Experten und Gutachter für DOTT

von Stephan Scholz

Projektarbeit im Fachbereich Informatik
AG Avenhaus
Universität Kaiserslautern

Betreuer : Dr. rer. nat. Jörg Denzinger

September 1996

Inhaltsverzeichnis

1. **Einleitung** 1
2. **Das Traveling Salesman Problem** 2
   2.1. **Problemstellung** 2
   2.2. **Tourkonstruktionsverfahren** 3
      2.2.1. Nearest Neighbour 3
      2.2.2. Insertion-Verfahren 3
   2.3. **Tourverbesserungsverfahren** 4
      2.3.1. 2-Opt 5
      2.3.2. 3-Opt 5
      2.3.3. Lin-Kernighan 5
   2.4. **Genetische Algorithmen** 6
      2.4.1. Edge Recombination Crossover 8
      2.4.2. 4-Opt-Mutation 8
      2.4.3. Fitneß-Funktion 9
      2.4.4. Kombination mit anderen Verfahren 10
3. **Die Teamwork-Methode** 11
   3.1. **Prinzip von Teamwork** 11
      3.1.1. Experten 13
      3.1.2. Gutachter 13
      3.1.3. Der Leiter 14
   3.2. **Anwendung von Teamwork auf das TSP** 15
      3.2.1. Experten 15
      3.2.2. Gutachter 15
      3.2.3. Der Leiter 16
4. **Das System DOTT** 17
   4.1. **Initiale Version** 17
      4.1.1. Datenstruktur 17
      4.1.2. Experten 19
      4.1.3. Gutachter 21
      4.1.4. Leiter 22
   4.2. **Neue Experten** 25
      4.2.1. OPT2 25
      4.2.2. MUT4_LO3 25
      4.2.3. MUT4_LO2 27

4.2.4. ER_LO3 28
4.2.5. ER_LO2 28
4.3. **Neue Gutachter** 29
4.3.1. REF_LENGTH 29
4.3.2. REF_DIVERS 31
4.4. Übersicht der Komponenten 34
5. **Experimente und Testläufe** 36
5.1. **Sequentielle Leistung** 36
5.1.1. MUT4_LO2 37
5.1.2. MUT4_LO3 38
5.1.3. ER_LO2 39
5.1.4. ER_LO3 41
5.2. **Teamleistung** 44
5.2.1. MUT4_LO3 und ER_LO3 46
5.2.2. MUT4_LO2 und ER_LO2 48
6. **Zusammenfassung und Ausblick** 52
Anhang A: Literatur 54
Anhang B: Verwendete Beispiele 55

1. Einleitung

Das Problem des Handelsreisenden oder Traveling Salesman Problem (TSP) beschäftigt sowohl Mathematiker, Wirtschaftswissenschaftler als auch Informatiker schon seit einigen Jahrzehnten. Der Reiz dieser Aufgabe besteht aus seiner einerseits sehr einfachen Formulierung jedoch andererseits extrem schweren Lösung. Das TSP besteht darin, die kürzeste Rundreise zwischen beliebig vielen Städten zu finden, wobei jede Stadt genau einmal besucht wird. Es gehört zu den sog. NP-vollständigen Problemen; Lösungsalgorithmen müssen daher mit großer Wahrscheinlichkeit exponentielle Komplexität haben.

Seit geraumer Zeit werden zur Lösung des TSP Computer eingesetzt. Aufgrund der hohen Komplexität des TSPs braucht man leistungsfähige Lösungsmethoden, man benutzt bei großer Städteanzahl daher parallele und verteilte Rechnerarchitekturen. Ein effizientes Verfahren zur verteilten Lösung von Problemen stellt die *Teamwork*-Methode dar. Sie wurde bereits erfolgreich auf die Aufgabe des automatischen Theorembeweisens angewendet [De93]. Die Arbeitsweise von Teamwork beruht auf dem Konzept von mehreren unabhängigen *Experten*, die sich mit dem eigentlichen Lösen der Aufgabe beschäftigen. Deren Arbeit wird von *Gutachtern* ausgewertet, die einerseits die Experten bewerten und andererseits deren besten Ergebnisse auswählen, um sie an den *Leiter* weiterzureichen. Dieser stellt aufgrund der Bewertungen durch die Gutachter ein neues Team und eine neue Aufgabenstellung für die Experten zusammen, er hat also die Kontrolle über den Ablauf. Die Experten stehen also einerseits in *Konkurrenz* zueinander, andererseits ist aber *Kooperation* dadurch gegeben, daß Ergebnisse von allen Experten verwendet werden.

Das in dieser Projektarbeit vorgestellte System DOTT (**D**istributed **O**ptimization of the **T**raveling Salesman Problem using **T**eamwork) ist eine Anwendung dieses Verfahrens auf das TSP. In der ersten Version von DOTT [Leo95] sind Gutachter praktisch noch nicht implementiert. Im Rahmen dieser Projektarbeit sind nun leistungsfähige Gutachter entwickelt worden, durch die eine weitere Leistungssteigerung des Systems erreicht werden konnte. Die Entwicklung eines automatischen, dynamischen Leiters ist jedoch noch späteren Arbeiten vorbehalten. Die seit der ursprünglichen Version von DOTT hinzugekommenen neuen Experten basieren durchweg auf dem Konzept der genetischen Algorithmen. Die Idee dieser Verfahren, eine Population durch Mutation und Selektion auf einen immer besseren Stand zu bringen, wurde angepaßt auf das TSP durch Kombination mit Tourverbesserungsverfahren. Dies ergab eine weitere Leistungssteigerung gegenüber den "reinen" genetischen Algorithmen.

Diese Arbeit teilt sich wie folgt ein:

In Kapitel 2 wird das TSP in allgemeiner Form beschrieben, wobei auf seine Problemstellung und existierende Lösungsalgorithmen kurz eingegangen wird. Die Heuristiken teilen sich auf in Tourgenerierungs- und Tourverbesserungsverfahren. Besonders betrachtet wird dabei die Gruppe der genetischen Algorithmen. Kapitel 3 stellt die Teamwork-Methode ausführlich vor. Neben dem allgemeinen Prinzip wird die Anwendung auf das TSP erklärt. Besonderes Gewicht bekommt die Rolle der Gutachter. Im 4. Kapitel wird das System DOTT behandelt. Die bereits implementierten Komponenten und insbesondere die neu hinzugekommenen Gutachter sowie die genetischen Experten werden beschrieben. Kapitel 5 stellt die durchgeführten Testläufe vor, die sich unterteilen in sequentielle (Einzelleistung eines Experten) und verteilte Läufe. Hierbei wird zweierlei gezeigt; zum einen, daß man mit der Teamwork-Methode tatsächlich bessere, also kürzere Touren als im sequentiellen Fall bekommt. Zum anderen daß die Ergebnisse der sequentiellen Läufe bei den Teamläufen früher gefunden werden. Es wird also die sequentielle- mit der Teamleistung verglichen. Schließlich gibt Kapitel 6 eine Zusammenfassung der Ergebnisse und einen Ausblick auf mögliche Erweiterungen des Systems.

2. Das Traveling Salesman Problem

In diesem Kapitel wird ein Überblick über das symmetrische TSP gegeben (vgl. [Re94], [LLRS85]). Neben der eigentlichen Problemstellung wird auf die unterschiedlichen Näherungsverfahren zur Lösung des TSP eingegangen, die sich unterteilen in Tourkonstruktions- und Tourverbesserungsverfahren. Besonderer Schwerpunkt wird dabei auf den genetischen Algorithmen liegen.
Verwandte Probleme sind das asymmetrische TSP und das Rural Postman Problem, mit denen wir uns hier aber nicht beschäftigen wollen. Wenn daher in dieser Arbeit vom TSP gesprochen wird, so ist damit implizit das symmetrische TSP gemeint.

2.1. Problemstellung

Das Traveling Salesman Problem befaßt sich mit der Aufgabe, die kürzeste Rundreise durch eine Menge von Städten zu finden, wobei jede Stadt genau einmal besucht wird. Im folgenden wird eine mathematische Definition mit Hilfe der Graphentheorie gegeben.

Def.2.1.1.: vollständiger Graph

Ein Graph $G=(V,E)$ heißt **vollständig** gdw für alle $u,v \in V$ die Kante von u nach v in E enthalten ist. Wir bezeichnen einen vollständigen Graphen der n Knoten enthält mit $K_n=(V_n,E_n)$ wobei $V_n=\{v_1,...,v_n\}, E_n=\{e_1,...,e_n\}$.
Er heißt **symmetrisch** gdw $c_{ij}=c_{ji}$ f.a. $i,j \in \{1,...,n\}$, wobei c_{kl} die Kantengewichtung der Kante von v_k nach v_l ist.
Oft stellt man die Gewichtungen durch eine Matrix dar. Sie wird Kostenmatrix oder **Entfernungsmatrix** genannt: $C=\begin{pmatrix} c_{11}, & \dots & c_{1n} \\ \vdots & \ddots & \vdots \\ c_{n1} & \dots & c_{nn} \end{pmatrix}$.

Def.2.2.2.: Weg

Ein **Weg** ist eine endliche, nichtleere Folge aufeinanderfolgender Kanten.

Def.2.2.3.: Tour

Eine **Tour** durch einen Graphen ist ein Weg der an dem gleichen Knoten beginnt und endet. Eine Tour läßt sich darstellen durch eine Permutation π von $\{1,...,n\}$, so daß die Tour wie folgt aussieht : $(v_{\pi(1)},v_{\pi(2)}),...,(v_{\pi(n)},v_{\pi(1)})$.

Def.2.2.4.: Hamiltonzykel

Ein **Hamiltonzykel** durch einen Graph ist eine Tour, bei dem jeder Knoten genau einmal besucht wird.

Def.2.2.5.: **Symmetrisches TSP**

Sei K_n ein vollständiger, symmetrischer Graph mit den Kantengewichtungen c_{ij} für $i,j \in \{1,...,n\}$. Das (symmetrische) TSP besteht darin, den kürzesten Hamiltonzykel durch K_n zu finden, d.h. eine Permutation π von $\{1,...,n\}$ so daß $\sum_{i=1}^{n} c_{\pi(i),\pi((i \bmod n)+1)}$ minimal ist. Diese kürzeste Tour wird auch **optimale Tour** genannt.

Anwendung findet das TSP in vielen Gebieten, z.B. dem VLSI-Entwurf, Scheduling in Betriebssystemen und natürlich der Routenplanung.

Es gibt unterschiedliche Lösungsverfahren für das TSP, die mit verschiedenen Ansätzen arbeiten. Zu den *exakten Lösungsverfahren* zählen beispielsweise Exhaustive Search sowie Branch and Bound. Diese Algorithmen sollen hier nicht näher erörtert werden (s.[Leo95]), stattdessen wollen wir uns mit Approximationsalgorithmen befassen, d.h. Verfahren die nicht unbedingt die optimale Tour finden, aber versuchen, ihr möglichst nahe zu kommen. Wir verwenden dazu zwei verschiedene Varianten: Tourkonstruktionsverfahren und Tourverbesserungsverfahren.

2.2. Tourkonstruktionsverfahren

Diese Klasse von Algorithmen besteht aus solchen, die neue, möglichst kurze Touren nur mit Hilfe der Entfernungsmatrix erzeugen. Hierzu können verschiedene Strategien verwendet werden.

2.2.1. Nearest Neighbour

Dieses Verfahren stellt eine Greedy-Methode dar (vgl. [MadO]) und arbeitet nach folgendem Prinzip: man startet bei einer beliebigen Stadt und geht dann weiter zu derjenigen Stadt, die (bezogen auf die Kosten in der Entfernungsmatrix) am nächsten ist. Von dort geht man weiter und nimmt nach dem gleichen Prinzip sukzessive alle noch nicht besuchten Städte in die Tour auf, bis man schließlich wieder zu der Startstadt zurückkehrt. Auf diese Weise entsteht eine komplette Rundreise.

Es ist sicherlich einsichtig, daß man mit dieser Methode zwar relativ schnell zu einem Ergebnis kommt, jedoch i.a. nicht die optimale Tour gefunden wird. In der Praxis hat sich diese Methode als recht schwach erwiesen. Die folgenden Insertion-Verfahren liefern demgegenüber wesentlich bessere Ergebnisse.

2.2.2. Insertion-Verfahren

Insertion-Verfahren beruhen gleichsam auf dem Prinzip, daß mit einer kleinen (trivialen) Tour begonnen wird, die noch sehr wenige Städte enthält (z.B. nur eine), und man diese Tour sukzessive durch Aufnahme von unbenutzten Städten erweitert, wobei darauf geachtet wird, daß das Einfügen minimales Ansteigen der Tourlänge verursacht. Entscheidend ist hierbei die Heuristik zur Auswahl der nächsten einzubauenden Stadt. Die Wahl der Starttour bzw. Startstadt hat übrigens auch Einfluß auf die resultierende Tour, was jedoch abhängig vom speziellen TSP ist. Vorgestellt werden an dieser Stelle vier verschiedene Insertion-Arten; desweiteren sind bekannte Verfahren Largest Sum Insertion und Smallest Sum Insertion, auf die hier jedoch nicht weiter eingegangen wird. Sie werden in [Re94] genauer beschrieben.

1. Nearest Insertion

Bei dieser Heuristik wird diejenige Stadt als nächste einzufügende ausgewählt, die die geringste Entfernung zu der bestehenden Tour hat, d.h. deren minimaler Abstand zu den Städten der Tour am kleinsten ist.

2. Cheapest Insertion
Es wird die Stadt als nächstes ausgewählt die die Tour am wenigsten in ihrer Länge ansteigen läßt. Man erhofft sich davon, daß auch die resultierende Tour dann eine möglichst kleine Länge hat.

3. Farthest Insertion
Man wählt diejenige Stadt als nächste aus, deren Abstand zur bestehenden Tour maximal ist, d.h. deren minimaler Abstand zu den Städten der Tour am größten ist. Obwohl dies nicht sofort einsichtig ist, erzielt diese Heuristik in der Praxis die besten Ergebnisse der bekannten Insertion-Verfahren. Erklären läßt sich dieses Ergebnis damit, daß durch die frühzeitige Aufnahme weit entfernter Städte die Tour sich rasch auf das Gesamtbild des TSP einstellt. Es wird praktisch erst ein grobes Gerüst aufgebaut, das dann schrittweise durch das Aufnehmen weiterer Städte verfeinert wird.

4. Arbitrary Insertion
Dieses Verfahren ist denkbar einfach: es wird zufällig die nächste einzufügende Stadt aus der Menge der unbenutzten Städte ausgewählt und eingefügt. Durch diese probabilistische Komponente *kann* das Verfahren in einigen Fällen gute Ergebnisse produzieren, jedoch sind diese i.a. nicht reproduzierbar. Dies ist auch ein allgemeines Problem von TSP-Lösungsverfahren die Zufallsgeneratoren verwenden.

Es zeigt sich, daß Insertion-Algorithmen wesentlich bessere Ergebnisse liefern als der einfache Greedy-Ansatz Nearest Neighbour.
Schließlich lassen sich die vorgestellten Verfahren abwandeln, um ein besseres Laufzeitverhalten zu erreichen. In der Praxis muß man offenbar immer einen Kompromiß eingehen zwischen Laufzeit und Güte der Ergebnisse; exakte Lösungsverfahren sind oft zu aufwendig, während Näherungsverfahren die Gefahr beinhalten, nicht ausreichend gute Ergebnisse zu produzieren. Es zeigt sich aber, daß man mit Approximationsalgorithmen dennoch sehr gute Ergebnisse bekommt, womit auf das Kapitel 4 verwiesen wird, das die Umsetzung des Teamwork-Konzeptes auf das TSP in Form des Systems DOTT beschreibt.

2.3. Tourverbesserungsverfahren

Hierbei handelt es sich um Algorithmen zur Verbesserung bestehender Touren, d.h. es wird eine bestehende Tour als Ausgangspunkt genommen und so modifiziert, daß eine kürzere Tour entsteht. Tourverbesserungsverfahren zählen zu den sog. Anytime-Algorithmen, d.h. man kann das System zu einem beliebigen Zeitpunkt anhalten und erhält in jedem Fall eine (approximierte) Lösung.
Das System DOTT kombiniert diese Verfahren mit den Konstruktionsalgorithmen: Es wird zuerst eine Menge von Touren generiert und auf diese dann Verbesserungsverfahren angewendet, so daß man eine immer bessere Qualität erreicht. Der Benutzer kann dann zu einem beliebigen Zeitpunkt die Berechnung abbrechen, sobald eine für ihn ausreichend gute Lösung gefunden wurde.

2.3.1. 2-Opt

Bei dieser Heuristik werden aus einer bestehenden Tour zwei Kanten herausgenommen (die nicht direkt nebeneinander liegen) und die beiden resultierenden Teiltouren durch zwei andere Kanten wieder zusammengesetzt. Hierbei gibt es genau eine Möglichkeit, sie wieder zu verbinden, wenn man nicht wieder die Ausgangstour erhalten möchte. Natürlich wird man einen 2-Opt-Schritt nur dann anwenden wenn die resultierende Tour tatsächlich kürzer ist als die ursprüngliche. Die Freiheit bei diesem Verfahren besteht in der Wahl der aufzubrechenden Kanten, wobei meist probabilistische Auswahlmethoden eingesetzt werden. Läßt sich eine Tour nicht mehr durch einen 2-Opt-Schritt verkürzen, so spricht man von einer *lokal optimalem* Tour bzgl. 2-Opt. Dies gilt auch entsprechend für ähnliche Verfahren, etwa 3-Opt.
Kombiniert man 2-Opt mit dem Nearest-Neighbour-Verfahren, so erhält man in der Praxis durchschnittlich Lösungen die etwa 9% länger als die optimale Tour sind ([Re94]), während die von Nearest Neighbour erzeugten Lösungen im Durchschnitt etwa 25% länger waren. Dies zeigt die Leistungsfähigkeit von Verbesserungsheuristiken; die folgende 3-Opt-Methode liefert sogar noch bessere Ergebnisse.

2.3.2. 3-Opt

Eine logische Fortführung von 2-Opt stellt das 3-Opt-Verfahren dar. Anstatt daß die Tour in zwei Teiltouren durch Herauslösen von zwei Kanten geteilt wird, nimmt man drei Kanten aus ihr heraus, so daß drei Teiltouren entstehen. Nun gibt es acht verschiedene Möglichkeiten, aus den drei Teilstücken wieder eine komplette Tour zusammenzusetzen. Ein 2-Opt-Schritt läßt sich durch einen 3-Opt-Schritt simulieren, indem eine der drei aufgebrochenen Kanten wieder benutzt wird. Auch bei 3-Opt sind wieder Freiheitsgrade gegeben durch die Auswahl der drei Kanten, wobei hierfür meist Zufallsalgorithmen verwendet werden. 3-Opt hat offensichtlich eine größere Komplexität und damit Laufzeit als 2-Opt. Bei der Wahl des richtigen Verfahrens kommt es darauf an, ob man mehr Wert auf Laufzeit oder Qualität der Lösung legt. Kombiniert man 3-Opt mit dem Nearest-Neighbour-Verfahren, so erhält man in der Praxis Lösungen die im Schnitt ca. 5% länger als die optimale Lösung sind.

2.3.3. Lin-Kernighan

Es wäre nun zu aufwendig, die Verallgemeinerung von 2-Opt bzw. 3-Opt, also ein k-Opt-Verfahren anzuwenden, da die Komplexität exponentiell mit k steigt. Aus dieser Überlegung heraus entstand das Lin-Kernighan-Verfahren. Die Idee hierbei ist, eine komplexe Tourmodifikation aus vielen einfachen Schritten aufzubauen. Hierbei ist es für die Einzelschritte nicht zwingend vorgeschrieben, daß die Tour kürzer wird. Übernommen wird schließlich aber diejenige Modifikation, die die Tour am stärksten verkürzt. Tourverbesserungsverfahren haben die Tendenz, zu einem lokalen Minimum hin zu konvergieren, ohne jemals die optimale Lösung zu finden (vgl. 2.4. Genetische Algorithmen). Das Lin-Kernighan-Verfahren kann bis zu einem bestimmten Grad lokalen Minima entkommen, da während eines Lin-Kernighan-Schrittes die Tour auch in ihrer Länge ansteigen darf. Trotzdem muß die resultierende Tour aber in jedem Fall kürzer sein, um den Schritt durchführen zu können. Es zeigt sich, daß Lin-Kernighan eine sehr leistungsfähige Heuristik ist; in Kombination mit Nearest Neighbour erreicht man im Durchschnitt Lösungen die 3% über der optimalen Lösung liegen.

2.4. Genetische Algorithmen

Immer wieder wurde versucht, Verhaltensweisen der Natur auf Rechnersysteme zu übertragen. Neben dem Simulated Annealing (s. [Otto94]) werden in diesem Zusammenhang vor allem Evolutionsstrategien und genetische Algorithmen genannt ([Blü91], [Rie93]). Insbesondere bei Optimierungsproblemen lassen sich mit diesem Ansatz erstaunliche Ergebnisse vorweisen. Entwickelt wurde diese Idee in den sechziger Jahren von John H. Holland an der Universität von Michigan. Genetische Verfahren basieren auf dem Prinzip der natürlichen Evolution (vgl. [Mi92]). Eine Menge von *Individuen*, *Population* genannt, wird über die Operationen *Mutation* und *Crossover* sowie *Selektion* auf einen immer höheren Stand gebracht; mit Hilfe einer sog. *Fitness-Funktion* (auch Qualitäts- oder Zielfunktion genannt) entfernt man schwächere Individuen aus der Population, so daß die stärksten, d.h. die besten im Sinne der Spezifikation, überleben. Eine bedeutende Rolle spielt hierbei der Zufall, so daß diese Art von Algorithmen i.a. nicht deterministisch arbeiten; ein Durchlauf eines genetischen Verfahrens läßt sich also nicht ohne weiteres wiederholen. Großen Einfluß auf den Erfolg hat auch die Fitness-Funktion; sie legt fest, welche Individuen im gegebenen Zusammenhang als "gut" oder vielversprechend anzusehen sind.
Die zwei erzeugenden Funktionen von genetischen Algorithmen sind:

a) Crossover :

Eltern: Individuum A (x1, x2, x3, x4, x5, x6) Individuum B (y1, y2, y3, y4, y5, y6)

Crossover

(x1, y2, y3, y4, x5, x6)
Nachkomme

Abb.2.4.1.: Crossover-Funktion im Beispiel

Individuen der Population besitzen bestimmte Eigenschaften, in diesem Zusammenhang meist *Gene* genannt. Beim Crossover werden nun mehrere (meist zwei) Individuen als Eltern ausgewählt und ihre Gene derart kombiniert, daß ein neues Individuum (der Nachkomme) entsteht. Diese Kombination kann natürlich auf die unterschiedlichsten Weisen geschehen, was auch der hauptsächliche Freiheitsgrad neben der Wahl von geeigneten Eltern darstellt. Man versucht, dem Nachkommen möglichst die positiven Eigenschaften der Eltern zu vererben und erhofft sich davon, ihn auf einen höheren Stand als beide Eltern zu bringen und somit die Evolution voranzutreiben. Da Crossover vorhandene Individuen benutzt, ist implizit das Vorhandensein einer initialen Population vorausgesetzt. Im Beispiel besitzt jedes Element der Population sechs Gene, die auf eine bestimmte Weise zu einem Nachkommen vermischt werden.

b) Mutation :

Individuum 1 (x1, x2, x3, x4, x5, x6)

$\downarrow$ Mutation

Individuum 2 (x1', x2, x3', x4', x5, x6)

Abb.2.4.2.: Mutations-Funktion im Beispiel

Individuum 2 entsteht aus Individuum 1 durch Manipulation seiner Gene, also Veränderung seiner Eigenschaften. Hiervon erhofft man sich wiederum, daß auf diese Weise ein im Sinne der Fitness-Funktion besseres Individuum entsteht. Sind die Gene von binärer Form, so bedeutet Manipulation eines Gens schlicht und einfach Invertieren. Die Mutation operiert auf bestehenden Elementen der Population, setzt also (wie Crossover) eine initiale Population voraus. Im Beispiel stellen x' manipulierte Gene dar.
Mutation und Crossover sind also die Funktionen die die Population anwachsen lassen, es entstehen neue Individuen. Die Fitness-Funktion sorgt dann für Selektion, d.h. eine Teilmenge der Population stirbt, während die anderen Elemente überleben. Wichtig ist, daß man eine Ausgangspopulation benötigt, um eine der beiden Funktionen anzuwenden; diese wird auch *Initialpopulation* genannt.

