Jan Hendrik Hansen

Modellbasierte Entscheidungsunterstützung für den Forstbetrieb

Optimierung kurzfristiger Nutzungsoptionen und mittelfristiger Strategien unter Verwendung metaheuristischer Verfahren und parallelen Rechnens

Cuvillier Verlag Göttingen
Internationaler wissenschaftlicher Fachverlag

Modellbasierte Entscheidungsunterstützung für den Forstbetrieb

Modellbasierte Entscheidungsunterstützung für den Forstbetrieb

Optimierung kurzfristiger Nutzungsoptionen und mittelfristiger Strategien unter Verwendung metaheuristischer Verfahren und parallelen Rechnens

Dissertation
Zur Erlangung des Doktorgrades
der Fakultät für Forstwissenschaften und Waldökologie
der Georg-August-Universität Göttingen

vorgelegt von
Jan Hendrik Hansen
geboren in Göttingen

Göttingen, im Juni 2011

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.
1. Aufl. - Göttingen : Cuvillier, 2012
Zugl.: Göttingen, Univ., Diss., 2011

978-3-95404-100-8

1. Gutachter: Prof. Dr. Jürgen Nagel
2. Gutachter: Prof. Dr. Dr. h. c. Klaus von Gadow
Tag der mündlichen Prüfung: Freitag, 5. August 2011

Nonnenstieg 8, 37075 Göttingen
Telefon: 0551-54724-0
Telefax: 0551-54724-21
www.cuvillier.de

1. Auflage, 2012
Gedruckt auf säurefreiem Papier

978-3-95404-100-8

„[] ... der spezifische Praktiker sah jeden mit scheelem Auge an, der es wagte, eigene, in theoretischem Wissen begründete Ansichten zur Geltung zu bringen, und da und dort den dogmatisch von Generationen zu Generationen fortgetragenen Erfahrungsregeln den Krieg zu erklären. Hat ja doch mancher Knasterbart seiner Zeit geringschätzig die Nase gerümpft, als G. L. Hartig in seinem, zuerst im Jahre 1791 erschienen Lehrbuche für Förster, die bis dahin als richtig anerkannten, aber besser gesagt, geglaubten waldbaulichen Regeln systematisch darzustellen, gesucht hat.“

August von Ganghofer (1827-1900) (1877, S. I-II)

Vorwort

Die Grundlage zur Verwirklichung des im Rahmen dieser Arbeit entwickelten Softwaresystems wurde in den ersten zwei Jahren meiner Tätigkeit an der damaligen Niedersächsischen Forstlichen Versuchsanstalt geschaffen. Vor dem Hintergrund verschiedener Holzaufkommensprognosen war es notwendig ein System zu erstellen, welches in annehmbarer Zeit auf Basis von Stichprobendaten unterschiedliche Waldentwicklungsszenarien auf Landes- oder Bundesebene berechnen kann. Da das so entstandene System auch für die forstliche Praxis ein gewisses Potential darstellt, setzte sich Prof. Dr. Jürgen Nagel bei der Deutschen Bundesstiftung Umwelt (DBU) dafür ein, in einem geförderten Projekt dieses Programm weiterentwickeln und an die Bedürfnisse der Praxis anpassen zu können. Die DBU hat daraufhin die Finanzierung des Projektes „WaldPlaner4All“ für zwei Jahre übernommen. Mein besonderer Dank gilt daher der DBU und speziell Herrn Dr. Reinhard Stock für die Unterstützung der vorliegenden Arbeit. Herrn Prof. Dr. Jürgen Nagel danke ich für die Betreuung und das Einräumen aller notwendigen Freiheiten, die das Entstehen dieser Arbeit erst möglich gemacht haben.

Weiterhin möchte ich mich bei Herrn Prof. Dr. Dr. h. c. Klaus von Gadow in zweierlei Hinsicht bedanken. Zum einen hat er bereits während meines Studiums mein Interesse für computergestützte, innovative Verfahren der Forstplanung durch viele interessante Vorlesungen und Übungen geweckt. Zum anderen freut es mich, dass Prof. von Gadow sich noch einmal bereit erklärt hat, die Begutachtung dieser Arbeit zu übernehmen.

Danken möchte ich auch Herrn Prof. Dr. Spellmann, der mich und meine Arbeit an der Nordwestdeutschen Forstlichen Versuchsanstalt immer unterstützt hat. Gleiches gilt auch für alle Mitarbeiter der Abteilung A. Besonders zu erwähnen sind meine Kollegen Dr. Swen Hentschel und Johannes Sutmöller, die mir nicht nur mit fachlichen Tipps und Hinweisen zur Seite standen, sondern auch meine mitunter absonderliche, durch das Doktorandendasein geprägte Art ertragen haben.

Meinen Eltern Renate und Gerd Hansen möchte ich an dieser Stelle für ihren immerwährenden Beistand und die stete Unterstützung danken. Ihnen widme ich diese Arbeit.

Inhalt

1 Einleitung

Das Landschaftsbild Deutschlands zeichnet sich durch einen hohen bewaldeten Flächenanteil aus. Rund 31 Prozent des Landes sind definitionsgemäß Wald. Der mittlere Vorrat von 320 m³/ha ist führend in Europa (BMVEL 2004). Herausragend sind die unverzichtbaren wirtschaftlichen und kulturellen Leistungen des Waldes (OESTEN u. ROEDER 2002), wobei die Anforderungen und Erwartungen an den Wald einem steten Wandel unterliegen. Die heute bestehenden Wälder sind stark durch die seit ca. 200 Jahren betriebene, zielgerichtete Forstwirtschaft geprägt. Nur wenige Flächenanteile werden seit mehreren Jahrzehnten bewusst den natürlichen Entwicklungsprozessen überlassen (MEYER et al. 2006).

Die Planung der Behandlung der einzelnen Waldflächen wird im Rahmen der Forsteinrichtung durchgeführt. Die klassische Forsteinrichtung beruhte lange auf der Festlegung verbindlicher Vorgaben zur Bestandesbehandlung, welche auf der Grundlage langfristiger Zielvorstellungen abgeleitet wurden (TREMER 2008). Die Planung und Durchführung von Einschlägen und Pflegemaßnahmen sowie das Management von Naturschutzmaßnahmen sind jedoch eine zunehmend komplexe Aufgabe, welche stark von der Dynamik der politischen, ökonomischen und natürlichen Rahmenbedingungen beeinflusst wird. Neben den möglichen Auswirkungen des Klimawandels auf die Entwicklung des Waldes ist der Trend zu beobachten, dass die gleiche Waldfläche von immer weniger Personal betreut und bewirtschaftet wird (MERKER 2006, HINRICHS 2006a). Hinzu kommt, dass die gesellschaftlichen, wirtschaftlichen und politischen Ansprüche an den Wald sich ständig ändern und untereinander konkurrieren. Dies erschwert zusätzlich die planerischen Aufgaben. Neben Aspekten wie der Wasserspende und der Erhaltung der biologischen Diversität oder der Funktionen des Waldes zur Erholung und Freizeitgestaltung stehen die wirtschaftliche Nutzung und eine steigende industrielle Nachfrage nach Rohholz. Zusätzlich zum Rohholzbedarf in den traditionellen, holzverarbeitenden Industriezweigen, wie der Säge- und Papierindustrie und der Holzwerkstoffindustrie mit ständig weiterentwickelten Verfahren zur Verbesserung der Holzeigenschaften und einer Ausweitung des Einsatzfeldes des Werkstoffs Holz, ist ein steigender Trend bei der Verwendung von Holz als Energieträger zu verzeichnen. Der Anwendungsbereich reicht dabei von Scheitholz für den privaten Bedarf bis hin zur Raffination von Biotreibstoff aus Holz (RÜTHER et al. 2007). Auch die Globalisierung der Rohstoff- und Warenmärkte stellt die deutsche Forst- und Holzwirtschaft vor neue Herausforderungen. Sie hat zu einer steigenden Holznachfrage und zum Aufbau neuer

Produktionskapazitäten der Holzindustrie geführt. Die steigende Bedeutung der Forst- und Holzwirtschaft zeigt sich auch in der Definition des Clusters *Forst und Holz* seitens des Europäischen Parlaments (BUNDESRAT 2001).

