In letzterem Ansatz mit Hilfe der Tatsache, dass die Anzahl der Versuche der ILS_r Heuristik unter Unabhängigkeitsannahme (die aufgrund der Zufälligen Startlösungen erfüllt ist) negativ-binomialverteilt mit einer Erfolgswahrscheinlichkeit, die der Misserfolgswahrscheinlichkeit einer einzigen lokalen Suche entspricht.
Kapitel 4 stellt den zu Kapitel 3 bezüglich SA analogen Abschnitt dar. Der Einfluss von n, t und d auf die Laufzeit wurde untersucht und prognostisch verarbeitet, obwohl die Laufzeit von SA (Variante A) *beinahe* deterministisch ist, da als einzige stochastische Komponente über das Auffinden einer Verbesserungsrichtung bzw. die Trotzwahrscheinlichkeiten einfließt. „Metrik“ und „Symmetrie“ waren hier wegen 2.5 und 2.6 nicht von Relevanz. Es stellte sich zudem heraus, dass die Laufzeiten von SA (nur Variante B, Variante A wurde bezüglich der Verteilung wegen beinaher Nichtzufälligkeit nicht untersucht) ebenfalls einer Exponentialverteilung folgen. Der Fit scheint allerdings im Gegensatz zu ILS erst bei größerem n wirklich brauchbar.
Kapitel 5 widmet sich der Leistungskraft beider Heuristiken und versucht vergleichende und absolute Aussagen darüber zu treffen. Dabei spielte „Metrik“ als auch „Symmetrie“ eine Rolle, da beide Faktoren nach Kapitel 2 Einfluss auf die Höhe der heuristisch ermittelten Lösungswerte haben. Tatsächlich stellte sich lediglich der Faktor „Symmetrie“ als problematisch heraus, da er, im Gegensatz zum Faktor „Metrik“, der die Werte aufgrund der Dreiecksungleichung konstant (also unabhängig von der Wahl der Heuristik) beeinflusst, signifikant kleiner Lösungswerte bei ILS_r als bei SA produzierte.
Für den absoluten Vergleich wurden bereits gelöste, bekannte TSP-Instanzen herangezogen, da mittels vollständiger Enumeration TSP nur für $n<14$ exakt in sinnvoller Zeit gelöst werden können.

Die wesentliche Schluss, der sich dabei herauskristallisierte war, dass ILS_r bei symmetrischen Instanzen effizienter war als SA und umgekehrt.
Für größere Instanzen kann allerdings aufgrund der Laufzeitexplosion von ILS_r im Gegensatz zu SA dieser Laufzeitunterschied (nicht zuletzt auch durch den 2-opt-Move) in eine derartige Performancesteigerung umgemünzt werden, dass auch im Symmetriefall SA der ILS_r überlegen ist (Abschnitt 5.3.2.)

6.2. Abbildungsverzeichnis

6.3. Tabellenverzeichnis

6.4. Ergänzender Programmcode

6.4.1. Grid-Search-Verfahren (R)

```
grid_search2<-function(x,y,prec,from=-4,to=-0.001)
{
   i<-1
   t<-seq(from=from,to=to,by=0.01)
   d<-1
   k<-10
   c<-vector()
   down<-TRUE
      while(down)
      {
         if(length(y[y^(1/t[i])!=Inf])>5)
         {
           lm1<-lm(y[y^(1/t[i])!=Inf]^(1/t[i])~x[y^(1/t[i])!=Inf])
          lm2<-lm(y[y^(1/t[i+1])!=Inf]^(1/t[i+1])~x[y^(1/t[i+1])!=Inf])
           c[i]<-summary(lm1)$r.squared
           c[i+1]<-summary(lm2)$r.squared
           down<-c[i]<c[i+1]
           if(down){i<-i+1}
         }
         else{i<-i+1}
      }
      c<-vector()
      up<-t[i+1]
      lo<-t[i-1]
      n<-1
      while((up-lo)>prec && lo!=up)
      {
         z<-seq(from=lo,to=up,length.out=k)
         n<-1
         for(j in z)
         {
            model<-lm(y[y^(1/j)!=Inf]^(1/j)~x[y^(1/j)!=Inf])
             c[n]<-summary(model)$r.squared
            n<-n+1
         }
         position<-(1:length(z))[max(c)==c]
         minimierer<-z[max(c)==c]
         minimum<-max(c)
         lo<-z[position-1]
         up<-z[position+1]
      }
      return(lo/2+up/2)
}
```