Hier noch einmal das Gesamtkonzept im Schaubild :

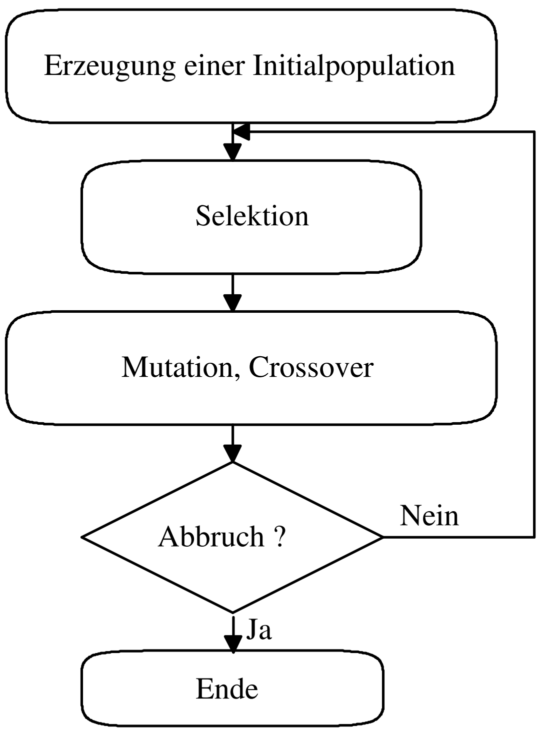

Abb. 2.4.3.: Ablauf von genetischen Algorithmen

Die Schritte Crossover, Mutation und Selektion wiederholen sich für jede *Generation* genau einmal.

Im klassischen Ansatz genetischer Algorithmen findet man immer wieder die Forderung nach binärer Darstellung der Gene, d.h. Individuen werden durch eine Kette von Dualzahlen repräsentiert. Mutation und Crossover geschieht dann ebenfalls auf dieser binären Ebene durch Bit-Manipulation, um der ursprünglichen biologischen Idee möglichst nahe zu kommen (ein biologisches Gen kann vier Zustände annehmen, bei Computern bieten sich zwei Zustände an).
Versucht man nun, dieses klassische Konzept auf das TSP anzuwenden, muß man viele Schwierigkeiten in Kauf nehmen. Neben der Kodierung von Touren in binärer Darstellung ist es nämlich notwendig, zusätzlich darauf achten, daß bitmanipulierte Touren immer noch korrekte Touren im Sinne ihrer Spezifikation sind. Man spricht in diesem Zusammenhang auch von *lebensfähigen* Individuen. Man muß nun entweder von vornherein verhindern, daß solche inkorrekten Touren erzeugt werden, oder sie nachträglich wieder löschen. Dadurch ergibt sich ein erheblicher zusätzlicher Aufwand und es wird u.U. viel überflüssige Arbeit getan. Ein geschickterer Ansatz besteht in der Abstrahierung der genetischen Idee, indem Mutation und Crossover nicht mehr für Bit-Manipulationen, sondern allgemein für Veränderung und Paarung stehen. Dies bringt eine bessere Ausnutzung der speziellen Eigenschaften des TSP, wodurch an Effizienz gewonnen wird. Nun gibt es verschiedene Ansätze wie man diese zwei Operationen Mutation und Crossover auf das TSP anwendet. Man kann die Eltern (beim Crossover) ja auf verschiedene Weise miteinander rekombinieren, was völlig unterschiedliche Strategien impliziert. Mutation von Touren kann ebenfalls auf die verschiedensten Arten geschehen. Im folgenden werden konkrete Ideen hierzu angegeben.

2.4.1. Edge Recombination Crossover

Dieses Crossover-Verfahren arbeitet nach folgendem Prinzip. Es werden aus der Population zufällig zwei Elemente als Eltern ausgewählt. Ein Nachkomme wird nun dadurch erzeugt, daß möglichst viele gemeinsame Kanten der beiden Eltern übernommen werden. Hierbei muß darauf geachtet werden, daß der Nachkomme tatsächlich eine vollständige Tour ist. Während der Konstruktion kann es daher zu sog. *Edge Failures* kommen, d.h. ausgehend von einer Stadt kann man weder von dem einen noch dem anderen Elternteil eine ausgehende Kante übernehmen, ohne die Tour inkorrekt werden zu lassen. Es ist offensichtlich, daß dieser Effekt vor allem bei sehr unterschiedlichen Eltern auftritt. Edge Failures werden behoben durch Ergänzung des Nachkommen um geeignete Kanten die nicht bei den Eltern zu finden sind. Diese "Reparatur" ist eine Veränderung des direkten Nachkommen; daher stellt Edge Recombination Crossover eigentlich eine Kombination von Crossover und Mutation dar. Da jedoch die wesentliche Idee dieses Verfahrens das Crossover ist, ist der gegebene Name eine sinnvolle Bezeichnung.
Eine Abwandlung des ER Crossover ist, statt gemeinsamer Kanten der Eltern gemeinsame Teiltouren zu betrachten und den Nachkommen dementsprechend zu bilden.

2.4.2. 4-Opt-Mutation

Mutation beim TSP kann auf die unterschiedlichsten Weisen geschehen. 4-Opt ist eigentlich ein Tourverbesserungsverfahren, trotzdem kann man eine Abwandlung davon als Mutationsheuristik verwenden. Es wird hierbei eine beliebige Tour aus der Population ausgewählt. Danach werden zufällig vier Kanten dieser Tour entfernt und die resultierenden Teiltouren auf beliebige Weise wieder zusammengesetzt (vgl. 2-Opt, 3-Opt), wobei nicht zwingend vorgeschrieben ist, daß die so entstehende Tour kürzer als die ursprüngliche ist. Es geht hierbei lediglich um eine von außen herbeigeführte Verfremdung der Tour.

2.4.3. Fitneß-Funktion

Die Qualität einer Tour beim TSP zu bestimmen ist alles andere als trivial, gilt es doch zu spezifizieren, was "gut" in diesem Zusammenhang bedeutet. Die intuitivste Lösung ist die, einfach die Länge der Tour zu betrachten: je kürzer eine Tour ist, desto größer ist auch ihre Qualität. Trotzdem kann man dies nicht immer so einfach sagen, da in der Praxis das Problem der lokalen Minima auftritt: eine Tour hat sich schon in einem großen Maße zu einer bestimmten Form entwickelt, in der sie nicht mehr ohne weiteres zu der optimalen Lösung gelangen kann. Es kann nötig sein, die Tour derart durch Mutation bzw. Crossover zu verändern, daß sie vorübergehend länger wird, um einem lokalen Minimum zu entkommen. Verglichen werden kann dies mit dem Newtonschen Näherungsverfahren zur Nullstellensuche: hat man sich erst in das Gebiet einer bestimmten Nullstelle bewegt, wird man unweigerlich zu dieser Nullstelle hin konvergieren und eine andere nicht mehr erreichen können. Eine weitere Parallele findet sich auf dem Gebiet des automatischen Theorembeweisens: hier entspricht das Bilden von Normalformen dem Erreichen von lokalen Optima. Zur Verdeutlichung nun eine graphische Darstellung des allgemeinen Problems (Abb. 2.4.4.). Individuen, die sich in großem Maße in die Nähe eines lokalen Optimums entwickelt haben, können nicht mehr ohne weiteres das globale Optimum finden.

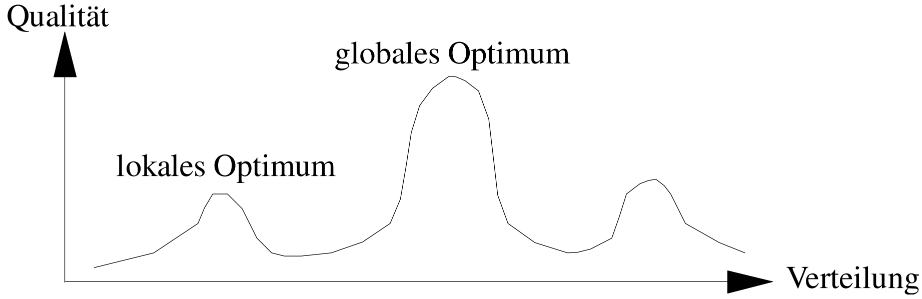

Abb. 2.4.4.: Problem der lokalen Optima

Es kann also notwendig sein, auch auf den ersten Blick nicht so gute Touren überleben zu lassen, da es sich vorher nicht mit Gewißheit vorhersagen läßt, welche Touren vielversprechend sind, d.h. das Potential haben, zur optimalen Lösung zu finden. Man wird in der Praxis wohl immer einen Kompromiß eingehen müssen zwischen Effizienz der Verfahren und dem Arbeiten mit einem möglichst breiten Spektrum von Touren.
Mit Hilfe der Teamwork-Methode läßt sich das Problem der lokalen Optima noch auf eine andere Weise angehen. Hier erlauben die Gutachter eine erneute Selektion der "besten" Touren bei den Teamtreffen. Verfügt ein Verfahren also über eine mäßige Fitness-Funktion, so läßt sich durch die Integration in Teamwork dieses Manko zum Teil beseitigen; ersetzen läßt sich die Fitness-Funktion auf diese Weise aber nicht. Der in dieser Arbeit entwickelte Gutachter REF_DIVERS benutzt diesen Ansatz; er berücksichtigt je nach Parametergewichtung auch die Unterschiedlichkeit von Touren.

2.4.4. Kombination mit anderen Verfahren

Um die Effizienz genetischer Algorithmen zu erhöhen, kann man sie mit anderen Tourverbesserungsverfahren kombinieren. Johnson stellt in [Jo90] einen "parthenogenetischen Algorithmus" vor, der eine Kopplung von 4-Opt mit Lin-Kernighan darstellt (vgl. [Leo95] Abschnitt 4.5.6). Die Idee hierbei ist die, nach der Mutations-Operation (4-Opt) ein weiteres Optimierungsverfahren (Lin-Kernighan) für jede Tour anzuwenden. Der Vorteil dabei ist der, daß erzeugte Touren i.a. besser sind als die des reinen genetischen Verfahrens. Touren, die durch die Mutation länger geworden sind, können durch den "Optimierer" nachgebessert werden, so daß sie eher von der Fitness-Funktion ausgewählt werden Dies ist wichtig im Zusammenhang mit dem Problem der lokalen Minima. Setzt man dieses Prinzip logisch fort, so lassen sich beliebige genetische Verfahren (Mutation oder Crossover) mit einem beliebigen weiteren Optimierungsverfahren wie 2-Opt, 3-Opt oder Lin-Kernighan koppeln. Es zeigt sich in der Praxis, daß durch diese Kopplung wesentlich schneller gute Touren gefunden werden. Der größte Vorteil gegenüber den reinen genetischen Verfahren liegt also in der Geschwindigkeit.
Hier die graphische Darstellung der allgemeinen Idee.

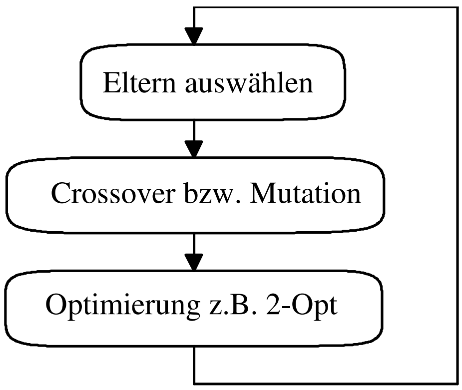

Abb. 2.4.5.: Kombination mit genetischen Verfahren

Im System DOTT sind vier Experten dieser Form implementiert: Edge Recombination Crossover und 4-Opt-Mutation werden jeweils mit den Verfahren 3-Opt bzw. 2-Opt kombiniert. Es zeigt sich in Testläufen, daß durch die Koppelung hervorragende Ergebnisse erzielt werden. Durch die Integration in das im nächsten Kapitel vorgestellte Teamwork-Konzept ist sogar eine weitere Leistungssteigerung möglich.

3. Die Teamwork-Methode

Die Lösung von großen Traveling Salesman Problemen bedeutet i.a. einen sehr hohen Rechenaufwand. Aus diesem Grund werden hierzu parallele und verteilte Rechnerarchitekturen eingesetzt. Eine gute Möglichkeit zu dieser Umsetzung ist das Teamwork-Verfahren, ein Multiagentensystem, das bereits erfolgreich auf das Problem des verteilten Theorembeweisens angewendet wurde und auf dem auch das System DOTT basiert. Eine ausführliche Beschreibung findet sich in [De93]. In diesem Kapitel wird die Teamwork-Methode in allgemeiner Form beschrieben, sowie erläutert, welche Punkte bei der Anwendung auf das TSP beachtet werden müssen.

3.1. Prinzip von Teamwork

Die Idee der Teamwork-Methode ist angelehnt an den Aufbau von menschlichen Projektteams eines Unternehmens bzw. einer Bildungseinrichtung (vgl. [Ye86]). Hierbei setzt sich ein Team zusammen aus mehreren *Experten* (Mitarbeitern), Stabsmitgliedern und einem *Leiter*. Dieser hat die Aufgabe, sein Team zu koordinieren und zu steuern; er tut dies mit Hilfe der Stabsmitglieder, die ihm von der Arbeit der Experten berichten. Die Experten bekommen alle den gleichen Auftrag, der von ihnen dann jeweils weitgehend selbständig bearbeitet wird. Von Zeit zu Zeit werden *Teamtreffen* einberufen, bei denen die Ergebnisse der Arbeiten der einzelnen Experten mit Hilfe der Stabsmitglieder gesammelt werden und von dieser Grundlage aus der weitere Weg durch den Leiter festgelegt wird. Hierbei stehen die Experten auf der einen Seite in *Konkurrenz* zueinander; da jeder möglichst gute Arbeit vorweisen will; auf der anderen Seite *kooperieren* sie aber auch, da gute Ergebnisse jedes Experten für die weitere Arbeit verwendet werden. Eine besondere Rolle bekommt der Leiter zugewiesen: er bestimmt die Zusammensetzung seines Teams und koordiniert es. Um diese Idee der Projektteams auf Rechnersysteme umzusetzen, ist es nötig, einige Anpassungen vorzunehmen. Die Experten sind in diesem Fall Algorithmen, die sich mit dem eigentlichen Lösen der Aufgabe beschäftigen. Zusätzlich benötigt werden sog. *Gutachter*, die den Stabsmitgliedern im menschlichen Projektteam entsprechen und die dem Leiter zum einen Aufschluß darüber geben, wie gut ein Experte ist und die ihm zum anderen die "guten" Ergebnisse übermitteln. Hier eine Abbildung der Hierarchie des Teams:

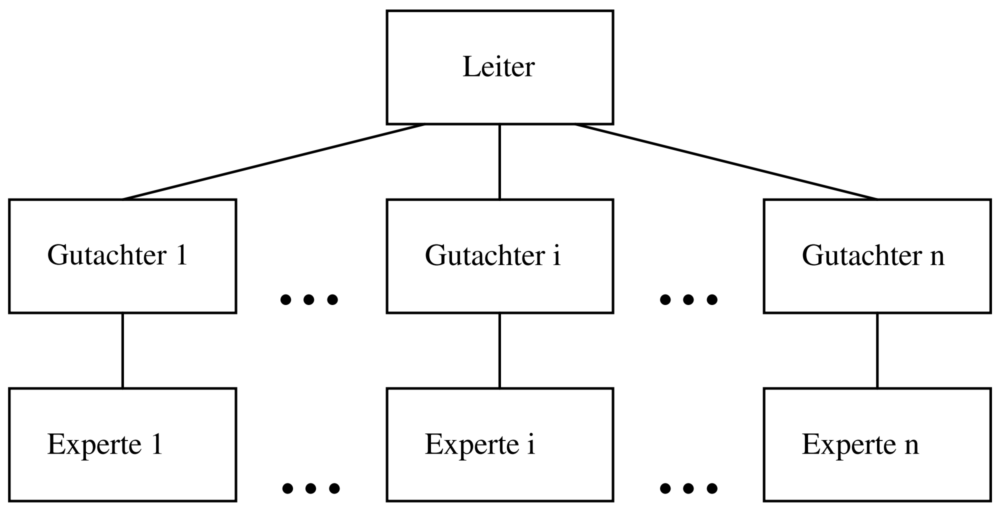

Abb. 3.1.: Hierarchie der Komponenten in Teamwork.

Es ist sinnvoll, die Gutachter und ihre jeweiligen zugeordneten Experten auf dem gleichen Prozessor ausführen zu lassen, da auf diese Weise Kommunikationszeit gespart wird, indem auf die lokal vorhandenen Daten zugegriffen werden kann. Dies ist bei verteilten Systemen besonders wichtig, da die Vorteile der Verteilung durch ein zu großes Maß an Kommunikation wieder zunichte gemacht werden können. In Teamwork findet Kommunikation daher nur während der Teamtreffen statt. Ein *Teamlauf* findet nun auf die folgende Art statt. Der Leiter stellt ein initiales Team von Experten und Gutachtern zusammen. Während der Arbeitsphase sind die Experten dann damit beschäftigt, die vom Leiter vorgegebene Aufgabenstellung in der ebenfalls von ihm fest vorgegeben Zeit zu bearbeiten. Nach Ablauf der Arbeitsphase der Experten treten die Gutachter in Aktion; es findet ein Teamtreffen statt. Die Gutachter teilen dem Leiter die Bewertungen ihrer jeweiligen Experten mit, woraus dieser einen *Sieger* unter den Experten wählt. Der Sieger ist offenbar zu diesem Zeitpunkt am besten für die Aufgabe geeignet, seine Ergebnisse werden daher vollständig für den weiteren Lösungsprozeß übernommen. Zusätzlich nimmt der Leiter die besten Ergebnisse der anderen Experten an (die ihm von den Gutachtern übermittelt werden), woraus er eine neue Aufgabenstellung für die Experten sowie ein ggf. verändertes Team bestimmt. Hier die graphische Darstellung eines Zyklus zwischen zwei Teamtreffen:

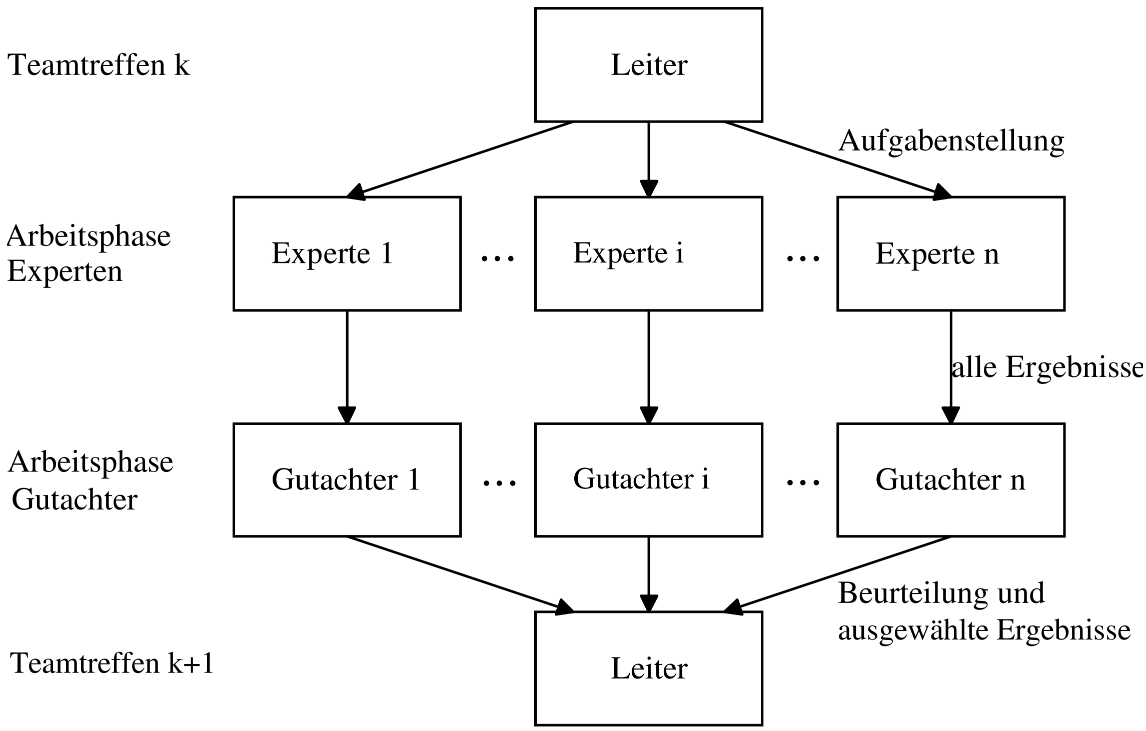

Abb. 3.2.: Zyklus zwischen zwei Teamtreffen

Man sieht aus Abb. 3.2., daß die Experten nur während ihrer Arbeitsphase aktiv sind, während die Gutachter und der Leiter es während der Teamtreffen sind. Kommunikation im Team findet nur während der Teamtreffen statt. Es ist übrigens möglich, eine Verteilung der Komponenten auf Prozessoren zu erreichen, so daß man genau die Anzahl von Prozessoren benötigt wie Experten im Team sind (vgl. [Leo95], Kapitel 3.3). Nun sind manche Experten für das vorgegebene Problem gut geeignet, während andere schlechter sind (dies gilt

insbesondere für das TSP !). Da es sich i.a. nicht von vornherein feststellen läßt, welches Verfahren das beste ist, hat man in Teamwork den großen Vorteil, verschiedene Ansätze parallel laufen lassen zu können, wobei das zum aktuellen Zeitpunkt beste Verfahren zum Sieger erklärt wird und somit seine gesamten Ergebnisse an den Leiter übermitteln darf. Auf diese Weise ist es möglich, die guten Eigenschaften verschiedener Algorithmen zu kombinieren: ein Verfahren könnte beispielsweise in der Anfangsphase besonders gut sein, aber zu einem späteren Zeitpunkt schlechter geeignet sein als ein anderes Verfahren, welches dann (zum späteren Zeitpunkt) durch den Leiter zum Sieger erklärt wird. Hierdurch lassen sich Ergebnisse erzielen, die besser sind als die jeweils beste Lösung beider Experten zusammen. Man spricht in diesem Zusammenhang auch von Synergie-Effekten. Dadurch daß nicht nur die Ergebnisse des Siegers verwendet werden, sondern auch die guten der Nicht-Sieger, erreicht man mittels Teamwork eine Kooperation der Experten, bei der die am besten geeignete Komponente auch am stärksten in den Lösungsprozeß eingeht, ohne daß die anderen vernachlässigt werden. Eine weitere Stärke von Teamwork ist, daß der Leiter je nach Verlauf der Arbeit sein Team verändern kann: er kann für die gegebene Aufgabe schlecht geeignete Experten aus dem Team entfernen sowie einen gut arbeitenden Experten besonders berücksichtigen. Der Leiter verfolgt dabei eine bestimmte Strategie. In Teamwork findet sich also eine gelungene Kombination aus Konkurrenz und Kooperation. Es ergibt sich (bei entsprechend vorhandenem Leiter) eine dynamische Anpassung an das Problem, d.h. während des Teamlaufes wird auf dessen Verlauf reagiert.
Im folgenden wird eine Beschreibung der Komponenten von Teamwork im einzelnen gegeben.