Aus der steigenden Anforderung an die Forstplanung und Forstwirtschaft resultiert der Bedarf nach einer detaillierten rationalen Informationsbasis über den Zustand und die zukünftig zu erwartende Entwicklung der Waldflächen. Diese Informationen können z. T. durch Inventuren wie beispielsweise die Bundeswaldinventur gedeckt werden. Die Bundeswaldinventur wurde bislang zweimal durchgeführt und ein dritter Durchlauf wird zurzeit geplant (BMVEL 2004). Auf der Bundeswaldinventur aufbauend wurden mehrere Untersuchungen des Clusters *Forst und Holz* durchgeführt (Nordrhein-Westfalen (SCHULTE 2003), Bayern (LWF 2005), Niedersachsen (RÜTHER et al. 2007), Sachsen-Anhalt (RÜTHER et al. 2008b), Schleswig-Holstein (RÜTHER et al. 2008a)). Diese Studien haben unter anderem das Ziel, unter den veränderten Rahmenbedingungen der Forst- und Holzwirtschaft Rohholzpotentiale abzuschätzen und eine betriebliche aber auch politische Diskussions- und Entscheidungsgrundlage zur Verfügung zu stellen. Auf Landesebene stellte beispielsweise Sander die Entwicklung und Zielsetzungen eines Informationssystems zur Betriebssteuerung in Niedersachsen vor (SANDER 2000). Dabei wird auf den klassischen Data-Warehouse-Ansatz verwiesen, der auf dem Zusammenführen von Daten aus unterschiedlichen Quellen in einer unternehmensweiten Datenbank mit integriertem Datenmodell beruht. Datenquellen sind neben Forsteinrichtungsdaten die Vollzugsdaten, welche durch einen Abgleich mit den Solldaten den aktuellen Waldzustand beschreiben. Ein erhöhter Informationsbedarf besteht gerade auch seitens mittelständischer und kleiner Forstbetriebe. In Deutschland sind ca. 29.000 forstliche Betriebe registriert, die mit unterschiedlichsten Interessen und Zielsetzungen agieren (MROSEK et al. 2005). Der Informationsbedarf für die Betriebsführung und die damit verbundenen Planungsaufgaben übersteigt oft jedoch den Informationsgehalt, der durch Inventuren oder Forsteinrichtungsdaten und unter Zuhilfenahme klassischer Planungsinstrumente (Ertragstafeln) direkt verfügbar ist. Aus diesem Informations- und Instrumentendefizit ergab sich bereits in den 1990er Jahren der Bedarf einer Neuausrichtung der klassischen Forstplanung (FRANZ 1987, GADOW 1991). Stehen zusätzliche Informationen und bessere Planungswerkzeuge zur Verfügung, können entsprechend qualitativ hochwertigere Entscheidungen getroffen werden. Der zusätzliche Informationsbedarf kann z. B. auf Basis computergestützter Waldwachstumssimulatoren bereitgestellt werden. Auf den Inventur- bzw. Forsteinrichtungsdaten aufbauend, können diese Systeme zusätzliche

Informationen generieren und den benötigten Informationsbedarf, vor allem bezüglich zukünftiger Auswirkungen verschiedener Maßnahmen z. T. abdecken. Im Allgemeinen können computergestützte Systeme in allen Funktionsbereichen und Betriebsebenen zur Unterstützung strategischer und operativer Aufgaben eingesetzt werden. Zur Entwicklung solcher Anwendungssysteme existieren diverse Ansätze, die meist als Informations-, Planungs- und Entscheidungsunterstützungssysteme bezeichnet werden. Programme, welche dispositive bzw. leitende Tätigkeiten im Rahmen von Planungs- und Entscheidungsprozessen unterstützen, werden überwiegend als Management-Unterstützungs-Systeme (MUS) oder Management-Support-Systems (MSS) bezeichnet (DAVIS 1997).

Im Rahmen der Forstplanung kommen bereits Waldwachstumssimulatoren zum Einsatz (BÖCKMANN 2004, HANSEN 2006, HASENAUER 2006). Die Simulatoren haben den entscheidenden Vorteil gegenüber den klassischen Ertragstafeln, dass sie komplexe Bestandesstrukturen und vor allem Mischungsformen sowie verschiedene Durchforstungs- und Endnutzugsstrategien abbilden und unter Berücksichtigung aktueller Wachstumsgänge Zuwächse abschätzen können (NAGEL et al. 2006, TEUFEL et al. 2006). Darauf aufbauend können mit Eingriffsmodellen verschiedene Eingriffsarten und deren Auswirkungen auf einen Bestand abgebildet werden. Die Simulatoren sind in der Regel in Form von Einzelplatzanwendungen implementiert und sind auf die Analyse und Simulation einzelner Bestände ausgerichtet. Systeme zur Verarbeitung aller Bestände eines Forstbetriebs mit integrierten Verfahren zur Ableitung von Handlungsempfehlungen in Verbindung mit einer benutzerfreundlichen Oberfläche und annehmbaren Rechenzeiten befinden sich z. Z. noch in der Entwicklungsphase oder berücksichtigen nicht alle entscheidungsrelevanten Parameter (TEUFEL et al. 2006). Es besteht folglich ein Handlungsbedarf für die Entwicklung und Verbesserung entscheidungsorientierter, modellbasierter Planungswerkzeuge (SCHÖLLER u. SPORS 2001).

1.1 Ziele der Arbeit

Um eine multifunktionale Waldwirtschaft und nutzerspezifische Waldbaustrategien effektiv abbilden und modellhaft umsetzen zu können, bedarf es eines Systems, welches aus allen Bereichen der verschiedenen Waldfunktionen (Nutzung, Schutz, Erholung) Indikatoren zum aktuellen Waldzustand aber auch für einen zukünftigen Zeitraum bestandesweise aufbereiten bzw. abschätzen kann. Die Planung forstlicher Maßnahmen unterliegt der Gliederung der Waldflächen in Planungseinheiten bzw. Bestände (Abteilung, Unterabteilung, Unterfläche…)

(GADOW 2003). Die Planung auf Ebene der durch unterschiedliche Standortbedingungen und durch Unterschiede in der historischen Nutzung resultierender Einzelbestände ist mit der Planung auf Waldlandschaftsebene untrennbar verbunden. Die Bestandesentwicklung ist durch eine Abfolge forstlicher Eingriffe und deren Auswirkungen auf das Ökosystem und den Betriebserfolg bestimmt (HINRICHS 2006b). Aus der Anzahl der einzelnen Planungseinheiten und der Vielzahl der verschiedenen Handlungsoptionen (Entwicklungspfade[1]) für jede Planungseinheit resultiert ein oft sehr großer Handlungsraum. Es ist daher sinnvoll, den Handlungsraum systematisch durch ein geeignetes computergestütztes Verfahren nach optimalen Handlungsalternativen durchsuchen zu lassen. Ein solches Verfahren muss waldbauliche Maßnahmen unter Berücksichtigung konkreter Zielvorstellungen bewerten können, um dann mit geeigneten Such- bzw. Optimierungsalgorithmen aus den möglichen Handlungsalternativen eine optimale Auswahl identifizieren zu können.