6.4.2. Simulation der auftretenden Funktionswertdifferenzen

Der Funktion *sample_dif* generiert Zufallsstichproben von TSP-Tourenlängendifferenzen. Diese sind bei der Berechnung der Trotzwahrscheinlichkeiten bei SA von Interesse. Ihr werden 5 Parameter übergeben:

- *n* ist die TSP-Dimension
- *k* ist die Anzahl der zu simulierenden Tourlängen
- *r* ist die Anzahl der aus der k-elementigen Menge zufällig ausgewählten Differenzen
- *min* und *max* definieren den Wertebereich der TSP-Distanzen

```
sample_dif<-function(n,k,r,min,max)
{
      A<-matrix(runif(n*k),nro=n,ncol=k)
      space<-apply(A,2,sum)
      select1<-floor(runif(r,min=1,max=k+1))
      select2<-floor(runif(r,min=1,max=k+1))
      diffs<-space[select1]-space[select2]
      return(diffs)
}
```

6.4.3. Der TWO-opt-Move

```
void tausch(int feld[],int dimension)
{
     int i,j,safe;

     i=(rand()%(dimension-1))+1;
     j=(rand()%(dimension-1))+1;

     while(i==j)
     {
               j=(rand()%(dimension-1))+1;
     }

     if(i>j)
     {
            safe=i;
            i=j;
            j=safe;
            safe=0;
     }

     while(i<j)
     {
               safe=feld[i];
               feld[i]=feld[j];
               feld[j]=safe;
               i++;
               j--;
     }
}
```

6.5. Quellen

6.5.1. Literatur

- H. R. Lourenco, O. C. Martin, T. Stützle:
  "Iterated Local Search" (2001)

- D. S. Johnson, L. A. McGeoch:
  "The Traveling Salesman Problem: A Case Study in Local Optimization" (1995)

- J. Ardelius, E. Aurell:
  "Bahavior of heuristics on large and hard satisfiability problems" (2006)

- D. L. Applegate, R. E. Bixby, V. Chvátal, W. J. Cook:
  "The Traveling Salesman Problem. A Computational Study" (2007)
  ISBN 978-0-691-12993-8

- E.L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, D. B. Shmoys:
  „The Traveling Salesman Problem. A Guided Tour of Combinatorial Optimization" (1985)
  ISBN 0-471-90413-9

- E. Aarts, J. K. Lenstra:
  "Local Search in Combinatorial Optimization" (1997)
  ISBN13: 978-0-691-11522-1

- M. R. Garey, D. S. Johnson:
  "Computers and Intractability: A Guide to the Theory of NP-Completeness" (1990)
  ISBN: 0716710455

- G. Gutin, A. P. Punnen
  „The Traveling Salesman Problem and Its Variations" (2006)
  ISBN: 0-387-44459-9

- W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, A. Schrijver
  "*Combinatorial Optimization"(1997)*
  ISBN 0-471-55894-X

- A. Ruigies
  "Simulated Annealing und verwandte Verfahren für das Traveling Salesman Problem" (1995)
  ISBN 978-3838606163

- P. Salamon, P. Sibani, R. Frost
  "Facts, Conjectures, and Improvements for Simulated Annealing" (1987)
  ISBN 978-0898715088

6.5.2. Weitere Informationsquellen

- The Travellings Salesman Problem Home
  http://www.tsp.gatech.edu/

- TSP LIB
  http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/

6.6. Curriculum Vitae

Persönliches:

Name:	Reinhard Bazant
Geboren:	6.11.1984 in Wien
Staatsangehörigkeit:	Österreich

Ausbildung:

- Juni 2003: Matura GRG De La Salle Schule Strebersdorf (1210 Wien)
- 2003 – 2004: Präsenzdienst
- 2004 – 2010: Statistik-Studium an der Universität Wien
- 2009: Abschluss des Bakkalaureatsstudiums Statistik

Sonstiges:

- 2009: Anstellung im Fachbereich „Mathematik und Statistik“ der Fachhochschule Wr. Neustadt
- Sprachkenntnisse: Deutsch, Englisch (fließend)
  Französisch (Grundkenntnisse)