3.1.1. Experten

Experten befassen sich mit dem eigentlichen Lösen der Aufgabe. Vor Beginn ihrer eigentlichen Arbeitsphase bekommen alle am Teamlauf beteiligten Experten die gleiche Menge von *Startresultaten* vom Leiter geschickt, auf deren Basis sie für eine fest vorgegebene Zeit ihr jeweiliges Verfahren anwenden. Sie produzieren dabei ihre sog. *Arbeitsresultate*, aus denen der Gutachter nach der Arbeitsphase dann die Qualität der Ergebnisse der Experten (bezogen auf die aktuelle Problemstellung) bestimmt. Die Experten haben selbstverständlich unterschiedliche Arbeitsweisen und Heuristiken. Das Besondere ist hierbei, daß jeder Experte für sich in der Lage wäre, das Problem zu lösen, wenn man ihm nur genug Rechenzeit gäbe. Ausnahme hiervon bilden die sog. *Spezialisten*, die sich entweder für spezielle Probleme besonders gut eignen oder die eine Aufgabe haben die nur indirekt zur Lösungsfindung beiträgt. Daher ist es sinnvoll, für Spezialisten auch besondere, auf sie abgestimmte Gutachter einzusetzen. Im Gegensatz zu anderen Experten ist bei den Spezialisten nicht gefordert, daß sie vollständig sind, d.h. das Problem prinzipiell alleine lösen könnten. Es ist Aufgabe des Leiters, je nach aktuellem Problemzustand geeignete Spezialisten einzusetzen. In Teamwork arbeiten also mehrere verschiedene Lösungsalgorithmen parallel an der gleichen Aufgabe. Durch die Teamtreffen ergänzen sich die Experten dann gegenseitig. Wichtig ist, daß alle Experten zu Beginn eines Zyklusses die gleiche Aufgabenstellung bekommen, d.h. initial das gleiche Wissen vom Leiter erhalten.

3.1.2. Gutachter

Die Gutachter sind die analysierenden und bewertenden Komponenten in Teamwork. Der Leiter ist auf ihre Arbeit angewiesen, um erkennen zu können, wie gut die einzelnen Experten gearbeitet haben und welches ihre besten Ergebnisse sind. Gutachter haben also zwei verschiedene Aufgaben; zum einen das Bewerten der Gesamtarbeit des ihnen zugeordneten

Experten, d.h. die Berechnung eines global vergleichbaren *Fortschrittsmaßes*, auf dessen Grundlage der Leiter die Siegerwahl durchführen kann. Zum anderen das *Auswählen von guten Resultaten* der Experten, wobei vom Sieger grundsätzlich alle Ergebnisse übernommen werden, sowie von den Nicht-Siegern eine vom Leiter maximal fest vorgegebene Anzahl. Die Gutachter besitzen also eine sehr wichtige Rolle in Teamwork, da sie es sind, die bestimmen, was "gut" zum aktuellen Zeitpunkt im Lösungsprozeß bedeutet. Verschiedene Gutachter sind daher i.a. auch unterschiedlich gut geeignet für die gegebene Problemstellung. Es ist Aufgabe des Leiters, hier auf das jeweilige Problem angepaßte Gutachter einzusetzen. Kriterien für Bewertungen der Experten durch die Gutachter kann man selbstverständlich nur für den jeweiligen Problemtyp festlegen, dennoch gibt es auch allgemeine Gesichtspunkte: ein Gutachter könnte beispielsweise auf die Anzahl der erzeugten Fakten achten sowie die Anzahl der gelöschten überflüssigen Fakten (letzteres ist besonders wichtig für das automatische Theorembeweisen); zusätzlich zu diesen statistischen Kriterien kann aber auch eingegangen werden auf die Beschaffenheit der Ergebnisse an sich. Die in dieser Projektarbeit entwickelten Gutachter berücksichtigen z.B. die Unterschiedlichkeit von TSP-Touren voneinander, was für die Bewertung der genetischen Algorithmen wichtig sein kann.

3.1.3. Der Leiter

Der Leiter ist das koordinierende und strategische Element von Teamwork und ist nur während der Teamtreffen aktiv, die aus seiner Sicht wie folgt ablaufen. Mit Hilfe der Beurteilungen der Experten durch die Gutachter erklärt der Leiter einen Experten zum Sieger. Danach fordert der Leiter von den Gutachtern ihre ausgewählten guten Ergebnisse an, wobei vom Sieger die gesamten Resultate übernommen werden. Diese Ergebnisse bilden die Startresultate für den nächsten Arbeitszyklus, zu dem der Leiter ein ggf. verändertes Team von Experten und Gutachtern zusammenstellt. Komponenten, die für den aktuellen Zustand der Lösungsfindung schlecht sind, werden entfernt und durch vielversprechendere andere ersetzt. Schließlich bestimmt der Leiter die Dauer des nächsten Arbeitszyklusses und übermittelt diese sowie die Menge der Startresultate via Broadcast an die Experten. Bei der Zusammenstellung des Teams für den nächsten Zyklus hat der Leiter also Freiheitsgrade. Er könnte beispielsweise Teams in kurzen Zeitphasen austesten und, falls sie sich bewähren, ihnen danach mehr Zeit zur Verfügung stellen. Führt das Team nicht im gewünschten Maß zum Erfolg, so kann der Leiter es nach Belieben verändern. Manche Experten sind für eine gegebene Aufgabenstellung besser geeignet als andere. Nun kann man jedoch i.a. nicht vor dem eigentlichen Programmlauf wissen, welcher Experte am besten geeignet ist. Die Teamwork-Methode bietet eine sehr gute Lösung für dieses Problem an: es lassen sich viele verschiedene Algorithmen parallel ausführen und die Ergebnisse des am besten geeigneten Experten gehen dann vollständig in den weiteren Lösungsprozeß ein, indem der Experte vom Leiter zum Sieger erklärt wird. Bei einem entsprechenden Leiter paßt sich das System also dynamisch auf die Aufgabe an, ohne direkte Einwirkung des Benutzers. Denkbar ist auch eine Art Langzeitgedächtnis des Leiters, in welchem er sich merkt, welche Teams für ein bestimmtes Problem besonders geeignet sind. In diesem Zusammenhang wäre eine Analysefunktion wünschenswert, die das gegebene Problem einer bestimmten Domäne zuordnet, für die der Leiter dann geeignete Teams kennt. Weitere Ideen zur Teamzusammenstellung und zur Leitung finden sich in [DK94]. In DOTT existiert bisher noch kein dynamischer Leiter; der Verlauf eines Teamlaufes wird komplett statisch aus einer Datei gelesen bzw. interaktiv vom Benutzer eingegeben. Von einer automatischen Anpassung an das Problem kann also zum jetzigen Entwicklungszeitpunkt noch nicht die Rede sein, obwohl die automatische Siegerwahl auf Grundlage der Beurteilungen durch die Gutachter bereits implementiert ist.

3.2. Anwendung von Teamwork auf das TSP

Teamwork ist ein allgemeines Verfahren das sich für eine Vielzahl von Aufgaben eignet. In diesem Abschnitt wird nun erläutert, wie die konkrete Umsetzung im Falle des TSP aussehen kann. Neben den obligatorischen Aufgaben wie dem Definieren einer effizienten und für alle Experten gleichartigen Repräsentation der Daten gilt es, die Komponenten Experte, Gutachter und Leiter für das TSP genauer zu spezifizieren. Was die Repräsentation im Speicher betrifft, findet sich in [Leo95] eine ausführliche Darstellung; die einzelnen Komponenten von Teamwork werden im folgenden genauer erläutert.

3.2.1. Experten

Um die Beschaffenheit von Experten beim TSP zu erläutern, ist es wichtig zu wissen, was man in diesem Zusammenhang unter den "Resultaten" versteht. Beim TSP sind die von den Experten produzierten Resultate naheliegenderweise Touren. Ausnahmen hiervon können Spezialisten sein, die beispielsweise Hilfsdaten berechnen wie eine angenäherte untere Schranke des bearbeiteten TSP (s. [Leo95], Kap. 4.4.2). Als Experten können alle in Kapitel 2 vorgestellten TSP-Lösungsverfahren (Tourkonstruktions- sowie Tourgenerierungsverfahren) verwendet werden. Grundlage der Berechnungen eines Experten ist zum einen die gegebene Entfernungsmatrix, also das TSP an sich, und zum anderen die Menge der Starttouren. Zusätzlich können dem Experten die besagten Informationen von Spezialisten zur Verfügung stehen. Es ist selbstverständlich, daß das System das Ziel hat, die optimale Tour oder wenigstens eine fast optimale Tour zu finden. Die in Kapitel 2 erläuterten Verfahren wie Farthest Insertion, genetische Algorithmen oder andere Tourverbesserungsverfahren wie 3-Opt können nun problemlos durch Teamwork in ein Gesamtsystem integriert werden, da jeder Experte die Ergebnisse eines anderen aufgrund der gemeinsamen Datenstruktur weiterverwenden kann. Im System DOTT sind seid der initialen Version drei weitere genetische Experten hinzugekommen, die im nächsten Kapitel genauer erklärt werden; die Ergebnisse der durchgeführten Testläufe lassen sich in Kapitel 5 nachlesen.

3.2.2. Gutachter

Die Gutachter spielen eine wichtige Rolle in Teamwork, da der Leiter ausschließlich aufgrund ihrer gelieferten Beurteilungen sein Vorgehen festlegt. Hat nun ein Gutachter eine schlechte Arbeitsweise, so wird auch der Leiter aufgrund dieser Informationen einen schlechten Teamlauf festlegen. Es ist also notwendig, leistungsfähige Gutachter zu haben, um die Stärken von Teamwork auszunutzen.. Bekanntlich haben die Gutachter zwei Aufgaben: zum einen die generelle Beurteilung der Arbeit des Experten; zum anderen die Auswahl guter Ergebnisse. Beim TSP gibt es nun eine Vielzahl von Kriterien die für diese zwei Aspekte berücksichtigt werden können. Der Charakter des jeweiligen Gutachters ist dann bestimmt durch seine Gewichtung dieser Kriterien. Er kann je nach Verfahren etwa mehr Wert auf Qualität (Länge der Touren) oder Quantität (Anzahl der produzierten Touren) der Arbeitslösungen legen Als Grundlage der Analyse eines Experten hat der Gutachter im Idealfall die Menge der Starttouren sowie die Menge der produzierten Arbeitstouren zur Verfügung. Das Verhältnis und die Beschaffenheit der Arbeitstouren im Vergleich zu den Starttouren bestimmt die

Produktivität und die Qualität des Experten bezüglich der aktuellen Situation im Lösungsprozeß. Was die Aufgabe der Beurteilung betrifft, können beim TSP im wesentlichen folgende Aspekte in Betracht gezogen werden:

- Anzahl der erzeugten Touren
- Länge der kürzesten produzierten Lösung
- Durchschnittslänge aller produzierten Touren
- Durchschnittslänge einer bestimmten Teilmenge der erzeugten Touren (z.B. die n kürzesten)
- Unterschiedlichkeit der Arbeitstouren (wichtig bzgl. lokalen Minima)
- Die Veränderung der Arbeitstouren im Vergleich zu der Startmenge im allgemeinen

Es ist zu erwähnen, daß der erzeugte Fortschrittswert global vergleichbar sein sollte bei gleichzeitigem Einsatz unterschiedlicher Gutachter, um Fairneß zu gewährleisten.

Um die Selektion von guten Ergebnissen durchzuführen, können u.a. folgende Kriterien bezüglich einer gegebenen Tour berücksichtigt werden:

- Länge der Tour
- Unterschiedlichkeit der Tour zu anderen Touren der Arbeitslösungen; z.B. zu der kürzesten Tour, meßbar durch folgende Kriterien:
  - Anzahl der gemeinsamen Kanten mit der zu vergleichenden Tour
  - Anzahl der gemeinsamen Teiltouren mit der zu vergleichenden Tour
  - Länge der größten gemeinsamen Teiltour mit der zu vergleichenden Lösung (hierbei gibt es zwei mögliche Maße: die Kantenanzahl oder die metrische Länge des TSP)
  - Längen der beiden Touren

Der Gutachter geht nun derart vor, daß er jede Tour nach diesen Kriterien bewertet, ihr also einen Gütewert zuweist. Je höher dieser Gütewert, desto wahrscheinlicher wird dann diese Tour an den Leiter übermittelt. Die Anzahl der zu schickenden Touren wird hierbei vom Leiter fest vorgegeben. Um die gute Integration von genetischen Algorithmen in Teamwork zu verdeutlichen, sei diese selektive Aufgabe der Gutachter besonders hervorgehoben. Wie im 2. Kapitel erwähnt, tendieren solche Verfahren dazu, in lokale Minima zu konvergieren. Durch Einsatz von Gutachtern ist es nun möglich, durch geschickte Lösungsauswahl das System in eine bestimmte Richtung zu lenken. Man könnte beispielsweise ein möglichst breites Spektrum von unterschiedlichen Touren auswählen, um sich das Potential zu erhalten, das globale Minimum, also die optimale Tour, zu finden. In DOTT werden die beiden Aufgaben der Gutachter in getrennten Komponenten realisiert. Mehr dazu im nächsten Kapitel.

3.2.3. Der Leiter

Ebenso wie die Gutachter spielt der Leiter eine wichtige Rolle im Team, da seine Strategie über den Verlauf der Lösungsfindung entscheidet. Besitzt er ein geschicktes Verfahren, so paßt er sich dem gegebenen TSP an, d.h. er ändert seine Strategie nach Bedarf und nach Verlauf des Teamlaufes. Im vollständigen Teamwork-System braucht der Benutzer überhaupt keine Einwirkung auf den Verlauf zu tätigen, da der Leiter die Planungsarbeit für ihn erledigt. Ein typischer Teamlauf würde beispielsweise am Anfang eine Menge von Touren mit Hilfe von Tourgenerierungsverfahren erzeugen und danach in beliebig vielen Zyklen (je nach gewünschter Qualität) Tourverbesserungsverfahren anwenden. Bei Bedarf können dann Experten bzw. Gutachter ausgewechselt werden. Wie bereits erwähnt, fehlt dem System DOTT noch solch eine dynamische Leiterkomponente, was auch die hauptsächliche Möglichkeit zur Weiterentwicklung des Systems darstellt.

4. Das System DOTT

Dieses Kapitel befaßt sich mit der eigentlichen Implementierung der Anwendung von Teamwork auf das Traveling Salesman Problem. Es wird eingegangen auf die Komponenten der ursprünglichen Version von DOTT sowie auf die Weiterentwicklung im Rahmen dieser Projektarbeit. Ein Schwerpunkt wird hierbei auf den neu hinzugekommenen Gutachtern liegen. Eine ausführliche Erläuterung der initialen Version von DOTT findet sich in [Leo95], Kap. 4. DOTT wurde auf einem Rechnerverbund von SUN-Sparc-Workstations mit Hilfe des GNU-C-Compilers entwickelt; die verwendete Programmiersprache ist ANSI-C. Kommunikation erfolgt mit Hilfe eines Singlecast/Broadcast-Systems, das in [Li93] genauer erklärt ist.

4.1. Initiale Version

4.1.1. Datenstruktur

Wichtig für die Beschreibung der Datenstruktur der Implementierung ist die Darstellung eines TSP (also der Problemstellung), von Touren sowie Mengen von Touren. Im folgenden wird die gewählte Umsetzung in DOTT erläutert. Die Repräsentation im Speicher ist hierbei für alle Experten gleich, daher ist eine gute Modularität des Systems gegeben.

Ein **Traveling Salesman Problem** wird in DOTT durch seine Entfernungsmatrix (vgl. Kapitel 2) charakterisiert. Die intuitive Lösung, also die Abbildung im Speicher durch ein zweidimensionales Array, ist sehr zeitaufwendig, da neben Addition auch Multiplikation beim Zugriff auf die einzelnen Einträge benötigt wird. Da der Zugriff auf die Matrix sehr häufig verwendet wird, ist es sinnvoll, hier eine effizientere Struktur zu wählen. In DOTT wird hierzu neben den eigentlichen (linear angeordneten) Matrixeinträgen zusätzlich eine Zeigerliste angelegt, durch die auf die einzelnen Zeilen der Matrix direkt zugegriffen werden kann (Abb. 4.1.1.).

Entfernungsmatrix

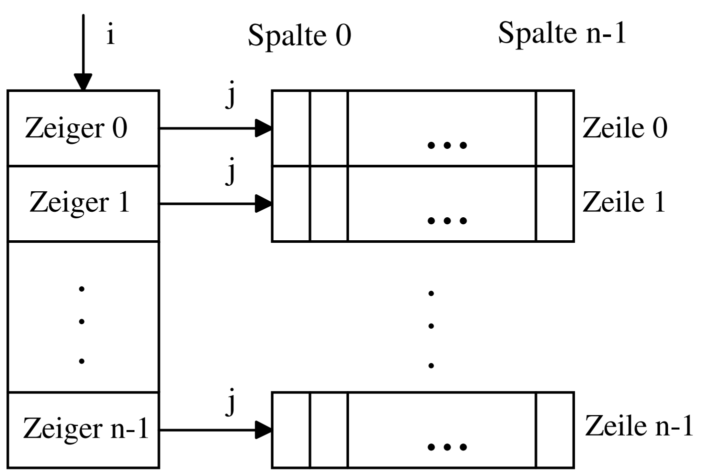

Abb. 4.1.1.: Darstellung der Entfernungsmatrix im Speicher

Man sieht in Abb. 4.1.1., daß durch die Variable i also die Zeile und mit der Variable j die Spalte der Matrix ausgewählt wird, so daß das Matrixelement den Wert $c_{i,j} = Mem(i)(j)$ hat. Diese Realisierung ist wesentlich schneller, da keine Multiplikation benötigt wird; es handelt sich lediglich um Addition und Dereferenzieren. Zusätzlich zur eigentlichen Entfernungsmatrix wird in DOTT selbstverständlich die Anzahl der Städte des TSP gespeichert.

Eine **Tour** wird in DOTT durch eine Nachfolgerliste dargestellt, die durch ein eindimensionales Array realisiert ist. An der i-ten Stelle im Array steht hierbei der Index des Nachfolgers zur Stadt Nr. i. Beispielsweise wird die Tour 0⇨2⇨1⇨3⇨0 durch folgende Nachfolgerliste dargestellt:

i	0	1	2	3
Array[i]	2	3	1	0

Zusätzlich werden zu einer Tour folgende Informationen gespeichert:
- die Länge der Tour
- der Zeitpunkt ihrer Erzeugung
- die Nummer des Experten der sie konstruiert hat
- die Nummer des Rechners auf dem sie erzeugt wurde

Darüberhinaus ist es für einige Experten hilfreich, eine Vorgängerliste zu einer Tour zu haben, d.h. eine Liste, bei der (analog zur Nachfolgerliste) zu jeder Stadt ihre jeweilige Vorgängerstadt gespeichert ist. Einige Experten (z.B. OPT2) erzeugen solche Vorgängerlisten, um effizienter arbeiten zu können, beispielsweise um Teiltouren umzudrehen.

Eine **Menge von Touren** wird in DOTT in einer hybriden, d.h. kombinierten Datenstruktur gespeichert. Zum einen wird ein binärer Suchbaum aufgebaut, in dem jeder Knoten eine Tour repräsentiert. Die Ordnung des Baumes erfolgt mit Hilfe der Längen der Touren. Zum anderen existiert eine lineare Verkettung der Knoten im Baum, die ebenfalls nach der Länge der zugehörigen Touren geordnet ist. Ein Knoten enthält demnach folgende Informationen:
- der Zeiger auf die entsprechende Tour
- Verweise auf den linken und rechten Teilbaum (für die Baumstruktur)
- Zeiger auf den Vorgänger- und Nachfolgerknoten (für die lineare Verkettung).

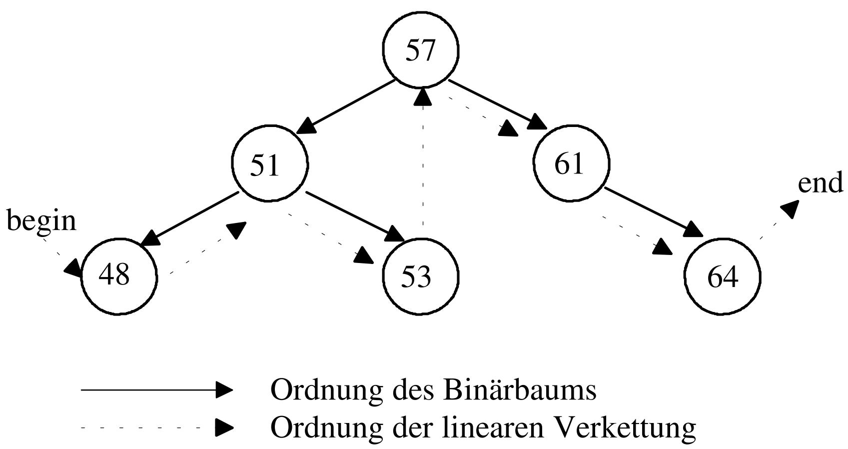

Abb. 4.1.2: Beispiel für die Darstellung einer Menge von Touren in DOTT

Abb. 4.1.2. zeigt einen solchen Lösungsbaum. Die Knoten entsprechen hierbei den einzelnen Touren, ihre zugehörige Länge ist eingezeichnet.

Um nun einen Lösungsbaum eindeutig zu charakterisieren, werden in DOTT folgende Punkte gespeichert:

- Zeiger auf den Wurzelknoten des Baumes
- Zeiger auf den ersten und letzten Knoten der linearen Verkettung

Diese beschriebene zweifache Datenstruktur hat den Vorteil, daß die darauf zugreifende Komponente je nach Arbeitsweise die für sie effizientere Struktur wählen kann.

4.1.2. Experten

In der initialen Version von DOTT ist eine Vielzahl von Experten (und Spezialisten) implementiert, von denen auf einige genauer eingegangen werden soll.

Der Spezialist FI

Hierbei handelt es sich um das in Kapitel 2.2 vorgestellte Farthest Insertion, also einem Tourgenerierungsverfahren. Es stellt auch das bisher einzige Generierungsverfahren in DOTT dar. FI hat als Parameter einen positive, ganze Zahl **`startval`**, welche die Nummer der Ausgangsstadt zur Tourgenerierung ist. FI erzeugt anfangs eine Tour mit **`startval`** als Ausgangsstadt, danach wird die Stadt mit der Nummer (**`startval`** +1) modulo n als Ausgang für die nächste zu generierende Tour gewählt u.s.w. Auf diese Weise wird Tour um Tour produziert, bis die Zeit des Experten abgelaufen ist. Neben der expliziten Angabe der Startstadt läßt sich eine zufällige Auswahl durch Angabe von **`startval`** = -1 erreichen. Hierbei wird für jede Tour die Ausgangsstadt zufällig ausgewählt. Es ist offensichtlich sinnvoll, FI als ersten Experten eines Teamlaufes zu wählen, damit eine Initialmenge von Touren erzeugt wird.

Der Spezialist SORT

Einige Experten (beispielsweise LIN2) benötigen sortierte Listen von Nachbarstädten des gegebenen TSP für ihre Arbeit. Eine solche Liste für eine Stadt besteht aus den Indizes der anderen Städte, die in aufsteigender Entfernung geordnet sind. Der Spezialist SORT generiert für jede Stadt genau eine solche Liste, er trägt also nur indirekt zur Lösungsfindung bei. Durch den Parameter **`sortlength`** kann die Genauigkeit der Listen angegeben werden: er garantiert, daß auf jeden Fall die **`sortlength`** naheliegendsten Städte richtig in der Liste einsortiert sind. Bei Angabe von **`sortlength`** = 0 wird die gesamte Liste korrekt sortiert.