Das Ziel dieser Arbeit ist es, ein für den nordwestdeutschen Raum gültiges, praktikables Softwaresystem zu entwickeln, welches die genannten Anforderungen erfüllt. Es soll auf Basis verschiedener Eingangsdaten (z. B. Betriebsinventurdaten, Forsteinrichtungsdaten) die Waldflächen eines Forstbetriebs modellhaft abbilden und verschiedene Nachhaltigkeitsindikatoren (CANADIAN COUNCIL OF FOREST MINISTERS 1997, SPELLMANN 2002) ableiten können. Darauf aufbauend sollen waldbauliche Szenarien definiert und simuliert werden, so dass der Anwender im Rahmen eines Variantenstudiums die Auswirkungen verschiedener Szenarien unter Berücksichtigung seiner Zielvorstellungen bewerten kann. Dadurch wird es möglich, eine Aussage darüber zu treffen, welche der Varianten den definierten Zielen am nächsten kommen. Die so ausgewählte Variante ist jedoch mit hoher Wahrscheinlichkeit nicht die bestmögliche Handlungsalternative. Um diese systematisch identifizieren zu können, soll das System des Weiteren die Möglichkeit bieten, die Zielvorstellungen des Nutzers abzufragen und dementsprechend eine optimale Handlungsalternative zu ermitteln. Eine Voraussetzung dafür ist, dass das System verschiedene Eingangsdaten (auf Bestandesebene aggregierte Daten, einzelbaumbasierte Daten) verarbeiten kann und der benötigte Informationsbedarf auf Daten beschränkt ist, die einem Forstbetrieb standardmäßig zur Verfügung stehen.

[1] Vgl. hierzu das Mehrpfadprinzip (GADOW 2006)

Dem Anwender eröffnen sich dadurch drei Entscheidungsunterstützungsebenen. Die erste Ebene umfasst die Bereitstellung verschiedener, entscheidungsrelevanter Informationen, welche auf Basis der Eingangsdaten abgeleitet werden (Standort, Geländesituation, Vorrat, Zuwachs, Artenzusammensetzung, Alter, Pflegedringlichkeit usw.). Diese Informationen zum Status quo (der Aufnahmen) sollen bestandesweise für die jeweilige Planungseinheit (Revier, Betrieb) oder für verschiedene Auswertungsstraten aggregiert in unterschiedlicher Form (tabellarisch, graphisch, kartographisch) abrufbar sein. Dem Entscheidungsträger stehen auf dieser Ebene am wenigsten Informationen zur Verfügung und der Grad der Eigenleistung zur Entscheidungsfindung ist am höchsten.

Die zweite Ebene ermöglicht es ein oder mehrere waldbauliche Szenarien zu definieren und diese für eine festgelegte Zeitspanne zu simulieren. Somit stehen zusätzliche Informationen über die Auswirkungen verschiedener waldbaulicher Handlungsoptionen zur Verfügung, die in Verbindung mit einem Abgleich mit den Zielen des Entscheidungsträgers eine weitere Entscheidungsgrundlage bieten.

Die dritte Ebene ist durch den geringsten Grad an Eigenleistung gekennzeichnet. Auf dieser Ebene sollen mittels der simulationsbasierten Optimierung unter Berücksichtigung nutzerspezifischer Zielvorstellungen optimale Maßnahmenpläne für die folgenden Problemstellungen generiert werden:

1. Optimale Bestandesauswahl zur Bereitstellung definierter Sortimente (z. B. nach einer konkreten Käuferanfrage)
2. Optimale Auswahl von Naturschutzflächen
3. Optimierung der Eingriffsstrategie (räumlich und zeitlich) für einen 10jährigen Einrichtungsturnus

Optimale Bestandesauswahl zur Bereitstellung definierter Sortimente

Das zugrunde liegende Problem beruht darauf, bei Käuferanfragen nach bestimmten Sortimenten die Bestände zu identifizieren, die das oder die Zielsortimente beinhalten. Kann die gewünschte Menge des Zielsortiments von verschiedenen Beständen oder einer Kombination mehrerer Bestände bereitgestellt werden, soll eine optimale Auswahl getroffen werden. Optimal bedeutet dabei, dass zum einen die ausgewählten Bestände möglichst dicht beieinander liegen (Aggregation), um die Ernte- und Bringungskosten gering zu halten. Zum anderen sollen Bestände mit einer möglichst hohen Handlungsdringlichkeit ausgewählt

werden. Dadurch kann der Pflegezustand der ausgewählten Bestände und somit des gesamten Betriebs verbessert und eine mögliche Entwertung hiebsreifer Bestände vermieden werden.

Optimale Auswahl von Naturschutzflächen

Diese Optimierungsvariante soll entscheidungsunterstützend bei der Auswahl von Naturschutzflächen zum Einsatz kommen. Ziel ist es, die betriebsspezifischen, mehrkriteriellen Zielvorstellungen durch eine konkrete Auswahl eines oder mehrerer Bestände bestmöglich zu erreichen. Die Zielvorstellungen beinhalten in diesem Kontext oft gegensätzliche Ziele. Beispielsweise sollen solche Flächen aus der Nutzung genommen werden, die von möglichst geringem wirtschaftlichen Interesse sind aber möglichst mit altem Laubholz bestockt sind, um ausreichende Habitatvoraussetzungen zu schaffen.

Optimierung der Eingriffsstrategie

Bezüglich dieser Problemstellung soll das System den Anwender dabei unterstützen, für eine Dekade (in Anlehnung an die klassische Forsteinrichtung) die Nutzung und Pflege für jeden einzelnen Bestand festzulegen. Es sollen für jeden Bestand des Betriebs oder beispielsweise eines Reviers die optimale Eingriffsstärke und der optimale Zeitpunkt der Maßnahme ermittelt werden. Hierzu wird vom Nutzer durch Gewichtung der Indikatoren *ökonomischer Erfolg*, *konstante Holznutzung*, *Pflegezustand* und *Aggregation der genutzten Bestände* die Zielausrichtung der Optimierung definiert. Als Ergebnis der Optimierung wird eine Nutzungsmatrix generiert, welche z. B. bei der Forstplanung mit einbezogen werden kann.

Ein weiteres Ziel dieser Arbeit ist es, eine möglichst hohe Nutzerfreundlichkeit des Softwaresystems zu erreichen. Der Anwender soll sich intuitiv die Bedienerführung erschließen können und beispielsweise die Einspeisung der Ausgangsdaten ohne Hilfe bewältigen können. Besonders in dem traditionsgeprägten Berufsfeld der Forstwirtschaft ist dieser Aspekt wichtig, um die Akzeptanz der Software zu erhöhen. Weiterhin soll das System möglichst autark lauffähig sein, d. h. es sollen keine weiteren Softwarepakete (z. B. ein externes GIS) vorausgesetzt bzw. installiert werden müssen, um das System anwenden zu können.

Neben dem wachsenden Bedarf an Informationen seitens des Forstbetriebs ist gleichzeitig eine rasante Entwicklung der Rechenleistung von Computersystemen zu beobachten. Eine wesentliche Entwicklung ist die Mehr-Kern-Architektur von Prozessoren. Somit stehen auch an Einzelplatzsystemen mehrere Recheneinheiten und damit enorme

Rechenkapazitäten zur Verfügung. Diese Leistungsfähigkeit kann jedoch erst durch den Einsatz speziell auf den Mehr-Kern-Betrieb ausgelegter Software voll ausgenutzt werden (RAUBER u. RÜNGER 2007, BENZ 2010). Gerade beim Einsatz rechenintensiver Optimierungs- und Simulationsverfahren stellt die Parallelisierung ein erhebliches Potential da. Die Antwortzeiten komplexer Rechenoperationen können deutlich gesenkt und somit die Anwenderfreundlichkeit und Akzeptanz der Software erhöht werden. Die volle Ausnutzung der potentiellen Rechenleistung kann auch dazu beitragen, die Modellauflösung und die Lösungsqualität der Optimierungsprozesse zu erhöhen. Deshalb sollte die Softwarearchitektur so konzipiert werden, dass die rechenintensiven Elemente (Waldwachstumssimulation, Optimierung) in Abhängigkeit vorhandener Hardwaregegebenheiten parallelisiert werden. Zusammenfassend ergeben sich folgende Anforderungen, die das computergestützte Entscheidungsunterstützungssystem erfüllen sollte:

Funktionsumfang

- Verarbeitung von Daten unterschiedlicher Formate und Strukturen (Stichproben, Forsteinrichtungsdaten)
- Verarbeitung und Darstellung raumbezogener Daten (GIS)
- Bereitstellung entscheidungsrelevanter Informationen/Indikatoren
- Möglichkeit zur benutzergesteuerten Simulation von waldbaulichen Szenarien
- Auswertung und Vergleich der Simulationsergebnisse
- Generierung optimaler Maßnahmen unter Berücksichtigung der nutzerspezifischen Präferenzen