Der Experte LIN

Dieser Experte stellt eine Umsetzung des 3-Opt-Verfahrens nach [Lin65] dar. Es werden sukzessive die kürzesten Touren der Ausgangsmenge durch 3-Opt-Schritte verändert, bis die Zeit des Experten abgelaufen ist. Hierbei werden für eine Tour alle Möglichkeiten der aufzubrechenden Kanten betrachtet und jeweils getestet, ob ein 3-Opt-Schritt Verbesserung erzielen würde. Ist dies der Fall, so wird dieser Schritt getan und die Tour durch ihre verbesserte Version ersetzt. Es ist zu bemerken, daß LIN nur "echte" 3-Opt Schritte durchführt, d.h. solche, die sich nicht durch 2-Opt nachbilden lassen. Es werden also auf jeden Fall drei neue Kanten für einen Schritt eingefügt. Um den Aufwand für eine Tour zu verringern, läßt sich durch den Parameter **`maxnum`** festlegen, wieviele 3-Opt-Schritte maximal auf eine Lösung angewendet werden sollen. Dieses hat das Ziel, daß sich das System nicht nur auf wenige Touren (also die kürzesten) beschränkt, sondern auch eine größere Menge bearbeiten kann.

Der Experte LIN2

Dieser Experte stellt eine erweiterte Variante von LIN dar. Das Problem bei 3-Opt ist, daß die Komplexität eines Schrittes sehr hoch ist. Um den Aufwand zu reduzieren, wurde durch LIN2 eine erweiterte Variante implementiert, die es zuläßt, die Auswahl der Menge der aufzubrechenden Kanten einzuschränken. Diese Beschränkung ist durch den Parameter **depth** gegeben, mit dessen Hilfe das Verfahren die Menge der zu berücksichtigenden Kantentripel auf solche reduziert, die mit hoher Wahrscheinlichkeit eine Verbesserung der Tour ergeben. Der Parameter **maxnum** steuert bei LIN2 wiederum, wieviele 3-Opt-Schritte maximal für eine Lösung angewendet werden sollen. Für eine genaue Beschreibung des Algorithmus s. [Leo95], Abschnitt 4.5.2. Es ist zu bemerken, daß LIN2 und alle Experten die dieses erweiterte 3-Opt-Verfahren verwenden, die Listen der sortierten Nachbarstädte benötigen, daher muß der Spezialist SORT in einer vorhergehenden Arbeitsphase eingesetzt werden.

Die Experten LIN2REV, LIN2ODD und LIN2EVEN

Diese Experten entsprechen dem LIN2-Experten, mit dem Unterschied, daß sie eine andere Bearbeitungsreihenfolge der Touren haben. LIN2REV bearbeitet die längsten Touren, LIN2ODD die kürzesten mit ungeradem Index, und LIN2EVEN die kürzesten mit geradem Index. Diese Experten sind besonders für die Verteilung geeignet, da sie unabhängig voneinander auf der gleichen Lösungsmenge operieren können, ohne daß sie die gleiche Arbeit mehrfach erledigen. Es ist somit möglich, schneller eine Menge von Touren 3-optimal zu bekommen.

Die Experten ER und ERCASCADE

Diese Experten sind genetische Verfahren. ER stellt das in Kapitel 2 beschriebene Edge Recombination Crossover dar. Variationsmöglichkeiten sind gegeben bei die Auswahl der Elterntouren sowie der eigentlichen Durchführung des Crossovers. Hierbei steuern die Parameter **p** und **pcorr** die Auswahl der Elterntouren. **p** gibt die Wahrscheinlichkeit zur Auswahl der kürzesten Tour in Prozent an, für die nächstlängere Tour wird **p** mit dem Faktor (**pcorr**/100) multipliziert u.s.w. Je nach Belegung dieser beiden Parameter ergeben sich unterschiedliche Wahrscheinlichkeitsverteilungen, beispielsweise eine Begünstigung von kürzeren Touren. Der Parameter **pcorr** kann hierbei selbstverständlich auch größer als 100 sein. Sind zwei Eltern ausgewählt, so läßt sich mit den Parametern **vfirst**, **vnext** und **vnone** die eigentliche Crossoveroperation steuern. Hierbei bestimmt **vfirst** das Verfahren zur Auswahl der Startstadt, **vnext** die Auswahl der nächsten Kante (die in beiden Eltern vorhanden ist) im Verlauf des Algorithmus, sowie **vnone** das Verfahren zur Behebung von Edge Failures, d.h. die Auswahl einer Kante die nicht in beiden Eltern zu finden ist. Wichtig ist zu bemerken, daß die Menge der Start- und Arbeitslösungen bei diesem Experten getrennt verwaltet werden, d.h. Startlösungen i.a. nicht in die Arbeitslösungen übernommen werden, es sei denn, sie werden zufällig wieder erzeugt.

Der Experte ERCASCADE stellt eine Erweiterung von ER dar. Im Gegensatz zu ER läßt ERCASCADE die Konstruktion von mehreren Generationen in einer Arbeitsphase zu. Durch den Parameter **ngen** wird die maximale Größe einer Generation angegeben, und sobald diese erreicht ist, ein Generationswechsel durchgeführt. Die minimale Zeit für eine Generation wird durch die Konstante GEN_TIME statisch festgelegt, um zu bestimmen, ab welcher Zeit eine weitere Generation sinnvoll erscheint. Zusätzlich wird in ERCASCADE über den Parameter **pgen** die Auswahl der Elternlösungen in Abhängigkeit von der Generationsnummer gesteuert: nach Ablauf einer Generation wird **p** mit dem Faktor (**pgen**/100) multipliziert. **pgen** ist also ähnlich dem Parameter **pcorr**, mit dem Unterschied, daß **pgen** die Wahrscheinlichkeitsverteilung im Bezug auf die Anzahl der Generationen bestimmt, während **pcorr** sich auf die Verteilung innerhalb der aktuellen Generation bezieht.

Zusätzlich zu den vorgestellten Experten sind folgende Verfahren in DOTT implementiert: LOWER (Spezialist zur Berechnung einer maximalen unteren Schranke) und BB (Berechnung einer exakten Lösung mittels Branch-and-Bound). Der Experte MUT4 (vgl. [Leo95], Abschnitt 4.5.6) ist erweitert worden und wird in Abschnitt 4.2.2 näher beschrieben. Für eine genauere Erläuterung aller genannten Experten s. [Leo95], Kapitel 4.

4.1.3. Gutachter

Wie in Kapitel 3 erläutert wurde, haben Gutachter in Teamwork zwei Aufgaben: das Bewerten der Gesamtarbeit eines Experten mit Hilfe eines Fortschrittsmaßes, sowie das Auswählen von Lösungen, die an den Leiter übermittelt werden. In DOTT sind diese zwei Aufgaben auch in zwei getrennten Komponenten realisiert, die unabhängig voneinander eingesetzt werden können.

Zunächst sollen die Gutachter zur Auswahl von Lösungen vorgestellt werden.

REF_SHORTEST:
Dieser Gutachter berücksichtigt die Länge von Touren. Er wählt (maximal) die vom Leiter vorgegebene Anzahl der kürzesten Touren aus und übermittelt sie an diesen. Das Verfahren hat eine sehr günstige Laufzeit, da die Knoten im Lösungsbaum schon nach ihrer Länge geordnet sind, und der Gutachter somit nur sequentiell auf sie zugreifen muß. Ein weiterer Vorteil ist der, daß durch die Begünstigung der kürzesten Touren recht schnell eine Tour guter Qualität gefunden werden kann. Gleichzeitig birgt dies jedoch die Gefahr, daß lokale Extrema bevorzugt werden. Es ist nämlich denkbar, daß eine längere Tour (die nicht ausgewählt wird) ähnlicher zur optimalen Lösung ist als jede der ausgewählten. Aus diesem Grund ist in dieser Arbeit der Gutachter REF_DIVERS entwickelt worden, der (je nach Parameterwahl) zusätzlich die Unterschiedlichkeit von Lösungen berücksichtigt.

REF_ONESHORT:
Dieses Verfahren benutzt ebenfalls die Länge von Touren als Kriterium. Die Idee hierbei ist die, daß Touren mit gleicher Länge mit großer Wahrscheinlichkeit auch sehr ähnlich zueinander sind. Um nun möglichst verschiedenartige Touren auszuwählen, werden die Touren gleicher Länge (bis auf eine) gelöscht und aus dieser resultierenden Menge die kürzesten an den Leiter übermittelt. Problematisch an diesem Ansatz ist, daß natürlich nicht garantiert werden kann, daß Touren gleicher Länge wirklich ähnlich sind; eventuell werden auf diese Weise also sehr gute Touren verworfen.

REF_EDGEFAIL:
Beim Edge Recombination Crossover (Experte ER) kann es zu Edge Failures kommen. Eine Tour, bei deren Konstruktion dies oft passiert, muß offenbar sehr verschieden von ihren Eltern sein. Der Gutachter REF_EDGEFAIL wählt nun solche Touren aus, bei denen es zu mindestens einem Edge Failure gekommen ist. Wiederum ist hierbei die Idee, das Problem der lokalen Extrema in den Griff zu bekommen, indem ein möglichst breites Spektrum von Lösungen ausgewählt wird. Der Tendenz des Experten ER, nämlich im Verlauf immer ähnlichere Lösungen zu erzeugen, wird auf diese Weise entgegengewirkt. Um dennoch eine Auswahl von kurzen Touren zu garantieren, läßt sich durch den Parameter **shortest** die

Anzahl der kürzesten Touren angeben, die auf jeden Fall ausgewählt werden. REF_EDGEFAIL ist speziell für die Experten ER und ERCASCADE entwickelt und nur in Kombination mit diesen sinnvoll.

Keiner der hier vorgestellten Gutachter zur Auswahl von Lösungen erlaubt eine explizite Analyse der Beschaffenheit von Touren. Einen Ansatz, der Touren auf Grundlage der Anzahl der gemeinsamen Kanten vergleicht, ist durch den Gutachter REF_DIVERS gegeben, der später vorgestellt wird.

Im folgenden sollen die Gutachter zur Beurteilung der Gesamtarbeit von Experten vorgestellt werden. Diese sind in der initialen Version von DOTT nur sehr rudimentär vorhanden. Ihnen allen gemeinsam ist, daß sie das Fortschrittsmaß in Form einer positiven ganzen Zahl (inklusive Null) an den Leiter übergeben.

REF_NONE:
Dieser Gutachter liefert unabhängig von der Arbeit des Experten stets den Wert 0. Dies ist sinnvoll, wenn man verhindern möchte, daß dieser Experte gewinnt, oder wenn es sich um einen Spezialisten handelt, der ja nicht beurteilt werden soll.

REF_INTERACTIVE:
Hier wird die Bewertung der Gesamtarbeit des Experten vom Benutzer (im laufenden Betrieb) erfragt. Dies erlaubt zwar eine dynamische Anpassung an das Problem, ist aber für den praktischen Einsatz ungeeignet, da das System vollkommen auf den Benutzer angewiesen ist.

REF_STATIC:
Dieser "statische" Gutachter liest die Bewertung des Experten aus der Laufkontrolldatei (s. Abschnitt 4.1.4.) und übermittelt sie an den Leiter. Die Beurteilung ist hierbei implizit durch die Angabe des Siegers in der Datei gegeben: jeder Experte mit Ausnahme des Siegers erhält die Beurteilung 0, der Sieger selbst die Bewertung 1.

4.1.4. Leiter

Der Leiter bestimmt die Zusammensetzung des Teams für den nächsten Zyklus, wobei er folgende Angaben machen muß, die für das System notwendig sind:

- die Zeitdauer der nächsten Runde (in Millisekunden)
- Angabe der einzusetzenden Experten und Spezialisten
- Angabe der einzusetzenden Gutachter zur Beurteilung
- Angabe der einzusetzenden Gutachter zur Auswahl von Lösungen

Selbstverständlich werden hierbei zusätzlich die jeweiligen Parameterbelegungen für die Experten und Gutachter benötigt.

Im folgenden werden die bisher implementierten Leitertypen näher beschrieben.

L_INTERACTIVE:
In diesem Modus wird der gesamte Ablauf zu Beginn eines Zyklus interaktiv vom Benutzer erfragt, d.h. er muß angeben, welche Experten und Gutachter eingesetzt werden mit ihren zugehörigen Parametern sowie die Zeitdauer der nächsten Runde. Die Anweisungen des Benutzers werden während des Laufes in der Leiterkontrolldatei gespeichert, die dann für den

statischen Betrieb (L_STATIC) verwendet werden kann, um den Teamlauf automatisch zu wiederholen.

L_STATIC:
Dieser Leiter liest alle für ihn relevanten Informationen aus der Leiterkontrolldatei. Diese Datei enthält alle wichtigen Angaben, insbesondere auch, welcher Experte zum Sieger erklärt werden soll. Es macht bei diesem Leitertyp also keinen Unterschied, welcher Gutachter zur Bewertung der Experten in der Datei angegeben ist, es wird auf jeden Fall der vorher festgelegte zum Sieger erklärt. Hier ein Beispiel einer Leiterkontrolldatei (kurz: LC-Datei für Leader-Control-Datei). Das Format dieser Datei ist seit der initialen Version von DOTT erheblich verbessert worden, für eine ausführliche Beschreibung dieser komfortablen, weitgehend selbsterklärenden Variante siehe [Leo95-8].

```
##Leader Control

# ------ 1. Runde ------

time = 10000                # Zeitdauer: 10000ms

# Daten für Rechner Nr. 1
exp = FI (0,0)              # Experte Farthest Insertion mit
                            # Parametern
winref = STATIC             # Siegerwahl aus Datei lesen
solref = SHORTEST           # kürzeste Lösungen auswählen

#Daten für Rechner Nr. 2
exp = SORT(0)               # Spezialist zum Konstruieren der
                            # Nachbarschaftsliste
winref = STATIC             # Siegerwahl aus Datei lesen
solref = SHORTEST           # wird ignoriert, da Spezialist

winner = 0                  # Experte FI gewinnt

$                           # Abschluß dieser Runde

# ------ 2. Runde ------

time = 130000               # 130 Sekunden

# Daten für Rechner 1
exp = LIN2 (10, 10)         # Experte erweitertes 3-Opt
winref = STATIC             # statische Siegerwahl
solref = SHORTEST           # kürzeste Lösungen auswählen

# Daten für Rechner 2
                            # Edge Recombination Crossover
exp = ER (10, 11, 21, 31, 50, 100)
winref = STATIC             # Sieger aus Datei lesen
solref = ONESHORT           # Nur eine Lösung einer Länge

winner = 1                  # ER Crossover gewinnt

$                           # Ende der 2. Runde
```

Abschließend sei erwähnt, daß DOTT einen sog. Batch-Betrieb kennt, d.h. einen Betriebsmodus, bei dem die Ausgaben des Systems in Dateien protokolliert werden.

L_STATIC1:
Dieser Leiter ist eine Erweiterung des statischen Leiters, der sog. semi-dynamischen Leiter. Im Gegensatz zu L_STATIC kann hier durch Angabe von Gutachtern zur Beurteilung der Experten der Sieger dynamisch (zur Laufzeit) gewählt werden. Zusätzlich enthält L_STATIC1 eine eingebaute Methode zur Siegerwahl, die den Zweck hat, möglichst viele der erzeugten Lösungen zu übernehmen. Dieser integrierte Beurteilungs-Gutachter läuft folgendermaßen ab. Jedem Experten wird einer der unten angegebenen Fortschrittswerte zugeordnet, der jeweils von den beschriebenen Voraussetzungen abhängig ist:

Bewertung 3 :
- Die Anzahl der erzeugten Touren ist größer als der Parameter **nlost**[1]
- Der Experte ist als Sieger in der LC-Datei angegeben

Bewertung 2 :
- Die Anzahl der erzeugten Touren ist größer als der Parameter **nlost**
- Der Experte ist nicht als Sieger in der LC-Datei angegeben

Bewertung 1 :
- Die Anzahl der erzeugten Touren ist kleiner gleich dem Parameter **nlost**
- Der Experte ist als Sieger in der LC-Datei angegeben

Bewertung 0 :
- Die Anzahl der erzeugten Touren ist kleiner gleich dem Parameter **nlost**
- Der Experte ist nicht als Sieger in der LC-Datei angegeben

Hierbei gewinnt (wie üblich) der Experte mit der höchsten Bewertung, die maximal 3 betragen kann. Durch diese Art der Beurteilung wird verhindert, daß eine große Menge von Ergebnissen verworfen werden, wenn der in der Datei als Sieger angegebene Experte nur wenige Lösungen produziert hat.
Die vorhandenen Leitertypen sind noch sehr stark vom Benutzer bzw. der LC-Datei gesteuert, abgesehen von der automatischen Siegerwahl durch den Leiter L_STATIC1. Es ist an dieser Stelle offenbar noch viel Entwicklungsarbeit notwendig, es wäre wünschenswert, einen vollautomatischen Leiter zu haben, bei dem der Benutzer nur noch wenige, konfigurationsbedingte Informationen angeben muß. Ideen für die Umsetzung eines solchen dynamischen Leiters werden in Kapitel 6 gegeben.

[1] **nlost** ist die Anzahl der zu übermittelnden Lösungen der Verlierer

4.2. Neue Experten

In diesem Abschnitt wird die Weiterentwicklung des Systems DOTT im Bezug auf die Experten beschrieben. Die neuen Experten gehören zum größten Teil der Gruppe der (mit anderen kombinierten) genetischen Verfahren an. Die Idee hierzu ist in Abschnitt 2.4.4 bereits beschrieben. Umgesetzt wurde dieser Ansatz in der ursprünglichen Version von DOTT durch den Experten MUT4, der nun nachträglich erweitert und in MUT4_LO3 umbenannt wurde. Allen Experten in DOTT gemeinsam (mit den Ausnahmen der Spezialisten SORT, LOWER und BB) ist der Parameter **nlost**. Dieser legt fest, wieviele Lösungen im Falle des Nicht-Siegens von diesem Experten ausgewählt werden sollen.

4.2.1. OPT2

Parameter: **nlost**, **maxnum**, **depth**

Dieser Experte ist eine Umsetzung des (erweiterten) 2-Opt-Verfahrens. Wie bereits in Abschnitt 2.3.1. erläutert, wird für einen 2-Opt-Schritt eine existierende Tour ausgewählt, und durch Aufbrechen von zwei Kanten in zwei Teiltouren zerlegt. Es gibt genau eine Möglichkeit, diese beiden Teiltouren zu einer vollständigen, von der ursprünglichen verschiedenen Tour zusammenzusetzen. Dieses Zusammenfügen entspricht dem Umdrehen einer Teiltour in der ursprünglichen Tour. Wie schon bei der erweiterten Fassung von 3-Opt (Experte LIN2), so ist auch hier eine zweifache Einschränkung ohne wesentlichen Qualitätsverlust mit Hilfe der Parameter **maxnum** und **depth** möglich. Hierbei beschränkt **maxnum** die maximale Anzahl der 2-Opt-Schritte auf einer Tour, während **depth** die Menge der aufzubrechenden Kantentupel auf solche reduziert, die mit entsprechend hoher Wahrscheinlichkeit zu einer Verbesserung der Tour führen. Der Experte OPT2 benötigt den Einsatz des Spezialisten SORT in einer vorhergehenden Arbeitsphase, da er die sortierte Listen der Nachbarstädte des TSP benutzt. Diese Bedingung gilt gleichermaßen auch für die neuen Experten MUT4_LO3, MUT4_LO2, ER_LO3 und ER_LO2. Für den genauen Algorithmus von OPT2 s. [Leo95-8], Abschnitt 1.1.
Alle weiteren im folgenden beschriebenen Experten gehören der Gruppe der kombinierten Verfahren an; es wird jeweils ein (genetischer) Mutations- bzw. Crossoverschritt auf ein bzw. zwei Touren angewendet, und anschließend die resultierende Tour mit Hilfe eines anderen Verfahrens verbessert. Man spricht bei dieser Verbesserung auch von lokaler Optimierung.

4.2.2. MUT4_LO3

Parameter: **nlost**, **maxnum**, **depth**, **p**, **pcorr**, **gentime**

Dieser Experte ist ein kombiniertes Verfahren aus 4-Opt-Mutation und 3-Opt-Verbesserung. In der ursprünglichen Version von DOTT stellte dieser Experte den einzigen seiner Art dar. Da nun weitere dieser kombinierten Experten zur Verfügung stehen, wurde MUT4 in MUT4_LO3 umgenannt, um die verwendete lokale Optimierung zu verdeutlichen. Der Experte arbeitet nun folgendermaßen. Er wählt zufällig eine Tour aus der Menge der Ausgangslösungen, wendet auf diese einen speziellen (s.u.) 4-Opt-Schritt an, und verbessert anschließend die resultierende Tour mit Hilfe des erweiterten 3-Opt-Verfahrens. Es folgt die graphische Darstellung der Arbeitsweise.

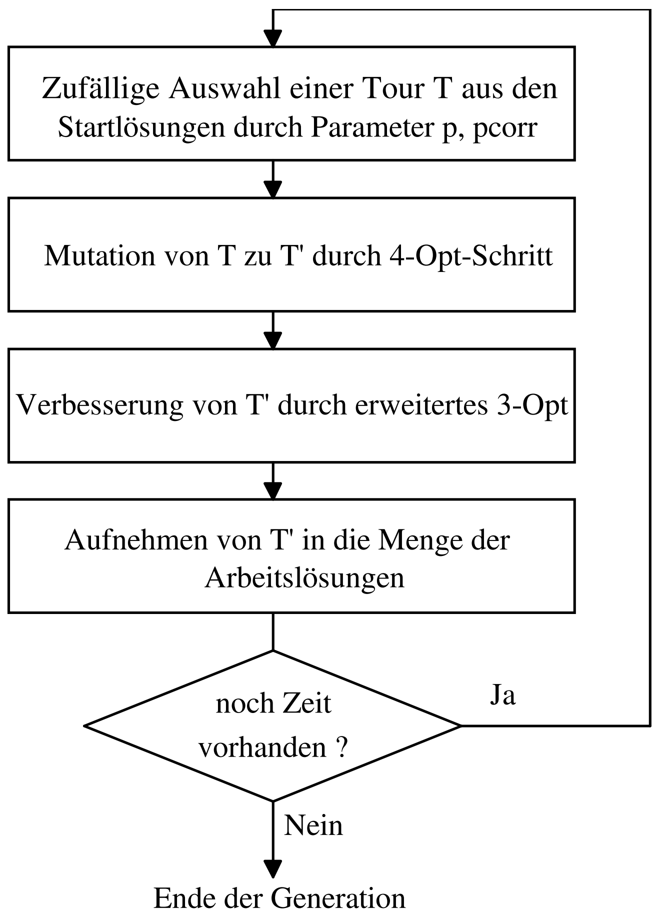

Abb. 4.2.1: allgemeiner Ablauf des Experten MUT4_LO3

4-Opt ist eigentlich ein Tourverbesserungsverfahren, d.h. eine Tour muß nach einem 4-Opt-Schritt kürzer sein als die ursprüngliche. Der in MUT4_LO3 (und MUT4_LO2) verwendete Schritt (Abb. 4.2.2) ist immer der gleiche und garantiert diese Zusage nicht, er wird hier lediglich für eine Mutation verwendet, bei der die Tour also auch länger werden kann. Es geht hierbei nur um eine künstliche Verfremdung, die nachfolgende lokale Optimierung soll dann dafür sorgen, daß die resultierende Tour tatsächlich eine gute Qualität bekommt.

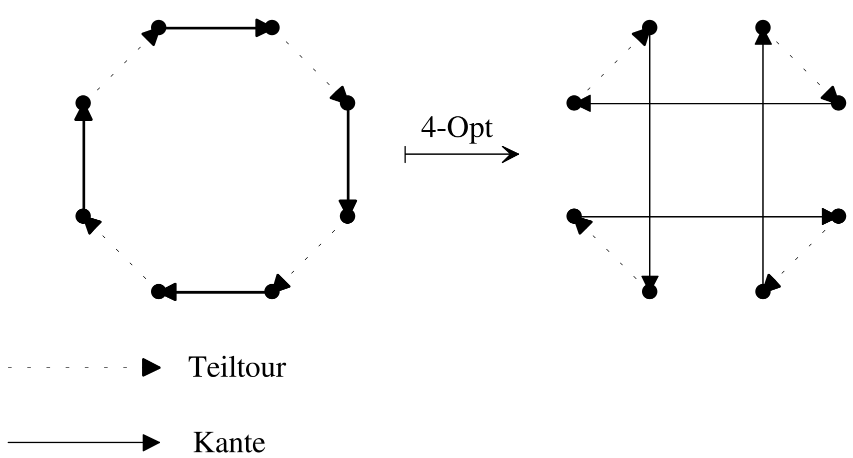

Abb. 4.2.2. Verwendeter 4-Opt-Schritt für MUT4_LO3 und MUT4_LO2

Die in MUT4_LO3 gegebenen Parameter sollen nun erläutert werden. Was die Auswahl der Ausgangstour betrifft, so wird sie durch die Parameter **p** und **pcorr** gesteuert. Analog zu den Experten ER und ERCASCADE ist mit **p** die Wahrscheinlichkeit der Auswahl der kürzesten Tour gegeben. Wird diese nicht ausgewählt, so beträgt die Wahrscheinlichkeit der nächstlängeren Tour **p**·(**pcorr/100**) u.s.w., solange bis eine Tour ausgewählt ist. Für den anschließenden 4-Opt-Schritt werden die vier aufzubrechenden Kanten zufällig ohne Einflußmöglichkeit des Benutzers gewählt. Es soll noch einmal erwähnt werden, daß die 4-Opt-Mutation nach der Wahl der aufzubrechenden Kanten eindeutig ist, nämlich genau wie in Abb. 4.2.2 dargestellt. Die Optimierung der resultierenden Tour mit Hilfe des erweiterten 3-Opt-Verfahrens kann durch die üblichen Parameter **depth** und **maxnum** (s. Beschreibung von LIN2) gesteuert werden. Wichtig zu erwähnen ist, daß MUT4_LO3 (und ebenso MUT4_LO2, ER_LO3 und ER_LO2) die Menge der Start- und die der Arbeitsresultate getrennt verwaltet. Es werden also bei der Erzeugung einer neuen Generation nur Touren der Startlösungen für die Auswahl verwendet; erzeugte Touren können nicht in der gleichen Generation weiter bearbeitet werden. In der ursprünglichen Fassung von MUT4_LO3 war demnach die Generationszeit implizit durch die Zykluszeit gegeben. Dies erwies sich als problematisch, da der Experte auf ein Teamtreffen angewiesen war, um eine neue Generation zu bilden und somit weiterzukommen. Insbesondere führte dies dazu, daß dieser Experte im Team schlechter abschnitt als im sequentiellen Fall. Aus diesem Grund wurde MUT4_LO3 um das Generationenkonzept erweitert, so daß innerhalb eines Zyklus mehrere Generationen gebildet werden können. Der Parameter **gentime** legt hierbei die Zeit einer Generation fest; MUT4_LO3 erzeugt dann die maximal mögliche Anzahl von Generationen während dieser Zeit in einem Zyklus. Ist die Generationszeit abgelaufen, so wird der ursprüngliche Lösungsbaum gelöscht und durch den neu erzeugten ersetzt, auf dessen Grundlage dann die nächste Generation gebildet werden kann. Um sicherzustellen, daß die beste Tour des ursprünglichen Lösungsbaumes nicht verloren geht, wird diese in jedem Fall beim Generationswechsel in den neuen Lösungsbaum übernommen. Dies ist wichtig für den Fall, daß keine gleichwertige oder kürzere Lösung als die bisher kürzeste produziert wurde; es wird hierdurch gewährleistet, daß das System keine Rückschritte macht. Mit der Erweiterung um das Generationenkonzept ist nun eine bessere Modularität und Verknüpfung der genetischen Experten mit anderen möglich. Die ursprüngliche Arbeitsweise von MUT4_LO3 läßt sich im übrigen vollständig nachbilden, indem man die Generationszeit der Zykluszeit angleicht; in diesem Fall wird genau eine Generation produziert. Um den Unterschied zum Experten ERCASCADE herauszustellen, sei erwähnt, daß bei diesem der Generationswechsel nicht von der Zeit, sondern von der Größe der erzeugten Generation abhängt: Sobald die Anzahl der Individuen eine vorgegebene Grenze erreicht hat, wird ein Wechsel durchgeführt.

4.2.3. MUT4_LO2

Parameter: **nlost**, **maxnum**, **depth**, **p**, **pcorr**, **gentime**

Der beschriebene Experte MUT4_LO3 liefert zwar qualitativ gute Ergebnisse, hat jedoch einen entscheidenden Nachteil: seine hohe Laufzeit. Insbesondere steht der Aufwand der Mutation in keinen Verhältnis zu dem der anschließenden lokalen Optimierung. Aus diesem Grund wurde ein weiterer Experte entwickelt, der statt 3-Opt das erweiterte 2-Opt-Verfahren verwendet. Analog zu MUT4_LO3 wird bei diesem Experten die in Abb. 4.2.2 dargestellte 4-Opt-Mutation auf eine Tour angewendet und diese anschließend mit Hilfe von 2-Opt verbessert. Da das hier verwendete erweiterte 2-Opt-Verfahren mit $O(n)$ eine wesentlich günstigere Komplexität hat als 3-Opt mit $O(n^3)$, liegt der entscheidende Vorteil von

MUT4_LO2 in seiner Geschwindigkeit, es werden wesentlich mehr Ergebnisse produziert als bei MUT4_LO3. Man erhofft sich nun, daß mit diesem Verfahren schneller gute Ergebnisse gefunden werden, obwohl die eigentliche Optimierung nicht so leistungsfähig ist wie 3-Opt. Die Mächtigkeit von MUT4_LO2 zeigt sich in der Praxis, es werden in der gleichen Zeit oft bessere Touren gefunden als von MUT4_LO3 (vgl. Kapitel 5).

4.2.4. ER_LO3

Parameter: **`nlost`**, **`maxnum`**, **`depth`**, **`vfirst`**, **`vnext`**, **`vnone`**, **`p`**, **`pcorr`**, **`gentime`**

Die Qualität der Lösungen von ER und ERCASCADE hat sich als eher mäßig erwiesen. Insbesondere tritt in ERCASCADE das Problem auf, daß im Laufe der Generationen sich die erzeugten Touren immer ähnlicher werden. Aus diesem Grund wurden ER_LO3 und ER_LO2 entwickelt, die ebenfalls der Gruppe der kombinierten (genetischen) Verfahren angehören. ER_LO3 wählt zwei Touren zufällig als Eltern aus, wendet auf sie einen Edge Recombination Crossover-Schritt an, und verbessert die Tour anschließend mit Hilfe des erweiterten 3-Opt-Verfahrens. Der einzige Unterschied zu MUT4_LO3 besteht demnach darin, daß statt der 4-Opt-Mutation auf einer Tour Edge Recombination Crossover auf zwei Touren angewendet wird. Parameter dieses Experten sind im wesentlichen die gleichen wie bei dem Experten ER (vgl. Beschreibung von ER), also **`vfirst`**, **`vnext`**, **`vnone`**, **`p`** und **`pcorr`**. Zusätzlich sind die Parameter des erweiterten 3-Opt-Verfahrens **`maxnum`** und **`depth`** vorhanden. Die Bemerkungen zu MUT4_LO3 gelten in angepaßter Form selbstverständlich auch für ER_LO3; Wiederum tritt der Effekt auf, daß das lokale Optimieren mit dem erweiterten 3-Opt- Verfahren wesentlich aufwendiger ist als der eigentliche Crossover-Schritt, aus diesem Grund wurde zusätzlich ER_LO2 analog zu MUT4_LO2 entwickelt.

4.2.5. ER_LO2

Parameter: **`nlost`**, **`maxnum`**, **`depth`**, **`vfirst`**, **`vnext`**, **`vnone`**, **`p`**, **`pcorr`**, **`gentime`**

Dieser Experte entspricht weitestgehend ER_LO3, jedoch wird hier statt 3-Opt das erweiterte 2-Opt-Verfahren zur lokalen Optimierung eingesetzt. Die Bemerkungen zu MUT4_LO2 gelten in angepaßter Form ebenfalls für ER_LO2.

Offensichtlich hängt das Laufzeitverhalten der vier neuen Experten in erster Linie von dem eingesetzten Verfahren zur lokalen Optimierung ab. MUT4_LO3 und ER_LO3 produzieren etwa genauso viele Lösungen in der gleichen Zeit, analog gilt dies für MUT4_LO2 mit ER_LO2. Aus diesem Grund eignen sich diese Experten hervorragend zur Zusammenarbeit im Team. In den im nächsten Kapitel dargestellten Testläufen wurde auch in erster Linie auf die Leistung dieser beiden Teams gesetzt. Es zeigt sich in der Praxis, daß mit den vier neuen zusammengesetzten Experten sehr gute Ergebnisse erzielt werden, wesentlich bessere als mit den "reinen" genetischen Verfahren. Der Vorteil ist der, daß auch Touren ausgewählt werden, die ohne eine lokale Optimierung verworfen werden. Durch die Einführung des Generationskonzeptes ist darüberhinaus ein sehr flexibler Einsatz dieser Experten in Teamwork möglich. So lassen sich die Experten MUT4_LO2 und ER_LO2 etwa mit dem erweiterten 3-Opt-Verfahren LIN2 kombinieren, indem dieses in periodischen Abständen die aktuelle Population komplett optimiert.

4.3. Neue Gutachter

Wie bereits in der Einleitung erwähnt, sind in der initialen Version von DOTT Gutachter nur recht rudimentär vorhanden. Aus diesem Grund wurden im Rahmen dieser Arbeit je ein Gutachter für die zwei vorgegebenen Aufgaben (Gesamtbewertung und Auswahl von Resultaten) entwickelt. REF_LENGTH ist ein Gutachter zur Bewertung der Gesamtarbeit von Experten auf Grundlage der Anzahl der erzeugten Touren sowie ihrer Länge. REF_DIVERS wählt Lösungen (je nach Parametergewichtung) sowohl nach Länge als auch nach Unterschiedlichkeit aus.

4.3.1. REF_LENGTH

Parameter: **`anz`**, **`p1`**, **`p2`**, **`p3`**, **`p4`**, **`p5`**, **`p6`**

Dieser Gutachter bewertet die Gesamtarbeit von Experten in Form einer positiven, ganzen Zahl, die anschließend an den Leiter übermittelt wird. Dieser bestimmt dann aus allen Beurteilungen der Gutachter den neuen Sieger, von dem der gesamte Lösungsbaum für den weiteren Teamlauf übernommen wird. Die Idee dieses Gutachters ist die, daß derjenige Experte gewinnen soll, der für den aktuellen Zeitpunkt des Lösungsprozesses die besten Ergebnisse produziert hat. Grundlage für die Berechnung des Fortschrittsmaßes ist hierbei sowohl die Länge der einzelnen produzierten Lösungen (Qualität) als auch ihre Anzahl (Quantität). Selbstverständlich benötigt REF_LENGTH hierfür den vollständigen Baum der Arbeitslösungen des Experten. Durch Parameterbelegung läßt sich auswählen, welche Touren im speziellen hierbei berücksichtigt werden sollen. Im folgenden wird eine Übersicht über diese Parameter gegeben.

- **`anz`** die Anzahl der (maximal) relevanten Touren für die Parameter **`p3`**, **`p4`** und **`p5`**
- **`p1`** je größer **`p1`**, desto stärker wird die Länge der kürzesten produzierten Tour berücksichtigt
- **`p2`** je größer **`p2`**, desto stärker wird die Durchschnittslänge aller produzierten Lösungen berücksichtigt
- **`p3`** je größer **`p3`**, desto stärker wird die Durchschnittslänge der **`anz`** kürzesten generierten Lösungen gewichtet
- **`p4`** je größer **`p4`**, desto stärker wird die Durchschnittslänge der **`anz`** längsten produzierten Touren berücksichtigt
- **`p5`** je größer **`p5`**, desto stärker wird die Länge der **`anz`** unähnlichsten (von der kürzesten) generierten Touren gewichtet
- **`p6`** je größer **`p6`**, desto mehr Wert wird auf die Anzahl der produzierten Lösungen gelegt (Quantität)

`p1`, ..., **`p6`** sind hierbei natürliche Zahlen (inklusive Null), während **`anz`** eine ganze Zahl größer Null verlangt. Mit Hilfe dieser Parameter wird für die Arbeitslösungen eine gewichtete Summe, das Fortschrittsmaß, errechnet, wobei zusätzlich ein Normierungsfaktor ins Spiel kommt.

Die Berechnungsformel lautet folgendermaßen:

Fortschrittsmaß = **p1** · (Normierung / Länge der kürzesten Lösung) +
p2 · (Normierung / Durchschnittslänge aller Lösungen) +
p3 · (Normierung / Durchschnittslänge der **anz** kürzesten Lösungen) +
p4 · (Normierung / Durchschnittslänge der **anz** längsten Lösungen) +
p5 · (Normierung / Durchschnittslänge der **anz** von der kürzesten Tour unähnlichsten Lösungen) +
p6 · (Anzahl der produzierten Lösungen / Anzahl der Städte des TSP)

Der Normierungsfaktor ist hierbei die Länge der kürzesten Tour der Startlösungen bzw. die Anzahl der Städte des TSP, falls diese Ausgangsmenge leer ist (bei Tourgenerierungsverfahren). **p1** bis **p5** bestimmen offensichtlich die Relevanz der Qualität der erzeugten Touren, während **p6** die Quantität berücksichtigt. Das errechnete Fortschrittsmaß ist eine natürliche Zahl und in dem Sinne zu deuten, daß, je größer dieser Wert ist, desto besser ist auch die Arbeit des Experten. Dies gilt selbstverständlich analog für die sechs Terme der Summe (die im folgenden durch $Term_1$, ..., $Term_6$ bezeichnet werden) im Bezug auf ihre jeweiligen Kriterien. Um die Durchschnittslänge der in $Term_5$ untersuchten Touren zu erhalten, ist es notwendig, herauszufinden, um welche Lösungen es sich dabei handelt. Dies geschieht durch Ordnen der Touren nach ihrer Unähnlichkeit bezüglich der kürzesten Tour und anschließender Auswahl (s. Algorithmus). Es folgt eine Abschätzung der sechs Summenterme. Kürzt man die Parameter **pi** (**i**=1,..., 6) aus den Termen heraus, so liegen die resultierenden $Term_1'$, ..., $Term_6'$ in folgendem Wertebereich:

$0 < Term_1' < \infty$, es gilt i.a. jedoch $1 \leq Term_1' < 1,1$
$0 < Term_2' < \infty$, es gilt i.a. jedoch $0 < Term_2 < 1$
$0 < Term_3' < \infty$, es gilt i.a. jedoch $0 < Term_3' < 1$
$0 < Term_4' < \infty$, es gilt i.a. jedoch $0 < Term_4' < 1$
$0 < Term_5' < \infty$, es gilt i.a. jedoch $0 < Term_5' < 1$
$0 \leq Term_6' < \infty$, es gilt i.a. jedoch $0 \leq Term_6' < 2$

Durch die Parameter **p1**,..., **p6** ergibt sich eine Streckung dieser Werte, so daß es sich anbietet, Parameterwerte im Prozent- bzw. Promillbereich (je nach Genauigkeit) zu verwenden.
Es soll nun der eigentliche Ablauf des Gutachters vorgestellt werden:

1.) Überprüfung, ob überhaupt Ergebnisse produziert wurden. Falls nein, so wird die Beurteilung 0 an den Leiter zurückgegeben.
2.) Falls der Parameter **anz** (Anzahl der relevanten Touren für $Term_3$, $Term_4$ und $Term_5$) größer ist als die Anzahl der generierten Lösungen, so wird **anz** auf diesen Wert reduziert.
3.) Es werden die Unterschiedlichkeitswerte im Bezug auf die kürzeste erzeugte Lösung berechnet (dies wird von $Term_5$ benutzt):
    a) Anlegen eines temporären Arrays im Speicher, dessen Einträge die Unterschiedlichkeitswerte kombiniert mit den Verweisen auf die entsprechenden Touren speichern.
    b) Berechnung der eigentlichen Werte der Unterschiedlichkeit und Speichern im Array. Hierbei wird die zu vergleichende Tour sequentiell durchlaufen und für jede ihrer Kanten geprüft, ob diese in gleicher oder umgedrehter Form in der kürzesten Lösung vorhanden ist. Ist dies nicht der Fall, so wird die Anzahl der

gefundenen, nicht gemeinsamen Kanten um Eins erhöht. Diese Prozedur wird für jede der erzeugten Touren durchlaufen, es ergeben sich die gewünschten Unterschiedlichkeitswerte, die im Array gespeichert werden.

c) Das Array wird mit dem Quicksort-Verfahren im Bezug auf die errechneten Werte sortiert.

4.) Das Fortschrittsmaß wird gemäß oben angegebener Formel unter Zuhilfenahme der Werte des Arrays ermittelt.

5.) Das errechnete Maß wird an den Leiter übermittelt.

6.) Der Speicher für das temporäre Array wird freigegeben.

Mit diesem Gutachter hat man erstmals in DOTT eine Möglichkeit, die Arbeit von Experten explizit zu untersuchen und zu bewerten. Man erhofft sich hiervon eine Stärkung der für das aktuelle Problem gut geeigneten Experten sowie eine Anpassung auf die aktuelle Situation im Lösungsprozeß. Mit Hilfe der Testläufe im nächsten Kapitel soll versucht werden, hierzu geeignete Parameterbelegungen zu finden, so daß dieses Ziel tatsächlich erreicht wird.

4.3.2. REF_DIVERS

Parameter: **`p_diversity`**, **`p_length`**

Dieser Gutachter wählt eine Teilmenge der produzierten Lösungen von Experten aus, um sie an den Leiter zu übermitteln. REF_DIVERS benötigt hierfür den gesamten generierten Lösungsbaum, um die Arbeit des Experten zu analysieren. Grundlage für die Auswahl bildet zum einen die Länge der Touren, und zum anderen ihre Unterschiedlichkeit bezüglich der kürzesten Lösung (vgl. Gutachter REF_LENGTH). Auf dem Gebiet des automatischen Theorembeweisens sind gute Erfahrungen mit dem Orientieren an dem Ziel (welches man zu beweisen versucht) gemacht worden: je ähnlicher ein Term diesem Ziel ist, desto wahrscheinlicher wird er zur Weiterbearbeitung ausgewählt. Bei der hier gegebenen Aufgabe, dem TSP, ist ein solches Ziel zwar nicht gegeben (denn man kennt bei der Berechnung die optimale Tour natürlich noch nicht); dennoch läßt sich diese Idee übertragen, indem man sich an der kürzesten bisher gefundenen Tour ausrichtet. Je nach Parametergewichtung ist es nun für die Selektion in REF_DIVERS möglich, stärker kürzere Touren oder solche die besonders verschieden von der kürzesten sind zu berücksichtigen. Insbesondere läßt sich der Gutachter REF_SHORTEST vollständig nachzubilden, indem ausschließlich Gewicht auf die Länge von Touren gelegt wird. REF_DIVERS erlaubt also den expliziten Vergleich von Lösungen und bei Bedarf auch eine Selektion von solchen, die besonders unterschiedlich von der kürzesten Lösung sind. Dieser Ansatz (der Unterschiedlichkeit) wird mit Hilfe dieses Gutachters in DOTT erstmalig verwirklicht. Entwickelt wurde REF_DIVERS, um dem Problem der lokalen Extrema entgegenzuwirken. Um einem lokalen Optimum zu entweichen, kann es nötig sein, auch längere Touren auszuwählen, falls diese das Potential in sich tragen, zur optimalen Lösung verbessert werden zu können. Durch Bevorzugung von stark unterschiedlichen Touren möchte man hierbei erreichen, daß ein möglichst breites Spektrum von Lösungen ausgewählt wird. Problematisch an diesem Ansatz ist noch, daß zwar tatsächlich Touren ausgewählt werden, die sehr verschieden von der kürzesten sind; diese Touren können sich untereinander jedoch wiederum sehr ähneln. In der hier implementierten Variante hat man auf der anderen Seite den Vorteil einer günstigen Laufzeit, da nicht alle Touren miteinander verglichen werden müssen. Erfahrungen mit diesem neuen Gutachter sind im nächsten Kapitel nachzulesen.

Kommen wir nun zu den Parametern von REF_DIVERS. Es existieren genau zwei Parameter, die im folgenden erläutert werden.

- **`p_length`** je größer **`p_length`**, desto eher werden kürzere Touren ausgewählt
- **`p_diversity`** je größer **`p_diversity`**, desto eher werden unterschiedlichere Touren (bzgl. der kürzesten) ausgewählt

Der zulässige Wertebereich dieser Parameter ist der der natürlichen Zahlen (inklusive Null). REF_DIVERS arbeitet nun folgendermaßen. Jeder erzeugten Tour wird ein Relevanzwert zugeordnet, der nach der unten angegebenen Berechnungsfunktion bestimmt wird. Die Werte aller Touren bilden dann das Entscheidungskriterium, welche Touren auszuwählen sind (nämlich diejenigen mit den höchsten Relevanzwerten). Ausnahme hiervon ist lediglich die kürzeste Tour; sie wird in jedem Fall an den Leiter übermittelt. Hier also die Berechnungsvorschrift für eine Tour:

Relevanz = **`p_length`** · (Länge der kürzesten Tour / Länge dieser Tour) +
`p_diversity` · (Anzahl der nicht gemeinsamen Kanten mit kürzester Tour) / Anzahl der Städte des TSP)

Die Länge der kürzesten Tour sowie die Anzahl der Städte des TSP dienen hierbei der Normierung der beiden Terme, die analog zu REF_LENGTH ab jetzt mit $Term_1$ und $Term_2$ bezeichnet werden. Die Berechnung der Anzahl der nicht gemeinsamen Kanten zweier Touren ist identisch zu dem Verfahren das für REF_LENGTH verwendet wird. Betrachten wir nun eine Abschätzung des Wertebereichs von $Term_1$ und $Term_2$. Seien $Term_1'$, $Term_2'$ die um **`p_length`** bzw. **`p_diversity`** gekürzten Terme. Dann gilt

$0 < Term_1' \leq 1$
$0 \leq Term_2' \leq 1$

Hat $Term_1'$ den Wert 1, so ist die Länge der betrachteten Tour genau gleich der Länge der kürzesten Tour; je näher der Wert jedoch an der Null liegt, desto länger ist die untersuchte Tour im Vergleich zur kürzesten. Was $Term_2'$ betrifft, so ist die Anzahl der nicht gemeinsamen Kanten der beiden Touren minimal Null und maximal gleich der Städteanzahl des TSP. $Term_2'$ hat daher den Wert 0, falls es sich genau um die kürzeste Tour (oder ihre umgekehrte Variante) handelt, und den Wert 1, falls sie gänzlich verschieden von ihr ist. Es bietet sich offenbar an, für die Parameter **`p_length`** und **`p_diversity`** Prozentzahlen, also Werte im Bereich von 0 bis 100 zu verwenden, durch die eine entsprechende Streckung der Werte von $Term_1'$ und $Term_2'$ geschieht. Je höher ein Relevanzwert einer Tour ist, desto wahrscheinlicher wird diese zur Übermittlung an den Leiter ausgewählt.
Schließlich sei noch eine genauere Beschreibung der Arbeitsweise des Gutachters gegeben:

1.) Überprüfung, ob überhaupt Ergebnisse produziert wurden.
2.) Falls die Anzahl der erzeugten Lösungen kleiner als die Anzahl der zu übermittelnden Lösungen ist, so wird der gesamte Lösungsbaum ausgewählt.
3.) Berechnung eines Relevanzwertes für jede der produzierten Touren:
    a) Anlegen eines temporären Arrays, in welchem die Relevanzwerte sowie die Verweise auf die zugehörigen Touren gespeichert werden.
    b) Berechnung der Unterschiedlichkeit zur kürzesten Lösung für alle Touren. Das verwendete Verfahren hierzu ist gleich dem in REF_LENGTH.

c) Bestimmung der eigentlichen Relevanzwerte nach obiger Formel sowie Speicherung dieser Werte in der Matrix für alle Lösungen.
d) Sortieren des Arrays mit Hilfe der Quicksort-Methode bezüglich der Relevanzwerte.