Technisch

- Einfache Handhabung (bedienerfreundliches GUI (= Nutzeroberfläche), schnelle Antwortzeiten)
- Anbindung der gängigen Datenbanksysteme (Access, MySQL, PostgreSQL, Oracle)
- Parallelisierbarkeit der rechenintensiven Prozesse (Optimierung/Simulation)
- Autarke Architektur (es werden keine zusätzlichen bzw. kostenpflichtigen Programme benötigt)
- Vom Betriebssystem unabhängige Implementierung

1.2 Gliederung der Arbeit

Die vorliegende Arbeit gliedert sich in vier Teile. Im ersten Teil wird das Konzept des zu erstellenden computergestützten Entscheidungsunterstützungssystems entwickelt. Hierzu wird der generelle Aufbau von Entscheidungsunterstützungssystemen erarbeitet und entsprechend der im vorigen Kapitel genannten Zielvorgaben angepasst. Weiterhin werden benötigte Systemkomponenten und Modelle identifiziert und diesbezüglich der Stand der Forschung dargestellt. Ein Schwerpunkt liegt dabei auf dem Bereich der metaheuristischen Optimierungsverfahren. Es wird die grundsätzliche Funktionsweise dieser Verfahren beschrieben und ein Überblick über die Systematik und Verfahrensauswahl gegeben. Abschließend werden Beispiele konkreter Implementierungen metaheuristischer Verfahren zur Lösung forstlicher Optimierungsprobleme dargestellt.

Der zweite Teil befasst sich mit Modifikationen verschiedener metaheuristischer Verfahren und einem Vergleich der Verfahren hinsichtlich ihrer Lösungsqualität und Lösungsgeschwindigkeit. Auf Basis der Ergebnisse werden die Verfahren identifiziert, welche für den Einsatz in dem zu implementierenden Entscheidungsunterstützungssystem am besten geeignet sind.

Im dritten Abschnitt werden der Aufbau und die Implementierung des gesamten DSS beschrieben. Es wird die Auswahl der benötigten und bereits vorhandenen Komponenten (Wuchsmodell, Maßnahmenmodell) getroffen. Weiterhin werden notwendige Erweiterungen der vorhandenen Modelle sowie neu erstellte Komponenten (z. B. Totholzmodul, GIS-Modul) beschreiben. Ein Schwerpunkt des dritten Teils liegt auf der Entwicklung eines parallelfähigen Simulationssystems, welches die einzelnen Modellkomponenten (Wuchsmodell, Eingriffsmodell, Sortierungsmodell) steuert und mehrere Bestände simultan verarbeiten kann, so dass Szenariorechnungen für große Auswertungsgebiete (z. B. die gesamte Waldfläche eines Betriebs) in vertretbarer Rechenzeit durchgeführt werden können. Aufbauend auf den Ergebnissen der Modellauswahl und der Konzeption des Simulationsmoduls werden die Konzepte zur Lösung der drei konkreten Optimierungsprobleme *Bestandesauswahl zur Bereitstellung definierter Sortimente*, *Optimierung der Nutzung und Pflege* und *Optimierung der mittelfristigen Bestandesbehandlung* erarbeitet.

Im vierten Abschnitt werden die Ergebnisse zu einem konkreten Anwendungsbeispiel vorgestellt und diskutiert. Zudem wird eine kritische Betrachtung des Gesamtsystems vorgenommen.

2 Konzeption des Entscheidungsunterstützungssystems

In diesem Kapitel wird ein Überblick über den generellen Aufbau von Entscheidungsunterstützungssystemen gegeben. Darauf aufbauend werden die einzelnen, für die Implementierung eines waldbaulichen Entscheidungsunterstützungssystems benötigten Komponenten ermittelt und eine entsprechende Konzeption eines vollständigen waldbaulichen Entscheidungsunterstützungssystems vorgestellt.

2.1 Entscheidungsunterstützungssysteme

Das Fällen von rationalen Entscheidungen ist eine wichtige Managementtätigkeit, die sich durch alle Ebenen eines Betriebs zieht. Mitunter wird der Entscheidungsprozess als die zentrale Managementaufgabe gesehen (HEINEN 1976, HEINEN 1991). In diesem Kontext können die Entscheidungsprozesse als Transformationen von Informationen in Aktionen (HOLTEN u. KNACKSTEDT 1997) aufgefasst werden. Informationen stellen daher einen wichtigen Wettbewerbsfaktor dar, so dass Ungleichverteilungen von Informationen zwischen Unternehmen deren Erfolg oder Misserfolg mit beeinflussen können (PICOT u. MAIER 1993). Allerdings muss die Information in einem für den Entscheidungsträger erfassbaren Zustand, als zweckbezogenes Wissen (WITTMANN 1959) vorliegen, um als Grundlage für Entscheidungen und damit auch für konkrete Handlungen zu dienen.

Unter dem Begriff Management-Unterstützungs-System (MSS, Management Support System) werden in neueren Ansätzen alle Informations- und Kommunikationstechnologien zusammengefasst, die Entscheidungsträgern im betrieblichen Kontext als Hilfsmittel zur Problemlösung dienen. Man differenziert zwischen Management-Informations-Systemen (MIS) und Entscheidungs-Unterstützungs-Systemen (DSS, Decision Support System) auf der einen Seite und den Expertensystemen auf der anderen. Mit Expertensystemen wird versucht, das Wissen und die Problemlösungsfähigkeit von Experten unter Zuhilfenahme von Computern zu erfassen und durch die Speicherung und gezielte Auswertung für Anwender mit weniger Erfahrung und Wissen auf dem entsprechenden Fachgebiet zugänglich zu machen.

Entscheidungs-Unterstützungs-Systeme übernehmen die Aufgabe der Informationsaufbereitung durch Selektion, Umwandlung oder Modellierung und bilden eine Komponente der betrieblichen Informationssysteme. Brennan und Elam verstehen unter einem DSS (BRENNAN u. ELAM 1986):

„Decision Support Systems are computer-based systems whose objective is to enable a decision maker to devise high-quality solutions to what are often only partially formulated problems."

Ein DSS ist nach dieser Definition eine Software, die es dem Entscheidungsträger ermöglicht qualitativ hochwertige Problemlösungen zu entwickeln. Geoffrions (GEOFFRION 1983) Definition setzt sechs Eigenschaften voraus, welche ein DSS aufweisen sollte:

- Lösen von un- bzw. semistrukturierten Problemen
- Ausgestattet mit einem leistungsfähigen und leicht bedienbaren User-Interface
- Ermöglicht flexible Kombination von Daten und Anwendungsmodellen
- Kann dem Anwender/Benutzer helfen, einen Lösungsraum (Alternativenraum) zu entwickeln und zu erforschen, d. h. es sollte die Möglichkeit bieten, unter Anwendung geeigneter Modelle eine Serie von möglichen Alternativen zu generieren
- die Vielfalt von Entscheidungsbildungsarten unterstützen und leicht anpassungsfähig sein, erweiterbar, um neue Fähigkeiten zur Verfügung zu stellen, die den Benutzeranforderungen entsprechen
- Bereitstellen interaktiver und rekursiver Lösungsstrategien (-prozesse) mit unterschiedlichen Lösungswegen

Werden in den Entscheidungsprozess räumliche Informationen mit einbezogen, kann ein DSS von einem sog. Spatial Decision Support System (SDSS) unterschieden werden (CZERANKA U. EHLERS 1997).

Fasst man die Definitionen eines DSS aus der Literatur zusammen (z. B. ALTER 1980, BONCZEK et al. 1981, KEEN u. SCOTT-MORTON 1978, SPRAGUE u. CARLSON 1982, BRENNAN u. ELAM 1986) kann ein DSS als ein computergestütztes, interaktives Mensch-Maschine-System zur Entscheidungsfindung beschrieben werden, welches den Entscheidungsträger unterstützt und nicht ersetzt. Hierzu greift ein DSS auf verschiedene Daten/Informationen und Modelle zurück.