4.) Es wird auf jeden Fall die kürzeste generierte Lösung an den Leiter übermittelt. Zusätzlich wird die (übrige) vorgeschriebene Anzahl von Lösungen mit den höchsten Relevanzwerten an den Leiter geschickt.
5.) Freigeben des Speichers für das Array.

Im folgenden Kapitel werden die beiden neuen Gutachter auf ihre Praxistauglichkeit getestet. Es ist zu erwarten, daß mit ihrer Hilfe tatsächlich bessere Ergebnisse gefunden werden als mit den bisherigen Gutachtern. Entscheidend hierbei ist das Finden von geeigneten Parametereinstellungen; Wird beispielsweise bei REF_DIVERS ausschließlich Wert auf die Unterschiedlichkeit gelegt, so wird man eventuell schlechtere Ergebnisse als beim Einsatz von REF_SHORTEST erhalten. Eine geeignete Kombination beider Kriterien, nämlich der Unterschiedlichkeit sowie der Länge, sollte jedoch zum gewünschten Erfolg führen. Durch die Entwicklung von REF_LENGTH und REF_DIVERS ist das System einer Automatisierung wieder einen Schritt näher gekommen. Die Bestimmung des Siegers kann jetzt unabhängig vom Eingreifen des Benutzers mit Hilfe von REF_LENGTH geschehen. Zusätzlich bietet DOTT nun durch den Gutachter REF_DIVERS die Möglichkeit, dem Problem der lokalen Extrema beizukommen. Von einer benutzerunabhängigen Steuerung des Systems kann in DOTT dennoch bisher nicht die Rede sein, da ein vollautomatischer Leiter noch nicht implementiert ist, der selbständig die Aufgabe analysiert und geeignete Teams von Experten und Gutachtern zusammenstellt.

4.4. Übersicht der Komponenten

Es soll abschließend eine Übersicht über sämtliche Experten, Gutachter sowie Leiter mit ihren jeweiligen Parametern in Kurzform gegeben werden. Für eine genauere Beschreibung sei auf die entsprechenden Kapitel sowie auf [Leo95] und [Leo95-8] verwiesen.

Gutachter:

Nr.	Name	Parameterliste
0	NONE (kein Gutachter gewählt)	
1	INTERACTIVE Beurteilung: wird vom Benutzer erfragt	
2	STATIC Beurteilung: wird aus LC-Datei gelesen	
3	LENGTH Beurteilung: berücksichtigt Länge sowie Anzahl (**p6**) der produzierten Lösungen	**anz**, **p1**, **p2**, **p3**, **p4**, **p5**, **p6**
11	SHORTEST Auswahl: es werden die kürzesten Lösungen übermittelt	
12	EDGEFAIL Auswahl: es werden Lösungen mit Edge Failures ausgewählt sowie eine feste Anzahl der kürzesten (**shortest**)	**shortest**
13	ONESHORT Auswahl: die kürzesten Lösungen werden gewählt, jedoch von jeder Länge nur eine	
14	DIVERS Auswahl: es werden je nach Gewichtung eher kürzere (**p_length**) oder unterschiedlichere (**p_diversity**) Lösungen gewählt	**p_diversity**, **p_length**

Leiter:

Nr.	Name
0	NONE (keine Angabe. Standardmäßig wird INTERACTIVE gewählt)
1	INTERACTIVE Interaktiv gesteuerte Team-Leitung durch den Benutzer
2	STATIC Statische Teamleitung, die aus Leiterkontrolldatei gelesen wird
3	STATIC1 Semi-Dynamische Teamleitung, die durch die Leiterkontrolldatei bestimmt wird.Der Sieger wird dynamisch zur Laufzeit bestimmt.

Experten:

Nr.	Name Erläuterung	Parameterliste
0	NONE (kein Experte gewählt)	
1	FI Spezialist: Farthest Insertion	**`nlost`**, **`startval`**
2	SORT Spezialist: Generierung der sortierten Listen der Nachbarstädte	**`sortlength`**
3	LOWER Spezialist: Berechnung einer unteren Schranke	
4	BB Branch and Bound	**`bound`**
11	LIN 3-Opt-Verfahren	**`nlost`**, **`maxnum`**
12	LIN2 Erweitertes 3-Opt, das die kürzesten Touren behandelt	**`nlost`**, **`maxnum`**, **`depth`**
13	LIN2REV Erweitertes 3-Opt, das die längsten Touren bearbeitet	**`nlost`**, **`maxnum`**, **`depth`**
14	LIN2ODD Erweitertes 3-Opt. Bearbeitung der kürzesten Touren mit ungeradem Index	**`nlost`**, **`maxnum`**, **`depth`**
15	LIN2EVEN Erweitertes 3-Opt. Bearbeitung der kürzesten Touren mit geradem Index	**`nlost`**, **`maxnum`**, **`depth`**
16	OPT2 Erweitertes 2-Opt-Verfahren	**`nlost`**, **`maxnum`**, **`depth`**
21	ER Edge Recombination Crossover	**`nlost`**, **`vfirst`**, **`vnext`**, **`vnone`**, **`p`**, **`pcorr`**
22	ERCASC Edge Recombination Crossover, kaskadiert (mehrere Generationen)	**`nlost`**, **`vfirst`**, **`vnext`**, **`vnone`**, **`p`**, **`pcorr`**, **`pgen`**, **`ngen`**
23	MUT4_LO3 4-Opt-Mutation mit anschließendem erweitertem 3-Opt, kaskadiert	**`nlost`**, **`maxnum`**, **`depth`**, **`p`**, **`pcorr`**, **`gentime`**
24	MUT4_LO2 4-Opt-Mutation mit anschließendem erweitertem 2-Opt, kaskadiert	**`nlost`**, **`maxnum`**, **`depth`**, **`p`**, **`pcorr`**, **`gentime`**
25	ER_LO3 ER Crossover mit anschließendem erweitertem 3-Opt, kaskadiert	**`nlost`**,**`maxnum`**,**`depth`**,**`vfirst`**,**`vnext`**,**`vnone`**,**`p`**,**`pcorr`**,**`gentime`**
26	ER_LO2 ER Crossover mit anschließendem erweitertem 2-Opt, kaskadiert	**`nlost`**,**`maxnum`**,**`depth`**,**`vfirst`**,**`vnext`**,**`vnone`**,**`p`**,**`pcorr`**,**`gentime`**

5. Experimente und Testläufe

Dieses Kapitel befaßt sich mit den Erfahrungen, die in der Praxis mit DOTT gemacht wurden. Hierzu wurden umfangreiche Testreihen durchgeführt, die sowohl die Einzelleistung (sequentielle Läufe) als auch die Teamleistung (Integration in Teamwork) der Experten betreffen. Es werden zum einen die vier zusammengesetzten genetischen Experten MUT4_LO3, MUT4_LO2, ER_LO3 und ER_LO2 auf ihre Leistungsfähigkeit hin untersucht, wofür sie mit den bisher implementierten Experten verglichen werden. Zum anderen wird analysiert, ob eine weitere Leistungssteigerung durch die Integration der Experten in Teamwork erreicht wird, insbesondere mit Hilfe der neuen Gutachter. Man erhofft sich hierbei, daß im Team jeweils eine Tour gefunden wird, die kürzer als die kürzeste ihrer beteiligten Experten ist, sowie gute Touren im Team schneller als im sequentiellen Fall gefunden werden. Zusätzlich sollen geeignete Parameterbelegungen für die neuen Gutachter REF_LENGTH und REF_DIVERS gefunden werden. Um die Testergebnisse zu veranschaulichen, sind sie, wo es sinnvoll erscheint, durch Graphiken dargestellt.

5.1. Sequentielle Leistung

Kommen wir nun zu den Testläufen im einzelnen. Dazu werden zuerst die vier zusammengesetzten, genetischen Experten im sequentiellen Fall untersucht. Es würde den Rahmen dieser Arbeit sprengen, eine erschöpfende Analyse der Experten mit Hilfe umfangreicher Parameterbelegungen durchzuführen. Aus diesem Grund beschränken wir uns hier auf wenige, sinnvoll erscheinende Belegungen, die jedoch bereits eine gute Aussage über die Leistungsfähigkeit der Experten ergeben. Die verwendeten TSP-Beispiele wurden allesamt der TSP-Bibliothek TSPLIB Version 1.2 (s. [Re91]) entnommen, ihre Eigenschaften (Städteanzahl der Beispiele, Länge der optimalen Lösung) sind im Anhang B nachzulesen. Die sequentiellen Testläufe wurden auf einer SUN SPARCStation 10 durchgeführt, auf der das Betriebssystem SUN OS 4.1 läuft.
Die Testläufe sind in zwei Phasen aufgeteilt und bestehen aus einer Tourkonstruktions- und einer Tourverbesserungsphase, die die eigentliche Arbeitsphase der zu testenden Experten darstellt. In der Konstruktionsphase wird mittels des Spezialisten FI eine Menge von Touren generiert. Um die kombinierten genetischen Verfahren einsetzen zu können, berechnet anschließend der Spezialist SORT die benötigten sortierten Listen der Nachbarstädte. Die Zeiten für FI und SORT sind für alle sequentiellen Testläufe (für ein gegebenes TSP) gleich und können in Tabelle 5.1.1 nachgelesen werden. Für die Verbesserungsphase besteht dagegen selbstverständlich eine Abhängigkeit vom jeweiligen Experten, da je nach Komplexität des Experten die entsprechende Laufzeit gewählt werden muß.
Was die Ergebnistabellen betrifft, werden folgende Schreibweisen verwendet. Der zugehörige prozentuale Wert zu einer Tour ist die Prozentzahl, um die die Tour länger ist als die optimale Lösung. Diese hätte demnach den Wert 0.0%. Zeitangaben zu Touren sind in dem Format *Stunden* **h** *Minuten*:*Sekunden*.*Zentisekunden* notiert. Sie geben den Entstehungszeitpunkt der jeweiligen Tour an. Als Hilfstellung für den Leser sind zusätzlich optimale Touren mit einem Symbol (*) markiert.

Beispiel	Zeit für FI	Anzahl erzeugter Touren von FI	Länge der kürzesten Tour von FI	%	Zeit für SORT
kroB100	00:01.20	91	22 693	2.49	00:00.10
gr202	00:10.00	197	41 875	4.27	00:00.20
pcb442	02:30.00	375	55 373	9.05	00:01.00
att532	05:00.00	516	29 198	5.46	00:01.00
gr666	11:40.00	614	318 627	8.24	00:01.50
u724	13:20.00	709	45 120	7.66	00:02.00
rat783	16:40.00	625	9 587	8.87	00:02.00
dsj1000	33:20.00	816	20 044 307	7.42	00:04.50

Tabelle 5.1.1: Eigenschaften der Tourkonstruktionsphase

Wie aus Tabelle 5.1.1 ersichtlich ist, sind die von FI produzierten Lösungen im Schnitt 6.68% länger als die optimale Lösung. In diesem Abschnitt gilt es zu analysieren, welche Verbesserungen die einzelnen Experten MUT4_LO3, MUT4_LO2, ER_LO3 und ER_LO2 erzielen können.

5.1.1. MUT4_LO2

Neben den Parametern der Konstruktionsphase, die ja lediglich vom gewählten TSP abhängig sind, gilt es, geeignete Parameterbelegungen für den Experten in der Verbesserungsphase zu bestimmen. Für ein gegebenes TSP wurden hierzu vier unterschiedliche, sinnvoll erscheinende Kombinationen gewählt, die im folgenden erläutert werden. Die maximale Anzahl der Durchgänge für eine Tour wird mit **`maxanz`**=200 (für alle Beispiele) auf einen Wert gesetzt, der praktisch nie erreicht wird. Dies hat sich in der Praxis als geeignet erwiesen, da eine Tour gewöhnlich schon nach wenigen Durchgängen optimal bezüglich der lokalen Optimierung ist (vgl. [Leo95], Abschnitt 5.1.4). Die Wahrscheinlichkeitsparameter werden mit **`p`**=50 und **`pcorr`**=100 derart gewählt, daß kürzere Touren bei der Auswahl zur Bearbeitung bevorzugt werden. Der Parameter **`depth`** wird auf 3 bzw. 5 gesetzt, während die Generationszeit **`gentime`** auf jeweils einen großen und kleinen Wert (dem Beispiel entsprechend) gesetzt wird. Es ergeben sich also vier verschiedene Parameterkombinationen für jedes TSP. Die Ergebnistabelle besteht aus einer Menge von Spalten; diese beinhalten der Reihe nach (von links nach rechts) folgende Angaben: den Namen des gewählten TSP, die Parameter **`depth`** und **`gentime`**, die Eigenschaften der kürzesten gefundenen Tour (Länge, Abweichung von der optimalen Lösung und Generierungszeitpunkt) und die Gesamtdauer des Testlaufs mit der Anzahl der erzeugten Generationen durch den Experten.

Beispiel	depth	gentime	kürzeste Tour			Lauf beendet	
			Länge	%	Zeit	Zeit	Gen.
kroB100	5	0.2s	22141*	0.00	2.46	21.30	100
	3	0.2s	22141*	0.00	10.05	21.30	100
	5	0.6s	22141*	0.00	14.36	1:01.30	100
	3	0.6s	22141*	0.00	59.30	1:01.30	100
gr202	5	0.5s	40160*	0.00	20.46	1:50.20	200
	3	0.5s	40187	0.07	36.40	1:50.20	200
	5	2s	40187	0.07	42.24	6:50.20	200
	3	2s	40187	0.07	2:34.57	6:50.20	200
pcb442	5	2.5s	51090	0.61	7:13.04	15:01.00	300
	3	2.5s	50991	0.42	5:20.53	15:01.00	300
	5	10s	51104	0.64	11:07.78	35:51.00	200
	3	10s	51133	0.72	33:11.69	35:51.00	200
att532	5	3s	27731	0.16	21:39.44	25:01.00	400
	3	3s	27750	0.25	22:43.33	25:01.00	400
	5	12s	27738	0.19	15:53.08	45:01.00	200
	3	12s	27750	0.23	19:04.36	45:01.00	200
gr666	5	4s	296872	0.85	21:50.41	45:01.50	500
	3	4s	295731	0.47	31:26.12	45:01.50	500
	5	16s	296575	0.75	34:08.39	1h 18:21.50	250
	3	16s	295795	0.49	1h 06:32.14	1h 18:21.50	250
u724	5	5s	42162	0.6	1h 02:49.70	1h 03:19.66	600
	3	5s	42127	0.52	49:45.16	1h 03:23.87	600
	5	20s	42080	0.41	1h 47:23.76	1h 47:23.76	300
	3	20s	42102	0.46	37:13.21	1h 53:23.43	300
rat783	5	5s	8856	0.57	52:57.59	1h 06:43.31	600
	3	5s	8856	0.57	50:49.34	1h 06:43.07	600
	5	20s	8829	0.26	1h 20:11.01	1h 56:43.14	300
	3	20s	8816	0.11	1h 47:50.97	1h 57:44.88	300
dsj1000	5	6s	18 822 209	0.87	1h 27:00.69	1h 43:26.43	700
	3	6s	18 807 068	0.79	1h 17:48.00	1h 43:24.62	700
	5	24s	18 749 543	0.48	2h 33:37.87	2h 53:24.85	350
	3	24s	18 812 512	0.82	2h 01:02.85	2h 53:25.08	350

Tabelle 5.1.2: Ergebnisse MUT4_LO2 (sequentiell)

Die errechneten Lösungen sind also im Durchschnitt 0.39% länger als die optimale Lösung, was eine erhebliche Steigerung gegenüber den Ergebnissen von FI bedeutet. Darüberhinaus wird die Leistung der bisher implementierten Experten (insbesondere LIN2, ER) bei weitem übertroffen (s. Tabelle 5.1.6).

5.1.2. MUT4_LO3

Die Generationszeiten müssen bei MUT4_LO3 natürlich größer gewählt werden als bei MUT4_LO2, da die lokale Optimierung komplexer ist und somit von MUT4_LO3 weniger Touren in der gleichen Zeit als von MUT4_LO2 erzeugt werden. Die weiteren Parameter (für die Konstruktionsphase sowie für den Experten selber) sind identisch zu denen beim sequentiellen Lauf von MUT4_LO2.

Beispiel	depth	gentime	kürzeste Tour			Lauf beendet	
			Länge	%	Zeit	Zeit	Gen.
kroB100	5	1s	22199	0.26	2.90	15.99	15
	3	1s	22141*	0.00	11.97	15.90	15
	5	4s	22141*	0.00	16.32	58.02	15
	3	4s	22141*	0.00	20.58	57.92	15
gr202	5	5s	40160*	0.00	4:10.01	6:17.08	73
	3	5s	40160*	0.00	1:41.49	6:15.85	73
	5	20s	40160*	0.00	4:58.28	8:12.06	25
	3	20s	40187	0.07	3:45.32	8:11.11	25
pcb442	5	40s	50814	0.07	41:53.86	47:48.80	69
	3	40s	50952	0.34	39:50.34	47:40.40	69
	5	160s	51004	0.45	35:20.42	1h 33:08.53	35
	3	160s	50912	0.26	1h 18:31.16	1h 33:16.59	35
att532	5	75s	27749	0.23	47:44.17	1h 00:22.91	45
	3	75s	27743	0.21	45:38.73	1h 00:21.32	45
	5	300s	27756	0.25	1h 58:12.79	2h 30:27.19	30
	3	300s	27750	0.23	2h 21:47.85	2h 30:19.35	30
gr666	5	120s	295000	0.22	2h 15:39.05	2h 34:16.48	73
	3	120s	295670	0.45	2h 32:28.58	2h 34:15.09	73
	5	480s	295376	0.35	2h 18:13.75	2h 34:06.53	19
	3	480s	294831	0.16	2h 21:00.76	2h 33:52.87	19
u724	5	250s	42145	0.56	3h 38:04.47	3h 40:37.78	48
	3	250s	42001	0.22	3h 19:20.56	3h 36:23.15	48
	5	1000s	42292	0.91	3h 17:17.91	3h 32:24.22	12
	3	1000s	42233	0.77	3h 31:26.20	3h 33:04.84	12
rat783	5	250s	8862	0.64	3h 39:47.31	3h 45:03.26	48
	3	250s	8896	1.02	3h 07:45.01	3h 40:21.48	48
	5	1000s	8894	1.00	3h 32:00.40	3h 36:42.86	12
	3	1000s	8887	0.92	3h 36:22.56	3h 36:43.15	12
dsj1000	5	450s	18906749	1.32	6h 21:59.67	6h 33:27.18	48
	3	450s	18866341	1.11	6h 28:15.52	6h 33:27.11	48
	5	1800s	18872543	1.14	6h 32:37.34	6h 33:29.13	12
	3	1800s	18857869	1.06	6h 30:40.40	6h 33:26.99	12

Tabelle 5.1.3: Ergebnisse MUT4_LO3 (sequentiell)

Die Ergebnisse von MUT4_LO3 sind also im Schnitt 0.44% länger als die optimale Lösung, und damit meist schlechter als die von MUT4_LO2.

5.1.3. ER_LO2

Die Wahl der Parameter für die Testläufe von ER_LO2 orientiert sich an denen von MUT4_LO2. Insbesondere gleichen sich die Eigenschaften der Konstruktionsphase. Was die Verbesserungsphase betrifft, werden die Parameter von ER_LO2 folgendermaßen belegt. Der

Parameter **maxanz** wird für alle Testläufe konstant auf 200 gesetzt (vgl. Abschnitt 5.1.1), **vfirst** auf FIRST_MIN_5, **vnext** auf NEXT_MIN_USED und **vnone** auf NONE_SORTED (vgl. [Leo95], Abschnitt 4.5.4). Da MUT4_LO2 und ER_LO2 ähnliche Laufzeiten aufgrund der gleichen lokalen Optimierung haben, sind die Generationszeiten für die jeweiligen Beispiele identisch. Es kommen also wiederum (analog zu MUT4_LO3 und MUT4_LO2) vier verschiedene Parameterbelegungen pro Beispiel für ER_LO2 zum Einsatz durch Variation von **depth** und **gentime**.

Beispiel	**depth**	**gentime**	kürzeste Tour			Lauf beendet	
			Länge	%	Zeit	Zeit	Gen.
kroB100	5	0.2s	22199	0.26	12.69	21.30	100
	3	0.2s	22235	0.42	1.90	21.30	100
	5	0.6s	22234	0.42	3.13	1:01.30	100
	3	0.6s	22199	0.26	3.24	1:01.30	100
gr202	5	0.5s	40187	0.07	31.12	1:50.20	200
	3	0.5s	40187	0.07	20.72	1:50.20	200
	5	2s	40160*	0.00	31.00	6:50.20	200
	3	2s	40160*	0.00	29.17	6:50.20	200
pcb442	5	2.5s	51206	0.84	2:35.72	15:01.00	300
	3	2.5s	51126	0.69	7:07.25	15:01.00	300
	5	10s	51282	0.99	25:15.64	35:51.00	200
	3	10s	51267	0.96	18:43.24	35:51.00	200
att532	5	3s	27994	1.11	19:06.76	25:01.00	400
	3	3s	28088	1.45	10:19.98	25:01.00	400
	5	12s	27800	0.41	10:43.65	45:01.00	200
	3	12s	27785	0.36	17:48.02	45:01.00	200
gr666	5	4s	299853	1.87	32:52.89	45:01.50	500
	3	4s	297159	0.95	13:36.48	45:01.50	500
	5	16s	296276	0.65	35:36.48	1h 18:21.50	250
	3	16s	298241	1.32	21:24.84	1h 18:21.50	250
u724	5	5s	42347	1.04	32:19.00	1h 02:54.93	600
	3	5s	42761	2.03	24:28.01	1h 03:13.14	600
	5	20s	42227	0.76	22:43.64	1h 53:23.61	300
	3	20s	42292	0.91	34:01.62	1h 53:22.86	300
rat783	5	5s	8995	2.15	29:53.05	1h 06:44.24	600
	3	5s	8965	1.81	25:55.47	1h 06:45.47	600
	5	20s	8847	0.47	1h 12:24.64	1h 56:44.02	300
	3	20s	8874	0.77	25:45.37	1h 56:43.55	300
dsj1000	5	6s	19 031 367	1.99	39:37.26	1h 43:25.18	700
	3	6s	19 144 650	2.60	1h 31:38.27	1h 43:25.44	700
	5	24s	18 852 390	1.03	1h 21:29.45	2h 53:26.50	350
	3	24s	18 803 744	0.77	2h 16:15.65	2h 53:26.05	350

Tabelle 5.1.4: Ergebnisse ER_LO2 (sequentiell)

Die Ergebnisse sind also durchschnittlich 0.92% länger als die optimale Lösung, und damit im Durchschnitt schlechter als die von MUT4_LO3 und MUT4_LO2.

5.1.4. ER_LO3

Analog zu MUT4_LO3 sind für diesen Experten längere Generationszeiten als für ER_LO2 nötig. Da MUT4_LO3 und ER_LO3 die gleiche lokale Optimierung haben, sind die Generationszeiten für die jeweiligen Beispiele identisch.