Hauptaufgaben eines DSS sind zum einen die Bereitstellung von entscheidungsrelevanten Informationen oder Daten, zum anderen die Unterstützung im Problemlösungsprozess durch entsprechende modellbasierte Planungs- und Entscheidungshilfen.

Üblicherweise besteht ein DSS aus folgenden drei Subsystemen (SPRAGUE u. CARLSON 1982): Dialogmanagement, Datenbankmanagement sowie Modell-, Methoden- und Ergebnismanagement (Abb. 1).

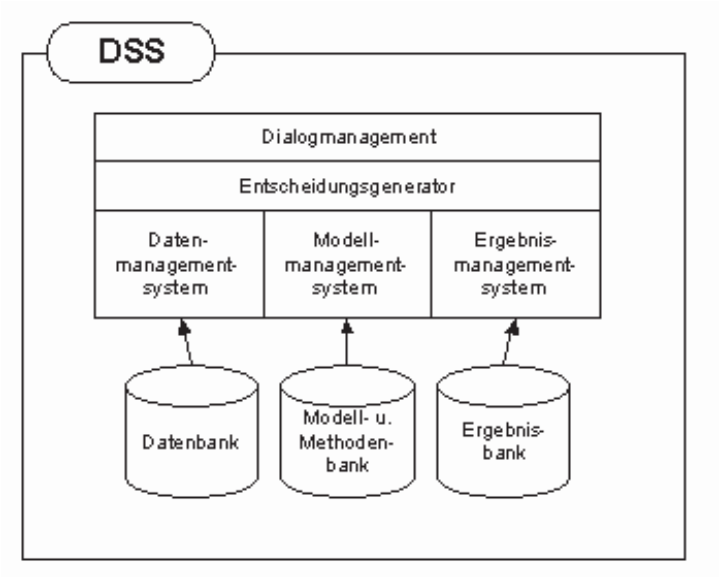


Abb. 1: Subsysteme eines DSS nach SPRAGUE u. CARLSON.

Die Kommunikation zwischen Softwaresystem und Anwender sowie die Ablaufsteuerung und Koordination der einzelnen Komponenten wird über die Dialogkomponente abgewickelt. Das Methodenbankmanagement unterstützt die Definition und Manipulation von problemspezifischen Modellen, die das Kernstück eines Decision Support Systems bilden. Vom Datenbankmanagement werden unabhängig von Modellen und Methoden die Objektdaten gespeichert und verwaltet.

Eine wichtige Aufgabe von Decision Support Systemen (DSS) ist es, die Auswirkungen verschiedener Handlungsoptionen und einzelner Maßnahmen auf das Untersuchungssystem transparent darzustellen. Die Entscheidungsfindung kann durch ein Variantenstudium unterschiedlicher Entwicklungspfade oder durch geeignete Optimierungsverfahren unterstützt werden. Für letztere bietet sich die simulationsbasierte Optimierung an.

Der Grundgedanke der simulationsbasierten Optimierung (SBO) beruht darauf, geeignete Simulationsmodelle mit einer Optimierungskomponente zu kombinieren (GOSAVI

2003). Die Optimierungskomponente variiert definierte Variablen eines Simulationsmodells solange, bis das Maximum oder Minimum einer definierten Zielfunktion gefunden wird. Simulationsmodelle dienen zur Prognose komplexer, realer Systeme, die zufälligen Einflüssen unterliegen. Oft werden in gängigen Decision Support Systemen Simulationsmodelle genutzt, um die Auswirkungen einzelner Handlungsalternativen zu untersuchen, ohne diese tatsächlich umzusetzen. So können mögliche negative Auswirkungen auf das reale System vermieden werden. Die Anzahl der dabei verwendeten Handlungsalternativen ist bei diesem Vorgehen relativ gering. Zwar wird die Simulation computergestützt durchgeführt, die Auswahl dann jedoch zum größtenteils durch einen humanen Entscheidungsträger getroffen. Die simulationsbasierte Optimierung beruht hingegen darauf, die Auswahl der „besten“ Handlungsalternative systematisch durch einen Algorithmus durchzuführen, der wiederum auf ein bestimmtes Simulationsmodell zurückgreift. Die verschiedenen Handlungsalternativen werden über die Modellvariablen beschrieben. Im Verlauf der Optimumsuche werden diese Variablen durch einen geeigneten Algorithmus variiert, die Auswirkungen auf das System bewertet und mit vorhergehenden Lösungen verglichen. Aus den dadurch gewonnenen Erkenntnissen wird entsprechend des gewählten Optimierungsverfahrens eine neue Variablenkombination ermittelt. Dieses Vorgehen wird solange wiederholt, bis verschiedene Abbruchkriterien darauf hindeuten, dass die bestmögliche oder eine vergleichbar gute Lösung gefunden wurde. Ein wesentlicher Vorteil dieses Vorgehens gegenüber den klassischen Optimierungsverfahren (z. B. Lineare Programmierung) besteht darin, dass das verwendete Modell keine besonderen Voraussetzungen erfüllen muss. Es können Optima für komplexe Modelle gefunden werden, zu denen das Optimierungsproblem nichtlinear, nichtdifferenzierbar oder nichtstetig ist. Auch können solche Probleme gelöst werden, zu denen keine geschlossene Zielfunktion vorliegt, so dass eine Optimabestimmung (z. B. durch Ableitung) auf mathematischem Wege nicht möglich ist.

2.2 Benötigte Modelle und Systemkomponenten

Im Zuge der Planung waldbaulicher Maßnahmen müssen komplexe Entscheidungen darüber getroffen werden, wie und wann die einzelnen Planungseinheiten (Bestände) eines Betriebs hinsichtlich der Pflege, Nutzung und Verjüngung zu behandeln sind. Oft sind dabei mehrere mögliche Alternativen gegeben, die durch Rahmenbedingungen mehr oder weniger stark eingeschränkt werden. Für den Entscheidungsträger ergibt sich in Folge dessen das Problem

jene Alternative auszuwählen, die die betrieblichen Ziele bestmöglich verwirklicht. Es liegt eine klassische Entscheidungssituation vor, die sich aus den Zielen und dem Entscheidungsfeld zusammensetzt. Das Entscheidungsfeld wiederum wird durch die möglichen Handlungsalternativen und durch nicht vom Betrieb beeinflussbare Rahmenbedingungen geprägt (OESTEN u. ROEDER 2002). Vor dem Hintergrund einer nachhaltigen, multifunktionalen Bewirtschaftung sind die herkömmlichen Verfahren der Forsteinrichtung hinsichtlich ihrer Eignung zu hinterfragen (SODTKE et al. 2004).

Das im Rahmen dieser Arbeit entwickelte Softwaresystem soll Entscheidungsträger eines forstlichen Betriebs bei der kurz- und mittelfristigen Planung waldbaulicher Maßnahmen unterstützen. Dazu müssen automatisiert Informationen aus verschiedenen Datenquellen aufbereitet, verwaltet und ausgewertet werden können. Darüber hinaus sollen auf Basis dieser Daten waldbauliche Szenarien unter Zuhilfenahme geeigneter Waldwachstums- und Maßnahmenmodelle gerechnet und gegenübergestellt werden. Durch die Kombination der verwendeten Modelle mit geeigneten Optimierungs- und Bewertungsverfahren können dem Anwender qualitativ hochwertige Handlungsalternativen aufgezeigt werden.

Da moderne Waldwachstums- und Behandlungsmodelle oft einen hohen Komplexitätsgrad aufweisen, ist ein simulationsbasierter Optimierungsansatz auf Basis metaheuristischer Verfahren für den Einsatz in einem forstlichen Entscheidungsunterstützungssystem ein geeigneter Ansatz. Basis dieses Systems mit der Möglichkeit zur Generierung optimaler Maßnahmenfolgen zur Erreichung definierter Ziele, ist ein Modell, welches den Zustand der jeweiligen Waldflächen in geeigneter Form erfassen und das Wachstum beschreiben kann. Dies kann in Form eines Einzelbaumwuchsmodells geschehen. Denkbar sind aber auch aggregierende Ansätze, die einen Bestand über Baumschichten und deren Kenngrößen (dg, hg, Alter) beschreiben. Weiterhin wird ein System benötigt, welches waldbauliche Maßnahmen wie Nutzungen, Pflanzungen und Naturschutzmaßnahmen modellhaft abbilden kann. Um die für den Entscheidungsträger optimale Lösung zu finden, müssen die verschiedenen Varianten bzw. Lösungsvorschläge bewertet werden. Hierzu bedarf es eines Verfahrens, welches objektiv die Zielvorstellung der Entscheidungsträger erfasst und in eine systemverständliche Zielgröße übersetzt (ALBERT 2007). Die Forstwirtschaft vieler Betriebe ist zunehmend nicht nur auf die Holzproduktion ausgerichtet, sondern berücksichtigt eine große Bandbreite an unterschiedlichen

Waldfunktionen (PUKKALA 2006). Somit existieren z. T. mehrere Ziele mit daraus resultierenden Zielkonflikten. Dies führt häufig zu Unsicherheit der Entscheidungsträger. Die resultierende komplexe Planungsaufgabe ist selbst durch qualifizierte Fachleute kaum mehr ohne angemessene Entscheidungsunterstützungssysteme zu leisten (TEUFEL et al. 2006). Bei der Optimierung betrieblicher Prozesse ist zunächst zu evaluieren, wie die Ziele des Betriebs definiert sind. In Bezug auf den Wald und die Behandlung der nutzbaren Flächen ist dabei vorab der Planungszeitraum abzugrenzen. Soll im Rahmen der Optimierung kurzfristig, mittelfristig oder langfristig nach einer Lösung gesucht werden? Besonders bei der lang- und mittelfristigen Planung ist zu klären, welche Zielsetzung der Forstbetrieb verfolgt und wie dementsprechend die Zielfunktion konstruiert werden muss. Im einfachsten Fall besteht die Zielfunktion lediglich aus einem Indikator, welcher zu minimieren oder zu maximieren ist. Oft ist es aber erforderlich, die Zielfunktion aus mehreren Indikatoren zusammenzusetzen. Dies führt wiederum zu einem Gewichtungsproblem. Um einen Gesamtnutzen ableiten zu können, muss ermittelt werden, wie die einzelnen Indikatoren zu gewichten sind und wie sie verknüpft werden. Die kurzfristige Optimierung hingegen bezieht sich zumeist auf verfahrenstechnische Probleme, die nicht so stark vom jeweiligen Leitbild des Betriebs abhängen. Auch die festzulegenden Restriktionen werden zum Teil von den betrieblichen Zielen bestimmt. Bezüglich solcher multikriterieller Problemstellungen liegen bereits Ansätze zur programmgestützten Ermittlung der jeweiligen Gewichtung einzelner Indikatoren vor (ALBERT 2007).

Zum Aufbau eines forstlichen Entscheidungsunterstützungssystems werden somit ein Wuchsmodell, ein Maßnahmenmodell, ein System zur multikriteriellen Bewertung und ein geeignetes Optimierungsverfahren benötigt. Diese Komponenten werden u. a. auch von Sodtke et al. (SODTKE et al. 2004) als grundlegende Bausteine eines forstlichen DSS definiert. Um auf Basis dieser vier Komponenten ein praktikables Softwaresystem aufzubauen, werden diese in den in Kap. 2.1 beschriebenen allgemeinen Aufbau eines DSS integriert (Abb. 2). Entsprechend dieses Aufbaus muss neben den Modellkomponenten ein übergeordnetes Modellmanagement implementiert werden, welches die Steuerung und Koordination der verschiedenen Teilmodelle übernimmt, um somit automatisiert verschiedene waldbauliche Szenarien simulieren zu können. Die Ergebnisse der Szenariorechnungen werden vom Ergebnismanagement transformiert und in einer für den Nutzer leicht verständlichen und ansprechenden Form dargestellt. Hierzu werden verschiedene Indizes berechnet und in tabellarischer Form, als Graphik und/oder Karte aufbereitet. Die berechneten Indikatoren

sollen dabei möglichst alle Bereiche der wichtigsten Waldfunktionen abdecken, da so erst die Bewertung waldbaulicher Optionen unter Verwendung einer mehrkriteriell ausgerichteten Zielfunktion möglich ist. Ein Bindeglied zwischen dem Modell- und dem Ergebnismanagement stellt die GIS-Komponente dar. Diese Komponente dient beiden Systembausteinen zur raumbezogenen Aufbereitung bzw. Auswertung verschiedener Modell- oder Ergebnisparameter.

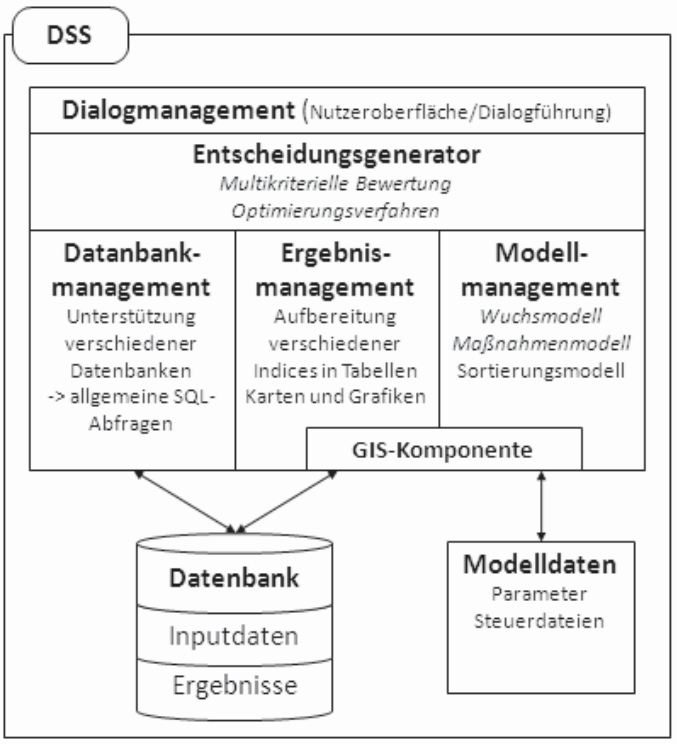


Abb. 2: Konzept zum Aufbau eines forstlichen DSS.

Die Simulationsergebnisse können an den Entscheidungsgenerator weitergegeben werden, welcher die verschiedenen Szenarien nach einer nutzerspezifischen Zielfunktion bewertet oder unter Verwendung eines geeigneten Optimierungsverfahrens unterschiedliche waldbauliche Fragestellungen optimiert. Da gerade bei der simulationsbasierten Optimierung oder bei der Simulation verschiedener waldbaulicher Szenarien im Rahmen eines Variantenstudiums je nach Betriebsgröße und Auflösung der verwendenden Modelle umfangreiche Datenmengen anfallen können, ist die Anbindung des Entscheidungs-Unterstützungs-Systems an eine geeignete Datenbank notwendig. So können schnelle Zugriffszeiten und eine entsprechende Datensicherheit garantiert werden. Die Schnittstelle zur Datenbank wird vom Datenbankmanagement verwaltet. Dieses steuert darüber hinaus das Einlesen von Inputdaten sowie das Speichern und Einlesen von Ergebnisdaten.

2.2.1 Wuchsmodelle

Schon früh in der Geschichte der Forstwissenschaften wurde erkannt, dass zur Unterstützung der waldbaulichen Planung und Praxis Kenntnisse über Zusammenhänge des Waldwachstums von Bedeutung sind. Einer großen Verbreitung und Akzeptanz erfreuten sich z. B. die klassischen Ertragstafeln (z. B. WEISE 1880, WIEDEMANN 1949, DITTMAR et al. 1986, SCHOBER 1995). Diese beinhalten die wichtigsten Bestandeskennwerte von Reinbeständen bei definierter Behandlung in festen zeitlichen (meist fünfjährigen) Intervallen in tabellarischer Form. Sie sind anhand der Beobachtungen auf speziellen, ertragskundlichen Versuchsflächen hergeleitet worden und stellen einfach zu handhabende Modelle zur Beschreibung des Wachstumsgangs der wichtigsten Baumarten dar.