Beispiel	depth	gentime	kürzeste Tour			Lauf beendet	
			Länge	%	Zeit	Zeit	Gen.
kroB100	5	1s	22199	0.26	12.92	16.30	15
	3	1s	22199	0.26	5.93	16.30	15
	5	4s	22141*	0.00	19.29	1:01.30	15
	3	4s	22141*	0.00	26.76	1:01.30	15
gr202	5	5s	40160*	0.00	4:12.65	6:15.20	73
	3	5s	40160*	0.00	1:58.95	6:15.20	73
	5	20s	40160*	0.00	6:46.24	8:30.20	25
	3	20s	40160*	0.00	1:52.83	8:30.20	25
pcb442	5	40s	50932	0.30	45:54.00	48:31.00	69
	3	40s	50932	0.30	18:49.36	48:31.00	69
	5	160s	50931	0.30	1h 06:06.09	1h 35:51.00	35
	3	160s	50795	0.03	1h 29:38.14	1h 35:51.00	35
att532	5	75s	27704	0.07	41:40.81	1h 01:16.00	45
	3	75s	27759	0.26	37:46.33	1h 01:16.00	45
	5	300s	27725	0.14	2h 33:11.17	2h 35:01.00	30
	3	300s	27782	0.35	1h 06:06.89	2h 35:01.00	30
gr666	5	120s	296460	0.71	2h 29:48.78	2h 37:41.50	73
	3	120s	295850	0.51	1h 26:23.02	2h 37:41.50	73
	5	480s	294700	0.12	2h 38:57.04	2h 43:41.50	19
	3	480s	296407	0.70	2h 40:01.86	2h 43:41.50	19
u724	5	250s	42061	0.36	3h 22:31.31	3h 36:32.96	48
	3	250s	42084	0.42	2h 14:00.43	3h 40:36.20	48
	5	1000s	42156	0.59	3h 26:17.08	3h 32:33.30	12
	3	1000s	42159	0.59	3h 00:10.31	3h 33:23.75	12
rat783	5	250s	8862	0.64	3h 18:43.23	3h 44:56.34	48
	3	250s	8846	0.45	2h 43:47.24	3h 40:15.22	48
	5	1000s	8893	0.99	3h 35:28.84	3h 36:38.37	12
	3	1000s	8902	1.09	3h 07:08.28	3h 36:41.50	12
dsj1000	5	450s	18751750	0.49	6h 28:45.45	6h 33:25.70	48
	3	450s	18848234	1.01	6h 27:11.16	6h 33:28.99	48
	5	1800s	18914905	1.37	6h 14:47.70	6h 33:31.26	12
	3	1800s	18854109	1.04	6h 07:56.20	6h 33:25.46	12

Tabelle 5.1.5: Ergebnisse ER_LO3 (sequentiell)

Die Ergebnisse sind im Schnitt 0.42% länger als die optimale Lösung, und damit durchschnittlich besser als die von ER_LO2.

Vergleich mit den bisher implementierten Experten

Es soll nun analysiert werden, ob die Kombination von genetischen Verfahren mit lokaler Optimierung den gewünschten Erfolg erzielt hat. Insbesondere soll beurteilt werden, ob sich die Verwendung einer schnelleren lokalen Optimierung bewährt. Hierzu werden die Ergebnisse verglichen mit der besten errechneten Tour der bisher implementierten Experten.

	bisheriger bester Experte			neuer Experte		
Beispiel	kürzeste Lösung	Name des Experten	Zeit	kürzeste Lösung	Name des Experten	Zeit
kroB100	22251	ER	2:25.43	22141*	MUT4_LO2	2.46
gr202	40549	LIN2	28.03	40160*	MUT4_LO2	20.46
pcb442	52238	LIN2	42:55.84	50795	ER_LO3	1h 29:38.14
att532	28152	LIN2	30:50.96	27704	ER_LO3	41:40.81
gr666	302818	LIN2	25:38.11	294700	ER_LO3	2h 38:57.04

Tabelle 5.1.6: Vergleich der neuen mit den bisher implementierten Experten

Die enorme Leistungsfähigkeit der zusammengesetzten Experten kommt in dieser Übersicht zum Ausdruck. Für jedes der getesteten Beispiele gibt es einen neuen Experten und eine zugehörige Parameterbelegung, mit denen eine Tour gefunden wird, die kürzer ist als die von den bisherigen Experten errechnete beste Lösung. Darüberhinaus wird sogar mit jedem der vier Experten mit jeder der gewählten Parameterbelegungen eine solche kürzere Tour gefunden ! (vgl. Tabellen 5.1.2 - 5.1.5). Die vier neuen Experten übertreffen mit ihrer Leistung demnach sämtliche der bisherigen. Die errechneten Touren sind durchschnittlich nur etwa 0.5% länger als die optimale Lösung.
Weiterhin ist ein Vergleich mit dem reinen genetischen Verfahren ER interessant, um zu erfahren, welche Ergebnisse durch die Kombination mit lokaler Optimierung erzielt werden. Hier eine Gegenüberstellung der beiden Verfahren ER und LIN2 mit ihrer direkten Zusammensetzung in Form des Experten ER_LO3 (nur für das Beispiel gr202). Hierbei wurde zum einen die reine Leistung von ER gemessen, und zum anderen die Kopplung mit LIN2, indem die Resultate von ER durch LIN2 lokal optimiert wurden.

ER			ER_LO3	
Nachbearbeitung durch LIN2	Länge der kürzesten Tour	Zeit der kürzesten Tour	Länge der kürzesten Tour	Zeit der kürzesten Tour
nein	41283	1:08.44	40160*	1:52.83
ja	40529	> 1:41.97		

Tabelle 5.1.7: Steigerung genetischer Verfahren durch Kombination mit anderen

Auch hier wird die enorme Leistungsfähigkeit der zusammengesetzten Experten deutlich. Zum einen errechnet ER_LO3 eine kürzere Tour als ER alleine, zum anderen ergibt die direkte Zusammensetzung in einem Experten ein besseres Ergebnis als die Hintereinanderausführung von ER und LIN2. Zusammenfassend kann also die Kombination genetischer Verfahren mit lokaler Optimierung als voller Erfolg angesehen werden. Gute Ergebnisse werden von zusammengesetzten Experten wesentlich schneller gefunden.

Vergleich der 2-Opt-Experten mit den 3-Opt-Experten

Es soll nun der Einfluß der lokalen Optimierung auf die erreichten Ergebnisse analysiert werden. Hierzu werden die durchschnittlichen Abweichungen von der optimalen Tour jeweils miteinander verglichen. Für die Durchschnittsbildung wurden sämtliche sequentiellen Testläufe (also mit allen Parameterbelegungen) berücksichtigt.

gewählter genetischer Teil	durchschnittliche Abweichung 3-Opt-Experte (...LO3)	durchschnittliche Abweichung 2-Opt-Experte (...LO2)
MUT4_...	0.44%	0.39%
ER_...	0.42%	0.92%

Tabelle 5.1.8: Vergleich 2-Opt-Team mit 3-Opt-Team

Wie aus der Tabelle hervorgeht, werden mit MUT4_LO2 im Schnitt die besten Lösungen der vier neuen Experten gefunden. Es ist bemerkenswert, daß die Ergebnisse der 2-Opt-Experten etwa im gleichen qualitativen Bereich sind wie die der 3-Opt-Experten, obwohl die Laufzeit der 2-Opt-Experten ja wesentlich geringer ist. Bei MUT4_LO2 werden sogar im Schnitt bessere Ergebnisse als mit MUT4_LO3 gefunden ! Es zeigt, daß es sich tatsächlich lohnt, einen schnelleren lokalen Optimierer zu verwenden, auch wenn dieser nicht so leistungsfähig ist wie ein komplexerer. Auch beim Vergleich ER_LO3 mit ER_LO2 schneidet der 2-Opt-Experte erstaunlich gut ab. Die Qualität der Ergebnisse ist trotz stark unterschiedlicher Laufzeit vergleichbar, wenn auch ER_LO3 (im Gegensatz zu MUT4_LO3 im Vergleich zu MUT4_LO2) bessere Ergebnisse produziert als ER_LO2. Eine mögliche Erklärung hierfür ist, daß Mutation zwar großes Potential, aber wenig Ausrichtung hat. Crossover dagegen operiert im wesentlichen auf schon vorhandener Information, daher macht sich die leistungsfähigere Optimierung hier stärker bemerkbar. Zusammengefaßt läßt sich die Einbeziehung einer schnelleren lokalen Optimierung als großer Erfolg verbuchen. Im nächsten Abschnitt soll nun analysiert werden, welche weiteren Steigerungen durch die Integration in Teamwork erreicht werden.

5.2. Teamleistung

Dieser Abschnitt befaßt sich mit den Erfahrungen, die mit der Einbettung der vier Experten MUT4_LO3, MUT4_LO2, ER_LO3 und ER_LO2 in Teamwork gemacht wurden. Es werden ausschließlich diese vier Experten verwendet, da sie die bisher leistungsfähigsten in DOTT darstellen. Es sei bemerkt, daß die erreichten Ergebnisse durchschnittlich um weniger als 1% von der optimalen Lösung abweichen. Die Experten sind also schon im sequentiellen Fall sehr leistungsfähig, daher ist es generell schwierig, eine weitere Verbesserung zu erzielen. Die Teamläufe wurden ausgeführt auf zwei SUN SPARCStations 10 mit dem Betriebssystem SUN OS 4.1. Es wurden im wesentlichen zwei Teams gebildet: die 3-Opt-Experten bilden das erste Team, die 2-Opt-Experten das zweite, so daß jeweils zwei Rechner an einem Teamlauf beteiligt sind. Man erhofft sich von den Teamläufen, daß die besten sequentiellen Ergebnisse im Team schneller gefunden werden als im sequentiellen Fall. Darüberhinaus sollen selbstverständlich bessere Touren als die besten sequentiellen gefunden werden.
Wie bereits in Abschnitt 5.1 erläutert, teilt sich ein Testlauf in zwei Phasen auf: in eine Tourkonstruktions- und eine Tourverbesserungsphase, an der die beiden zu testenden Experten beteiligt sind. Für die Konstruktionsphase wird (wie im sequentiellen Fall) der Spezialist FI zur Generierung einer Ausgangspopulation verwendet, parallel dazu läuft der Spezialist SORT, der die sortierten Listen der Nachbarstädte erzeugt. Die Zeiten für die Konstruktionsphase sind demnach identisch zu denen der sequentiellen Läufe (s. Tabelle 5.1.1, Einträge für FI), um einen fairen Vergleich zu ermöglichen. In der darauffolgenden Verbesserungsphase arbeiten die Experten MUT4_LO3 und ER_LO3 (bzw. MUT4_LO2 und ER_LO2) parallel, wobei zu festen Zeitpunkten Teamtreffen einberufen werden. Selbstverständlich nimmt diese zweite Phase den weitaus größten Teil eines Testlaufes ein. In Tabelle 5.2.2 ist der Aufbau dieser Verbesserungsphase (Anzahl der Teamtreffen, Anzahl der Generationen eines Zyklus) dargestellt.
Die vier getesteten Experten gehören alle der Gruppe der genetischen Verfahren an. Da diese stark auf Zufallsalgorithmen beruhen, erweist sich die Analyse ihrer Arbeit als nicht trivial. Bereits im sequentiellen Fall ist ein Testlauf i.a. nicht wiederholbar, sondern abhängig von Rechnerauslastung und ähnlichen Faktoren. Dieser Effekt wird in den Teamläufen noch verstärkt, da hier noch die Kommunikationszeiten berücksichtigt werden müssen, die ebenfalls nicht deterministisch sind. Was die Parameter für die Experten betrifft, sind diese im wesentlichen zu denen der sequentiellen Läufe identisch. Darüberhinaus müssen die Parameter eingestellt werden, die den Teamlauf beeinflussen, insbesondere die der Gutachter. Zu jedem TSP-Beispiel wurden hierfür 10 Testläufe mit verschiedenen Parameterbelegungen durchgeführt (s. Tabelle 5.2.1). Der Sinn der Testläufe L1...L6 ist der, daß eine geeignete Einstellung für den Gutachter REF_LENGTH (Gesamtexpertengutachter) gefunden werden soll, wobei jedes der ihm eigenen Kriterien in einem eigenen Testlauf berücksichtigt wird. Die Einstellung, die das beste Ergebnis ergibt, wird für die Testläufe D1...D4 übernommen, deren Zweck es ist, nun eine geeignete Parametereinstellung für den Gutachter REF_DIVERS zu finden. In den Dx-Testläufen soll daher untersucht werden, ob es sich in der Praxis bewährt, die Unterschiedlichkeit von Lösungen bei der Auswahl durch den Gutachter zu berücksichtigen. Es ist zu erwarten, daß eine alleinige Berücksichtigung der Unterschiedlichkeit als einziges Kriterium zu einer Verschlechterung der Ergebnisse führt, da auf diese Weise ja i.a. längere Touren übernommen werden. Auf der anderen Seite sollte die ausschließliche Wahl der kürzesten Touren dazu führen, daß das System leicht in lokale Minima konvergiert. Es sollte sich dennoch eine günstige Konstellation der beiden Parameter finden lassen (etwa mit den Werten 50-50 oder 90-10 bzw. 10-90) mit der kürzere Touren schneller gefunden werden und das System insgesamt bessere Ergebnisse findet. Es sei darauf

hingewiesen, daß in der Parameterbelegung für die Testläufe L1...L6 der Gutachter REF_DIVERS dem Gutachter REF_SHORTEST vollständig in seiner Arbeitsweise gleicht.

Testlauf Nr.	Parameter REF_LENGTH						Parameter REF_DIVERS	
	p1	**p2**	**p3**	**p4**	**p5**	**p6**	**p_diversity**	**p_length**
L1	1000	0	0	0	0	0	0	100
L2	0	1000	0	0	0	0	0	100
L3	0	0	1000	0	0	0	0	100
L4	0	0	0	1000	0	0	0	100
L5	0	0	0	0	1000	0	0	100
L6	0	0	0	0	0	1000	0	100
D1	wie bester Lx-Lauf						100	0
D2	wie bester Lx-Lauf						75	25
D3	wie bester Lx-Lauf						50	50
D4	wie bester Lx-Lauf						25	75

Tabelle 5.2.1: Parameterbelegungen der Gutachter für die Teamläufe

Die weiteren Parameter für die Teamläufe sind den jeweiligen Beispielen und Experten angepaßt und auf folgende Weise gebildet. Der Parameter **anz** für den Gutachter REF_LENGTH wird auf etwa 10% der Anzahl der produzierten Lösungen eines Experten gesetzt. Die Anzahl der Lösungen, die von einem Verlierer übernommen werden (Parameter **nlost**), ist ebenfalls etwa 10% der Anzahl der produzierten Touren. Die Anzahl der gleichzeitig im Speicher zu haltenden Lösungen (Populationsgröße) brauchte nicht zusätzlich beschränkt zu werden, da diese implizit durch die Generationszeiten beschränkt wird. Die Generationszeiten der Experten sind die jeweils längeren der sequentiellen Läufe, der Parameter **depth** wurde für alle Beispiele auf den Wert 3 gesetzt. Hier eine Übersicht über den Ablauf der Verbesserungsphasen:

	3-Opt-Team		2-Opt-Team	
Beispiel	Anzahl der Generationen eines Zyklus	Anzahl der Zyklen	Anzahl der Generationen eines Zyklus	Anzahl der Zyklen
kroB100	5	3	7	15
gr202	5	6	10	20
pcb442	7	5	10	20
att532	6	5	10	20
gr666	7	3	12	20
u724	3	4	15	20
rat783	3	4	15	20
dsj1000	3	4	15	20

Tabelle 5.2.2: Einstellungen für die Teamläufe

Der Vollständigkeit halber sei abschließend noch eine Übersicht über die Expertenparameter gegeben:

ER_LO2/3 :	**`nlost`**	(etwa 10% der Anzahl der produzierten Lösungen)
	`maxnum`	200
	`depth`	3
	`vfirst`	FIRST_MIN_5
	`vnext`	NEXT_MIN_USED
	`vnone`	NONE_SORTED
	`p`	50
	`pcorr`	100
	`gentime`	(längere Generationszeit des sequentiellen Laufes)
MUT4_LO2/3:	**`nlost`**	(etwa 10% der Anzahl der produzierten Lösungen)
	`maxnum`	200
	`depth`	3
	`p`	50
	`pcorr`	100
	`gentime`	(längere Generationszeit des sequentiellen Laufes)

5.2.1. MUT4_LO3 und ER_LO3

Kommen wir nun zur Analyse des ersten Teams, das aus den 3-Opt-Experten besteht. Diese beiden Experten sind von der Laufzeit her sehr ähnlich (s. Abschnitt 5.1), daher bietet sich ihre Zusammenarbeit im Team an. Es ist zu untersuchen, ob die kürzesten sequentiellen Ergebnisse früher gefunden werden (Spalte "Zeit sequentielle Tour") und natürlich eine bessere Lösung in der gleichen Arbeitszeit gefunden wird.

Beispiel	kürzeste sequentielle Tour der beteiligten Experten		Teamleistung			
	Länge	Zeit	Länge kürzeste Tour	Zeit kürzeste Tour	Zeit sequentielle Tour	Gesamt-laufzeit
kroB100	22 141*	11.97	22 141*	5.89	5.89	1:05.19
gr202	40 160*	1:41.49	40 160*	1:11.40	1:11.40	10:13.44
pcb442	50 795	1h 29:38.14	50 785	1h 08:20.87	< 29:44.08	1h 36:09.49
att532	27 704	41:40.81	27 703	2h 11:25.30	2h 11:25.30	2h 34:37.75
gr666	294 700	2h 38:57.04	294 522	2h 55:56.31	< 2h 27:07.48	2h 58:17.28
u724	42 001	3h 19:20.56	41 956	3h 17:55.61	< 2h 44:00.87	3h 34:23.16
rat783	8 846	2h 43:47.24	8 836	3h 26:18.32	< 3h 08:50.82	3h 38:02.02
dsj1000	18 751 750	6h 28:45.45	18 768 891	6h 22:30.42	-	6h 35:07.54

Tabelle 5.2.3: Vergleich 3-Opt-Team mit sequentiellen Läufen

Wie aus Tabelle 5.2.3 ersichtlich ist, gibt es zu fast jedem Beispiel (Ausnahme: dsj1000) einen Teamlauf mit einer zugehörigen Parameterbelegung, in dem die kürzeste sequentielle Lösung auf jeden Fall gefunden wird. Darüberhinaus werden im Team kürzere Touren als diese errechnet. Wenn die Verbesserung bei den 3-Opt-Experten auch noch nicht so offensichtlich

ist, so ist doch zumindestens eine Steigerung der Leistung durch die Integration in Teamwork ersichtlich: es wird eine Tour errechnet, die weder von dem einen noch von dem anderen Experten (sequentiell) in dieser Zeit gefunden wird. Stärker ersichtlich wird dieser Effekt bei dem 2-Opt-Team, das in Abschnitt 5.2.2 behandelt wird.
Der zweite Aspekt, der von Interesse ist, ist das Finden von geeigneten Parameterbelegungen für die neuen Gutachter. Hierzu ist eine graphische Darstellung der Testergebnisse hilfreich, die die jeweiligen Abweichungen von der optimalen Lösung beinhaltet (Abb. 5.2.1). Da die Beispiele kroB100 und gr202 in diesem Zusammenhang keine Aussage liefern (ihre kürzeste Tour ist für jede Belegung gleich), sind diese nicht im Diagramm berücksichtigt. Das Diagramm ist in gestapelter Darstellung, d.h. die jeweiligen Werte werden für das nachfolgende Beispiel jeweils aufaddiert, so daß man leicht einen durchschnittlichen Gütefaktor für die jeweilige Parametereinstellung erkennen kann.

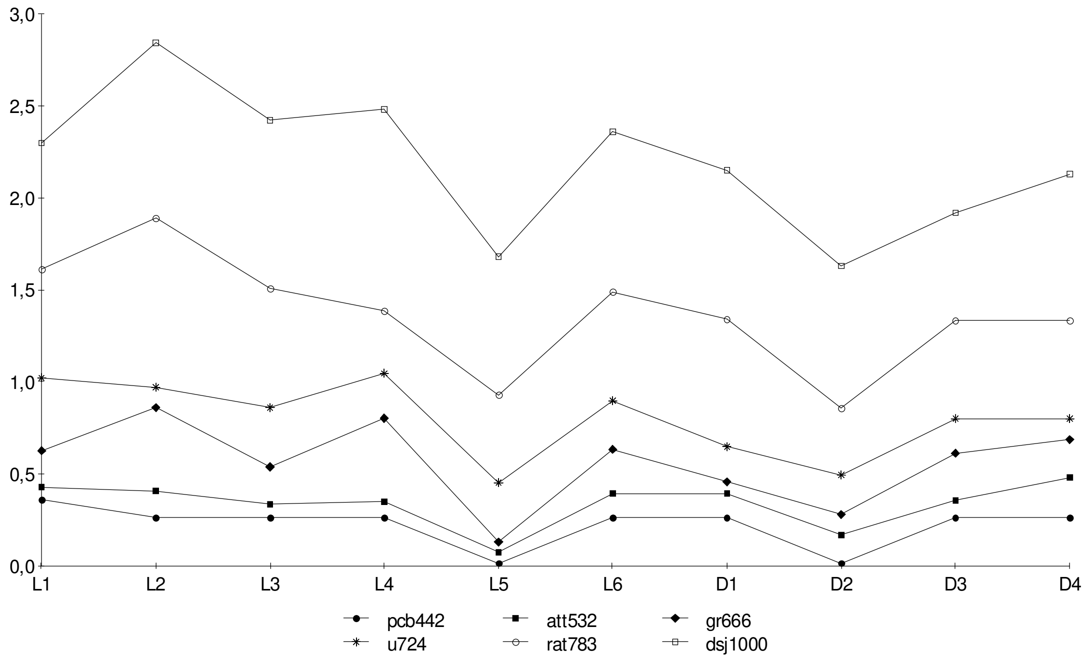

Abb. 5.2.1: gestapeltes Diagramm der prozentualen Abweichungen (3-Opt-Team)

Offenbar ergibt die L5-Parameterbelegung im Schnitt die besten Ergebnisse, was den Gutachter REF_LENGTH zur Beurteilung der Experten betrifft (bei dieser Einstellung sind die **`anz`** unähnlichsten Touren alleine relevant). Interessant in diesem Zusammenhang ist das relativ schlechte Abschneiden mit der Einstellung L2, bei der die Durchschnittslänge aller Touren berücksichtigt wird. Die Auswirkungen der Variationen der Parameter für REF_DIVERS (D1...D4) sind mit diesem Team offenbar nicht in jedem Fall positiv. Zwar wird eine leichte Verbesserung mit der Parameterbelegung D2 (**`p_diversity`**=75, **`p_length`**=25) erreicht, jedoch ist diese minimal. Die anderen Belegungen (D1, D3 und D4) haben allesamt negative Auswirkungen auf das Gesamtergebnis. Deutlich wird dies auch in Abb. 5.2.2, in der die prozentualen Abweichungen von der optimalen Tour im (ungestapelten) Liniendiagramm dargestellt sind. Die einzige Verbesserung bei D1...D4 im Gegensatz zu der Belegung **`p_diversity`**=0, **`p_length`**=100 ergibt sich bei dem Beispiel dsj1000; bei den anderen Beispielen führt diese Variation entweder zu gar keiner oder zu einer negativen Veränderung des Ergebnisses. Es scheint, daß der Einsatz von REF_DIVERS im Zusammenhang mit dem 3-Opt-Team nicht unbedingt positiv sein muß. Erklären ließe sich das damit, daß diese Experten ohnehin sehr wenige Lösungen produzieren, und somit eine

Auswahl von zur kürzesten Lösung unterschiedlichen Touren zu einer Verschlechterung der (recht kleinen) Population führt. Trotz dieser Einschränkungen ist mit der Ausnahme des Beispiels dsj1000 für jedes TSP der Einsatz der Experten im Team als positiv zu vermerken: es werden Touren gefunden, die kürzer sind als die beste der beteiligten Experten.