Durch die rapide steigende Leistung von Computersystemen und eine ständig wachsende waldwachstumskundliche Datengrundlage war es möglich, komplexere Modelle zu entwickeln, die mittlerweile auch in der forstlichen Praxis zum Einsatz kommen (NAGEL 1999, PRETZSCH 1992, PUKKALA 1987). Mit Hilfe dieser Modelle lassen sich Szenariosimulationen rechnen, welche als ein ergänzendes Instrument in der strategischen Waldbauplanung zum Einsatz kommen können (SPELLMANN 1998, HANEWINKEL 2001, CHEN et al. 2002, DÖBBELER u. SPELLMANN 2002, PRETZSCH 2002, NAGEL et al. 2006).

In der Regel handelt es sich dabei um einzelbaumorientierte Wuchsmodelle mit einer mehr oder weniger starken Standortsensitivität. Diese Modelle sind eine notwendige Weiterentwicklung der rein deskriptiven Ansätze wie den Ertragstafeln, die vor dem Hintergrund sich ständig ändernder, das Wachstum der Bäume maßgeblich beeinflussender Rahmenbedingungen (Stoffeinträge, Niederschläge usw.) und des steigenden Anteils von Mischbeständen sowie verschiedenster Waldbauverfahren an ihre Grenzen stoßen (GADOW 2006). So sind beispielsweise die beschriebenen Zuwächse, Stammzahlen und Durchmesser oft nicht mehr auf heutige Bestände übertragbar. Im Rahmen der Waldwachstumsforschung werden die computergestützten Modelle ständig verbessert und um Modellkomponenten erweitert oder es wird die Modellgültigkeit ausgeweitet (SCHMIDT u. HANSEN 2007). Diese Systeme fördern das allgemeine Verständnis der Wuchsdynamik (PRETZSCH 2001a) und stellen ein wichtiges Hilfsmittel für die Forschung und die forstliche Praxis dar.

Je nach Einsatzfeld sind neben den Einzelbaummodellen die verschiedensten Modellansätze mit unterschiedlicher räumlicher und zeitlicher Auflösung vorhanden. Bedingt durch Zielsetzung und Kenntnissen über das abgebildete System variiert der

Komplexitätsgrad. In Abhängigkeit von den Dimensionen Zeit und Raum können verschiedene Modellansätze unterschieden werden. Die höchste zeitliche und räumliche Auflösung weisen die ökophysiologischen Prozessmodelle gefolgt von den Einzelbaummodellen auf. Die zeitliche und räumliche Aggregation steigt über die Bestandesmodelle und Sukzessionsmodelle bis hin zu den Biommodellen an (PRETZSCH 2001b). Für die Forstwirtschaft und Forstwissenschaft sind solche Modelle von besonderer Bedeutung, bei der ein Waldbestand die betrachtete Systemeinheit bildet und die Systemzustandsgrößen mindestens bis zur Bestandesebene integriert sind und somit relevante Variablen für Praxis und Wissenschaft liefern. Pretzsch gibt einen umfassenden und systematischen Überblick über diese Modelltypen und Umsetzungen. Hierzu werden fünf Kategorien gebildet (PRETZSCH 2001b):

- Ertragstafeln/Bestandeswachstumssimulatoren mit der Auflösung des Bestandes (Summen u. Mittelwerte)
- Bestandesmodelle auf der Grundlage von Stammzahlhäufigkeiten
- Einzelbaumorientierte Managementmodelle für Rein- und Mischbestände
- Sukzessionsmodelle auf der Basis von Kleinflächen und Biomen
- Ökophysiologisch basierte Prozessmodelle auf der Basis von Stoff- und Energieflüssen

Für den hier vorgestellten Ansatz sind vor allem die einzelbaumorientierten Modelle von Interesse, da sie eine ausreichende Auflösung bieten, um Wachstum und Behandlungskonzepte sowie die Wechselwirkung von Behandlung und Zuwachs oder Mortalität für Rein- und Mischbestände abzubilden. Darüber hinaus können auf Basis dieser Modelle eine Vielzahl relevanter Indikatoren (Diversität, Sozioökonomie, Produktion etc.) abgeleitet werden. Zu den bekanntesten, europäischen einzelbaumorientierten Wuchsmodellen zählen BWINPro (bzw. TreeGrOSS) (NAGEL 2005, NAGEL 2009), MOSES 3.0 (HASENAUER et al. 2006), PrognAus 2.2 (LEDERMANN 2006), SILVA (PRETZSCH et al. 2006), STAND (PUKKALA u. MIINA 2006) und SIBYLA (FABRIKA u. DURSKY 2006).

2.2.2 Behandlungsmodelle

Eine wichtige Komponente für die Konstruktion eines forstlichen DSS ist die modellhafte Umsetzung von waldbaulichen Maßnahmen. Erst über die Abbildung von Bestandesbehandlungen im Modellsystem können die Auswirkungen unterschiedlicher Eingriffe und die Bestimmung eines optimalen Eingriffes realisiert werden (HASENAUER

2004). Voraussetzung ist, dass das verwendete Wuchsmodell sensitiv auf einen simulierten Eingriff reagieren kann und eine Rückkopplung zum Zuwachs und zur Mortalität besteht. Da im Rahmen des Variantenstudiums und des Optimierungsprozesses die modellierten Eingriffe hinsichtlich vieler Parameter wie z. B. Nutzungsmassen, Zielstärke und Durchforstungsbeginn variiert werden müssen, sollte das verwendete Behandlungsmodell möglichst flexibel hinsichtlich der Steuergrößen gestaltet sein. Module zur Simulation forstlicher Eingriffe können grundsätzlich nach drei Kategorien klassifiziert werden (STERBA et al. 2006): (1) nach der Herleitungsmethode, (2) nach dem forstlichen Eingriffstyp und (3) nach der Art und Weise der Implementierung. Die Entwicklung der Modelle kann zum einen empirisch und zum anderen analytisch fundiert sein. Empirische Modelle schätzen die Erntewahrscheinlichkeit einzelner Bäume in Abhängigkeit einer oder mehrerer Variablen. Die Modelle hierzu werden auf Basis empirischer Daten wie Waldinventuren oder Auszeichnungsversuchen parametrisiert. Die analytischen Ansätze hingegen definieren Regelsysteme nach den Grundsätzen gängiger Strategien, welche entscheiden, ob ein Baum entnommen oder im Bestand belassen wird.

Einen Überblick über verschiedene Nutzungsalgorithmen geben Söderberg und Ledermann (SÖDERBERG u. LEDERMANN 2003). Einen umfassenden Ansatz stellt Duda (DUDA 2006) vor. Zur computergestützten Beurteilung verschiedener Managementstrategien (LÖWE, Ertrag, Prozessschutz, Förderung der potentiellen natürlichen Vegetation (PNV)) werden basierend auf dem Einzelbaumwuchssimulator BWINPro mehrere Maßnahmenelemente zur Bestandesbehandlung definiert und implementiert, welche durch Zusammenschalten zu einer sog. Maßnahmenkette die automatisierte Umsetzung verschiedener Strategien ermöglichen. Die Steuerparameter zu den einzelnen Elementen sind in einer Metadatenbank hinterlegt, so dass die definierten Strategien abgeändert oder neue Ansätze hinzugefügt werden können. Die einzelnen Maßnahmenelemente können den Kategorien Schutz (Minderheitenschutz, Habitatbaumauswahl), Endnutzung (Zielstärkennutzung, Entnahme einer erntereifen Schicht etc.), Durchforstung (Z-Baum-Auswahl, Freistellung von Z-Bäumen etc.) und Verjüngung (Pflanzung) zugeordnet werden.