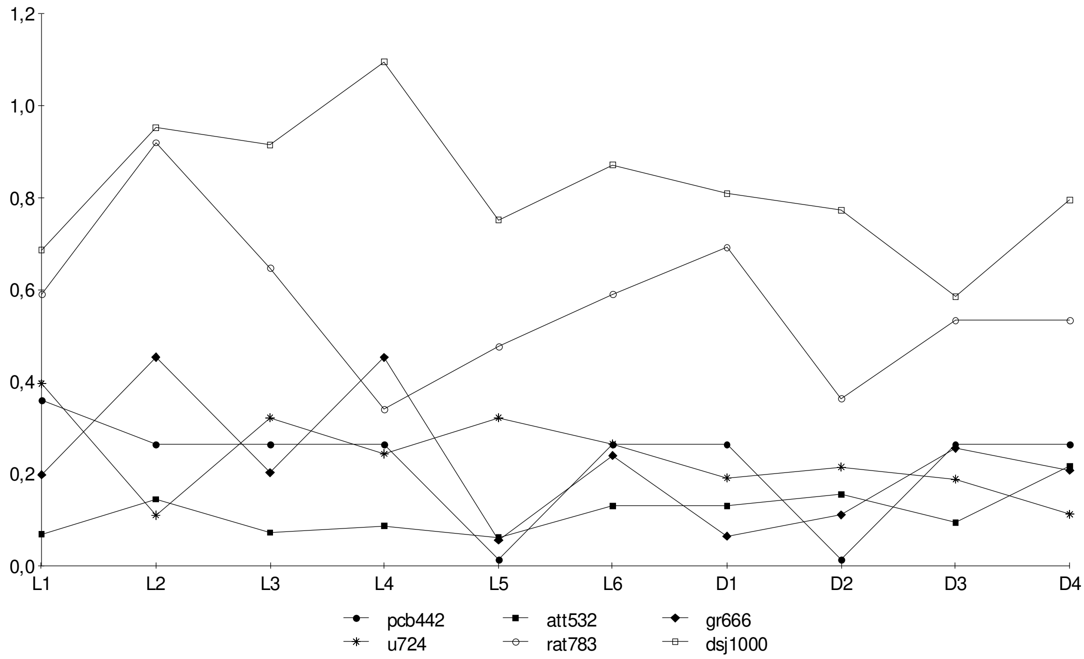

Abb. 5.2.2: Liniendiagramm der prozentualen Abweichungen (3-Opt-Team)

5.2.2. MUT4_LO2 und ER_LO2

Ebenso wie das 3-Opt-Team passen diese beiden Experten hervorragend zueinander, da sie eine vergleichbare Komplexität haben. Die Parameterbelegungen dieses Teams entsprechen weitestgehend denen des 3-Opt-Teams, mit dem Unterschied, daß natürlich andere Generationszeiten verwendet werden (jeweils die längere der beiden Generationszeiten des sequentiellen Laufs) sowie der Parameter **`nlost`** auf die jeweiligen Experten angepaßt wird. Wie schon bei dem 3-Opt-Team läßt sich aus den Ergebnissen (Tabelle 5.2.4) die Leistungsfähigkeit von Teamwork ablesen. Die kürzesten sequentiellen Ergebnisse werden mit der einzigen Ausnahme des Beispiels rat783 im Team gefunden, und dies in den meisten Fällen sogar schneller. In dem Beispiel u724 wird diese Tour sogar in weniger als einem Drittel der Zeit der sequentiellen Tour gefunden (Synergieeffekt) ! Darüberhinaus wird in den Fällen, in denen die optimale Lösung noch nicht im sequentiellen Fall gefunden wurde, eine bessere Tour im Team errechnet (Ausnahme: rat783). Für das Beispiel pcb442 wird sogar erstmalig die optimale Lösung gefunden. Es zeigt sich also noch stärker als bei dem 3-Opt-Team, daß mit Hilfe der Teamwork-Methode verschiedene Verfahren in ein System integriert werden können, die sich gegenseitig begünstigen (Kooperation).

Beispiel	kürzeste sequentielle Tour der beteiligten Experten		Teamleistung			
	Länge	Zeit	Länge kürzeste Tour	Zeit kürzeste Tour	Zeit sequentielle Tour	Gesamt-laufzeit
kroB100	22 141*	2.46	22 141*	5.28	5.28	1:20.22
gr202	40 160*	20.46	40 160*	21.98	21.98	6:59.75
pcb442	50 991	5:20.53	50 778*	22:05.34	< 5:38.41	34:35.15
att532	27 731	21:39.44	27 705	15:42.04	< 11:48.30	44:02.87
gr666	295 731	31:26.12	294 435	25:29.50	< 15:15.27	1h 17:41.13
u724	42 080	1h 47:23.76	41 994	1h 34:17.40	< 28:31.28	1h 49:16.91
rat783	8 816	1h 47:50.97	8 829	1h 19:20.33	-	1h 54:43.88
dsj1000	18 749 543	2h 33:37.87	18 740 224	1h 56:31.66	< 1h 11:11.90	2h 30:04.47

Tabelle 5.2.4: Vergleich 2-Opt-Team mit sequentiellen Läufen

Um eine geeignete Parameterbelegung der Gutachter zu erhalten, werden die Testergebnisse wiederum in graphischen Darstellungen (Abb. 5.2.3 und 5.2.4) verdeutlicht. Wie schon bei dem 3-Opt-Team werden hier die Beispiele kroB100 und gr202 nicht berücksichtigt.

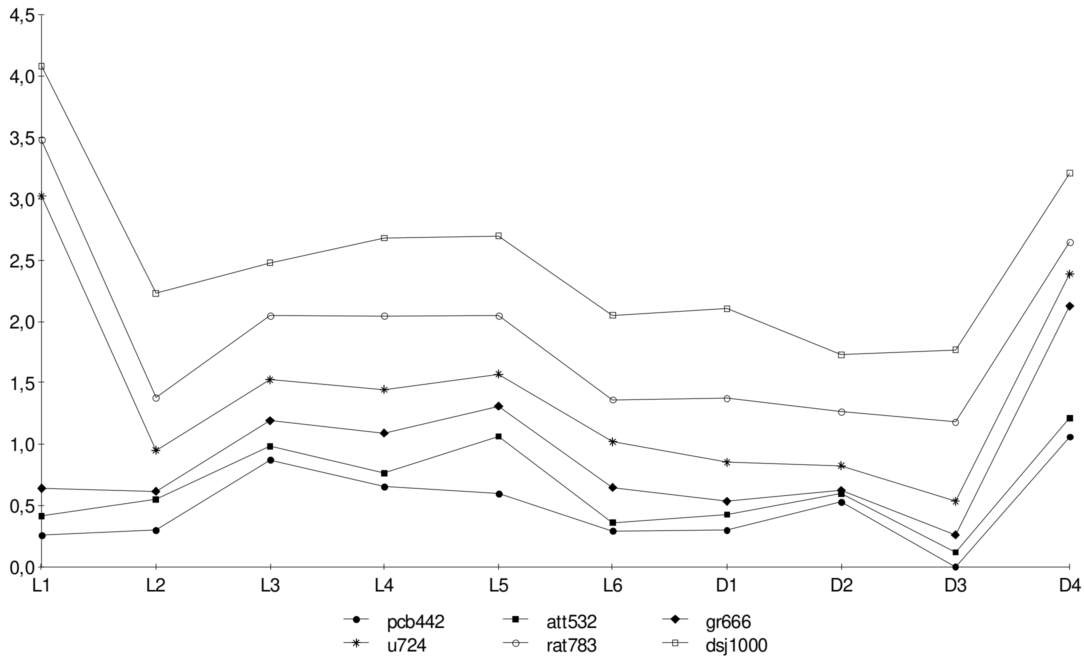

Abb. 5.2.3: Gestapeltes Diagramm der prozentualen Abweichungen (2-Opt-Team)

Aus Abb. 5.2.3 läßt sich die durchschnittliche Leistung des Teams mit den jeweiligen Parameterbelegungen ablesen. Offenbar erzielt die Belegung L6 (Berücksichtigung der Anzahl der Lösungen) im Schnitt die besten Ergebnisse für die Lx-Läufe. Man sieht, daß es sich also für das 2-Opt-Team tatsächlich lohnt, die Quantität der produzierten Menge zu betrachten. Interessant ist das schwache Abschneiden der Belegung L1 (Länge der kürzesten Tour berücksichtigt), sowie das relativ gute Ergebnis für die Belegung L2 (Durchschnittslänge aller

Touren), da mit dieser Belegung beim 3-Opt-Team ja das schlechteste Ergebnis erzielt wurde. Was die Variation der Parameter für REF_LENGTH in D1...D4 betrifft, so läßt sich hier eine deutliche Verbesserung der Ergebnisse erkennen gegenüber der reinen Berücksichtigung der Länge der Touren. Wie schon beim 3-Opt-Team erzielt auch hier die Einstellung D2 (**`p_diversity`**=75, **`p_length`**=25) die besten Resultate. Diese Übereinstimmung ist erfreulich, da die verwendeten Experten ja stark vom Zufall bestimmt sind, so daß es schwierig ist, allgemeine Parametereinstellungen zu erhalten. Überhaupt bieten die Resultate der Dx-Läufe bei beiden Teams (2-Opt und 3-Opt) ein tendenziell stark ähnliches Bild.

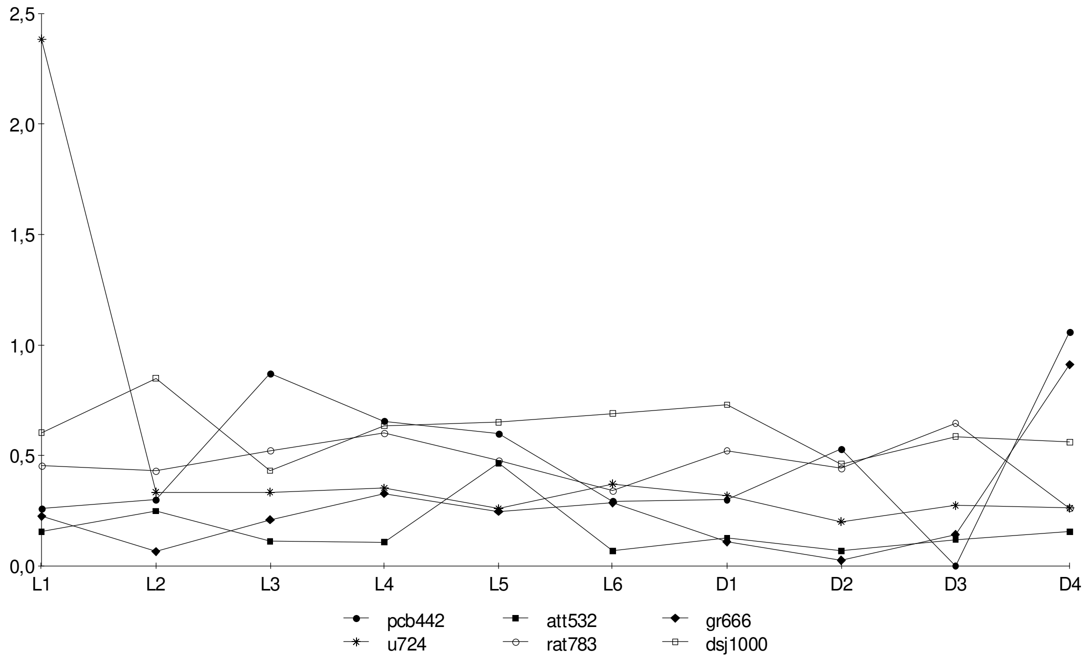

Abb. 5.2.4: Liniendiagramm der prozentualen Abweichungen (2-Opt-Team)

In der expliziten Darstellung (Abb. 5.2.4) der Testergebnisse bestätigen sich die von Abb. 5.2.3 abgeleiteten Vermutungen. Für fast alle Beispiele werden mit der Berücksichtigung der Unterschiedlichkeit von Touren (D1...D4) bessere Ergebnisse erzielt als mit der einfachen Auswahl der kürzesten. Dennoch trifft dies für das TSP dsj1000 nicht zu, zumindestens nicht mit den hier benutzten Parametereinstellungen. Dies zeigt, daß sich sehr schwer ein allgemeines Rezept oder allgemeine Parameterbelegungen beim Traveling Salesman Problem finden lassen, insbesondere wenn die verwendeten Verfahren auf Zufallsmechanismen beruhen.

Vergleich mit bisherigen Teams

Es soll nun analysiert werden, welche Leistungssteigerung im Vergleich zu den bisher in DOTT verwendeten Teams zu beobachten ist. Dabei macht es selbstverständlich nur Sinn, solche Teams zu betrachten bei denen zwei Experten beteiligt sind, um einen fairen Vergleich zu gewährleisten. In Tabelle 5.2.5 sind die erreichten Ergebnisse dargestellt. Wie man leicht sieht, wird für jedes der getesteten Beispiele mit den neuen Teams entweder die optimale Lösung schneller gefunden (kroB100, gr202) oder aber eine deutliche Verbesserung der bisher kürzesten Tour erreicht. Besonders bemerkenswert ist hierbei das Beispiel gr666, bei dem in der gleichen Zeit eine erheblich kürzere Tour gefunden wird. Insgesamt ist also die Leistungsfähigkeit der neuen Experten offensichtlich.

Beispiel	kürzeste Tour bisher		kürzeste Tour jetzt	
	Länge	Zeit	Länge	Zeit
kroB100	22 141*	9.87	22 141*	5.28
gr202	40 160*	43.58	40 160*	21.98
pcb442	50 931	8:58.39	50 778*	22:05.34
att532	27 775	24:09.28	27 703	2h 11:25.30
gr666	296 704	25:52.94	294 435	25:29.50

Tabelle 5.2.5: Vergleich bisheriger Teams mit den neuen 2-Opt/3-Opt-Teams

Vergleich 2-Opt-Team mit 3-Opt-Team

Man sieht aus den Abbildungen 5.2.1 und 5.2.3 schon sehr deutlich, daß das 2-Opt-Team im Schnitt bessere Touren findet als das 3-Opt-Team, obwohl dessen Laufzeit ja wesentlich länger ist. Bestätigt wird dies in Tabelle 5.2.6. Mit der Ausnahme des Beispiels att532 wird mit dem 2-Opt-Team entweder die optimale Lösung gefunden oder eine, die kürzer ist als die kürzeste des 3-Opt-Teams. Trotzdem ist auch bei diesem Ausnahmebeispiel die Qualität der Lösung sehr nahe an derjenigen des 3-Opt-Teams. Besonders auffällig ist aber die wesentlich geringere Laufzeit der 2-Opt-Experten im Gegensatz zu dem 3-Opt-Team. Obwohl 2-Opt nicht so leistungsfähig ist, werden also mit den 2-Opt-Experten bessere Ergebnisse gefunden als mit dem 3-Opt-Team, und diese in geringerer Zeit ! Dies zeigt, daß sich die Idee in der Praxis deutlich bewährt hat, einen schnelleren lokalen Optimierer als 3-Opt zu verwenden. Aufbauend auf diesem Wissen könnte man weitere solcher kombinierten Experten implementieren, die eine schnelle lokale Optimierung benutzen. Darüberhinaus wäre es interessant, in Teamwork die 2-Opt-Experten mit dem Experten LIN2 zu koppeln (zu festen Zeitpunkten die gesamte Population mit LIN2 bearbeiten), um zu untersuchen, ob dadurch eine weitere Verbesserung erzielt wird.

Beispiel	kürzeste Tour 3-Opt-Team		kürzeste Tour 2-Opt-Team	
	Länge	Zeit	Länge	Zeit
kroB100	22 141*	5.89	22 141*	5.28
gr202	40 160*	1:11.40	40 160*	21.98
pcb442	50 785	1h 08:20.87	50 778*	22:05.34
att532	27 703	2h 11:25.30	27 705	15:42.04
gr666	294 522	2h 55:56.31	294 435	25:29.50
u724	41 956	3h 17:55.61	41 994	1h 34:17.40
rat783	8 836	3h 26:18.32	8 829	1h 19:20.33
dsj1000	18 768 891	6h 22:30.42	18 740 224	1h 56:31.66

Tabelle 5.2.6: Vergleich 3-Opt-Team mit 2-Opt-Team

6. Zusammenfassung und Ausblick

Das Ziel des Systems DOTT war es, das Traveling Salesman Problem mit Hilfe von Rechnernetzen zu lösen. Insbesondere sollten die Vorteile vieler verschiedener Lösungsverfahren in einem System mit Hilfe der Teamwork-Methode vereinigt werden, um eine Kooperation dieser Algorithmen zu erreichen. Man kann ruhigen Gewissens von einem beachtlichen Erfolg dieser Idee sprechen, der gewünschte Ansatz wurde sehr gut umgesetzt.

Werden bereits in der initialen Version von DOTT gute Ergebnisse erzielt, so sind inzwischen noch wesentlich leistungsfähigere Experten in das System integriert, so daß eine weitere Verbesserung erreicht wird. Die Kombination von genetischen Verfahren mit lokaler Optimierung in Form der Experten MUT4_LO3, MUT4_LO2, ER_LO3 und ER_LO2 hat den gewünschten Erfolg erzielt, diese Experten sind die leistungsfähigsten überhaupt in dem System. Mit Hilfe dieser Experten ist eine erhebliche Verbesserung gegenüber den reinen genetischen Algorithmen erreicht worden. Insbesondere die Verwendung eines schnellen lokalen Optimierers in Form des erweiterten 2-Opt-Verfahrens weist hervorragende Ergebnisse auf.

Durch die Kooperation der vier neuen Experten in Teamwork sind sehr gute Resultate gefunden worden, es traten sogar Synergieeffekte in den Teamläufen auf. Das wichtigste Ziel, das Auffinden von besseren Touren im Team, sowie das schnellere Finden der besten sequentiellen Tour, ist für die meisten Beispiele erreicht worden. Mit den in dieser Arbeit entwickelten Gutachtern konnte eine weitere Steigerung der Teamleistung im Vergleich zu den bisherigen Gutachtern erreicht werden, für REF_DIVERS konnte eine offenbar sinnvolle, grobe Parametereinstellung gefunden werden. Es ist offensichtlich, daß die Berücksichtigung der Unterschiedlichkeit von Touren ein gutes Werkzeug für das Problem der lokalen Minima darstellt, da der Einsatz von REF_DIVERS zu einer Verbesserung führte. Als nachteilig erweist sich in diesem Zusammenhang die starke Zufallskomponente der vier neuen Experten, da sich ein Testlauf i.a. nicht wiederholen läßt.

Interessant wäre nun eine Kombination des 2-Opt-Teams mit Verbesserungsverfahren wie LIN2; es ist zu untersuchen, ob mit Hilfe eines solchen Teams noch bessere Touren gefunden werden.

Durch die Entwicklung der neuen Gutachter ist das System DOTT seiner Automatisierung wieder ein Stück näher gekommen. Die Siegerwahl der Experten geschieht jetzt unabhängig vom Einwirken des Benutzers und richtet sich tatsächlich nach dem Verlauf der Lösungsprozesses. Um eine vollkommene Benutzerunabhängigkeit zu erhalten, ist die Entwicklung eines automatischen Leiters nötig, der das TSP analysiert, sich dynamisch an die Problemstellung anpaßt und geeignete Teams von Experten und Gutachtern zusammenstellt. Dieser Punkt stellt auch die wichtigste zukünftige Weiterentwicklung von DOTT dar.

Im folgenden sollen einige Ideen zur Realisierung eines automatischen Leiters gegeben werden. Grundsätzlich ist es sicherlich sinnvoll, Kurz- und Langzeitstrategien zu unterscheiden. Während Kurzzeitstrategien nur den nächsten Zyklus oder einige wenige berücksichtigen, kann durch Langzeitstrategien festgestellt werden, ob das System in einer Sackgasse steckt und ein neuer Weg beschritten werden muß. Weiterhin könnte sich ein Teamlauf in mehrere Phasen einteilen. In der ersten Phase könnten Straight-forward-Methoden verwendet werden, wofür man Standardeinstellungen für die Parameter und Teams hat. Diese Phase würde so lange dauern bis keine Verbesserung (für eine feste Anzahl von Zyklen) mehr erzielt wird. Man könnte in dieser ersten Phase beispielsweise kurze Lösungen bevorzugen, um überhaupt erst einmal in einen qualitativ guten Bereich von Touren vorzustoßen. In der darauffolgenden Phase könnte man dann mit anderen Experten herumexperimentieren, und ungewöhnliche Lösungswege beschreiten. In der Schlußphase könnten dann reine Zufallsalgorithmen zum Tragen kommen, falls man mit den bisherigen Methoden keine Verbesserung mehr erzielt. Es stellt sich selbstverständlich die Frage, wann ein geeigneter Abbruchzeitpunkt für solch einen Lösungsprozeß wäre. Darüberhinaus sollte ein automatischer Leiter lernfähig sein, d.h. aus den Testläufen, die er durchgeführt hat, Informationen speichern, die er für spätere Situationen wieder verwendet. Wünschenswert wäre hierbei ein Verfahren zur Klassifizierung von Traveling Salesman Problemen, was sich aber als sehr schwierig erweist. Weitere Ideen zur Entwicklung eines vollautomatischen Leiters finden sich in [Leo95], Abschnitt 6.2.

Anhang A: Literatur

[Ber89] Stefan Berchtold: Optimierung mit genetischen Algorithmen. c't 6/89, S.130-138

[Blü91] Thomas Blümecke: Optimierung mit Evolutionsstrategien und genetischen Algorithmen. c't 12/91, S. 228-239

[De93] Jörg Denzinger: Teamwork: Eine Methode zum Entwurf verteilter, wissensbasierter Theorembeweiser. Dissertation an der Universität Kaiserslautern, Fachbereich Informatik, 1993.

[DK94] Jörg Denzinger, Martin Kronenburg: Planning for distributed theorem proving; The Team Work Approach. SEKI-Report SR-94-09. Universität Kaiserslautern 1994.

[Jo90] D.S. Johnson: Local optimization and the Traveling Salesman Problem. Proc. ALP, LNCS 443, 1990, S. 446-461

[Leo95] Thomas Leopold: Verteilte Lösung des Travelling Salesman Problems durch Teamwork. Diplomarbeit, Universität Kaiserslautern, Fachbereich Informatik, 1995.

[Leo 95-8] Thomas Leopold: DOTT - Distributed Optimization of the Travelling Salesman Problem using Teamwork. Anleitung für die Systementwicklung. Universität Kaiserslautern. August 1995.

[Li93] J.Lind: Sicheres Broadcasting für DISCOUNT. Projektarbeit, Universität Kaiserslautern, Fachbereich Informatik, 1993

[Lin65] S.Lin: Computer solutions of the Traveling Salesman Problem. Bell Syst. Tech. J.44, 1965, S. 2245-2269

[LLRS85] E.L.Lawyer, J.K.Lenstra, A.H.G.Rinnooy Kan, D.B.Shmoys: The Traveling Salesman Problem. John Wiley & Sons, Chichester, 1985

[MadO] Klaus Madlener, Friedrich Otto: Effiziente Algorithmen. Skript zur gleichnamigen Vorlesung, Fachbereich Informatik, Universität Kaiserslautern

[Mi92] Z. Michalewicz: Genetic Algorithms + Data Structures = Evolution Programs. Springer Artificial Intelligence 1992

[Otto94] Thomas Otto: (Simulated Annealing:) Travelling Salesman - eine neue Strategie für eine alte Aufgabe. c't 1/94, S.188-194

[Re91] Gerhard Reinelt: TSPLIB - A Traveling Salesman Problem Library. ORSA Journal on Computing 3, 1991. S. 376-384

[Re94] Gerhard Reinelt: The Traveling Salesman. Computational Solutions for TSP Applications. Springer-Verlag 1994

[Rie93] Christian Rieck: Naturanaloge Verfahren in der Computersimulation. c't 11/93, S. 201-208

[Ye86] D. Yeates: System Project Management. PITMAN Publ. 1986

Anhang B: Verwendete Beispiele

Sämtliche verwendeten Beispiele dieser Arbeit sind der TSPLIB, Version 1.2 entnommen, die in [Re91] genauer beschrieben wird.

Name des TSP	Anzahl der Städte	kürzeste bekannte Lösung
kroB100	100	22 141
gr202	202	40 160
pcb442	442	50 778
att532	532	27 686
gr666	666	294 358
u724	724	41 910
rat783	783	8 806
dsj1000	1000	18 659 688