2.2.3 Optimierungsverfahren

Die bestmögliche Gestaltung betrieblicher Prozesse (Produktion, Logistik etc.) bedarf einer sorgfältigen Planung. Im Rahmen des Planungsprozesses müssen geeignete Maßnahmen zur Erreichung des definierten Ziels ermittelt werden (KLEIN u. SCHOLL 2004). Dabei sind

basierend auf einem Ist-Zustand gültige Handlungsalternativen zu ermitteln. Oft sind die zugrunde liegenden Planungsprobleme so komplex, dass ohne geeignete Hilfsmittel die Handlungsalternativen nicht identifiziert werden können und/oder die Anzahl der Handlungsalternativen so groß ist, dass eine Auswahl einer guten oder optimalen Alternative durch humane Entscheidungsträger nicht möglich ist. Die bei der Ermittlung von Handlungsalternativen unterstellten Wirkungszusammenhänge (Modelle) beeinflussen dabei maßgeblich, wie der Zielerreichungsgrad ausfällt. In Abhängigkeit von unterstellten Wirkungszusammenhängen führt die Entscheidung für bestimmte Handlungsalternativen hinsichtlich der unterstellten Präferenzfunktion zu unterschiedlichen Zielerreichungsgraden. Komplexe Optimierungsprobleme, wie z. B. die Maximierung des Ertrags eines forstlichen Betriebs erfordern den Einsatz von Simulationsrechnungen (Zuwachs, Mortalität und waldbauliche Maßnahmen) zur Bewertung von Handlungsalternativen. Je realitätsnäher die Modelle konstruiert werden, umso komplexer wird das Optimierungsproblem, da nicht-lineare Wirkungszusammenhänge oder diskrete Entscheidungsvariablen zu NP-schweren[2] Planungsmodellen führen können (GAREY u. JOHNSON 1979). Im Kontext des Planungsprozesses zu einem klar abgegrenzten Problem sind somit der Ausgangszustand (Datengrundlage), mögliche Handlungsalternativen (Entscheidungsvariablen), Wirkungszusammenhänge sowie die Zielsetzung zu bestimmen. Die Datengrundlage spielt dabei eine entscheidende Rolle, da der Einsatz möglicher Modelle mit einer entsprechenden Auflösung vor allem von der Struktur der Ausgangsdaten abhängt. Die Entscheidungsvariablen fallen im Kontext forstlicher Optimierungsprobleme in die Bereiche Nutzung bzw. Pflege, Bestandesbegründung und Naturschutzmaßnahmen (Habitatbäume, Totholz …).

Dabei sind die Optimierungsprobleme meist kombinatorischer Natur, so dass Handlungsalternativen wie etwa Zuordnungs-, Auswahl- und Reihenfolgeentscheidungen im Mittelpunkt stehen. Im Rahmen der kombinatorischen Optimierung geht es darum, aus einer großen Menge von diskreten Elementen eine Teilmenge zu ermitteln, die gewisse

[2] NP-Schwere (NP-hard = Non-deterministic Polynomial-time hard) ist ein Begriff aus der theoretischen Informatik und dient zur Klassifizierung von Komplexitätsproblemen. Für Such- oder Optimierungsprobleme wurde der Begriff der NP-Äquivalenz eingeführt. Diese liefert eine prinzipielle Aussage darüber, ob Suchprobleme mit wachsender Größe des Suchraums noch praktisch lösbar sind, oder ob die benötigten Rechenressourcen nicht zur Verfügung stehen (WANKA 2006).

Restriktionen nicht verletzt und eine definierte Zielfunktion optimiert (maximiert oder minimiert) oder jedem Element eine bestimmte Behandlung zuzuweisen, so dass wiederum eine Zielfunktion optimiert wird. Eine gute Basis für die Lösung waldbaulicher Optimierungsprobleme ist das sog. Mehrpfadprinzip (GADOW 2006, HINRICHS 2006a, GADOW u. PUKKALA 2008). Dieser Ansatz basiert auf der Annahme, dass jeder Bestand unterschiedliche Handlungspfade (Entwicklungen) durchlaufen kann. Der optimale Entwicklungsplan für einen Forstbetrieb ist durch die Pfad-Kombination gegeben, bei der auf Ebene der Einzelbestände und des Gesamtbetriebs die definierten Ziele simultan optimal erfüllt werden.

Zur Lösung einer solchen Problemstruktur sind Optimierungsverfahren erforderlich, welche die Auswahl aus der Menge der Handlungsalternativen automatisch vornehmen oder zumindest die Auswahl unterstützen. An dieser Stelle könnte die Frage aufkommen, warum nicht alle Pfadkombinationen berechnet/simuliert werden und die Lösung ausgewählt wird, welche die gegebene Zielfunktion optimiert (vollständige Enumeration). Da die Anzahl möglicher Kombinationen in den meisten Fällen exponentiell mit der Anzahl der einzelnen Planungseinheiten (Bestände) wächst (Gl. 1), ist die vollständige Enumeration komplexer Probleme gar nicht oder nur mit sehr großem Rechenaufwand durchführbar. In solchen Fällen scheiden die klassischen Verfahren der mehrkriteriellen Entscheidungsfindung (MCDM) zur Bestimmung der besten Lösung aus (vgl. Kap. 2.2.4).

$$n=p^u$$ *Gl. 1*

wobei:
n= Anzahl Pfadkombinationen
p= Anzahl möglicher Alternativen je Bestand/Behandlungseinheit
u= Anzahl Behandlungseinheiten

Beispielsweise müssten für einen Betrieb mit lediglich 10 Beständen, für welche jeweils 4 Behandlungsalternativen möglich sind 1.048.576 Varianten berechnet werden. Bei 100 Beständen wären es 4^{100} =1,61E+60. Das Problem ist NP-vollständig (GAREY u. JOHNSON 1979), was bedeutet, dass der Aufwand von exakten Verfahren bzw. Enumerationsverfahren exponentiell mit der Problemgröße (Anzahl Bestände/Parameter) ansteigt. Daher können exakte Verfahren das Problem nicht in akzeptabler Zeit lösen. Hinzu kommt, dass beim Einsatz komplexer Modelle das Optimierungsproblem oft nichtlinear, nichtdifferenzierbar und/oder nichtstetig ist. Es liegt keine geschlossene mathematische Formel der Zielfunktion vor, so dass eine Optimabestimmung auf mathematischem Wege nicht möglich ist. Somit

können exakte Verfahren aufgrund der Komplexität und der Struktur des Problems nicht verwendet werden. Für die Lösung derart strukturierter Probleme haben sich metaheuristische Verfahren etabliert. Besonders für kombinatorische Optimierungsprobleme konnten heuristische Verfahren erfolgreich eingesetzt werden. Heuristische Verfahren werden vor allem benutzt, um in einem System (z. B. Produktionssystem) eine Parameterkonfiguration (z. B. Anzahl Arbeiter/Maschinen, Steuerparameter der Maschinen, Zuweisung von Lagerplatz etc.) zu finden, die eine bestimmte Zielgröße (z. B. die Produktionsrate) optimiert. Ist die Zielgröße eine Funktion einzelner Teilziele (z. B. Holzproduktion und Förderung des Naturschutzes), resultiert daraus zusätzlich das Problem der multikriteriellen Bewertung. Auf diese Thematik wird näher in Kap 2.2.4 eingegangen. Heuristische Standardverfahren sind z. B. Direct Search, Hillclimbing, Genetische Algorithmen, Tabu Search oder Simulated Annealing. In den folgenden Kapiteln wird eine Übersicht zur systematischen Einordnung und der Funktionsweise der wichtigsten Vertreter dieser Verfahren gegeben.

2.2.3.1 Heuristische Verfahren

Der Begriff „Heuristik“ geht auf das griechische Verb heuriskein (ευρισκειν) (finden, entdecken) zurück. Im Sinne der Mathematik werden Heuristiken speziell in der Optimierung den sog. Eröffnungsverfahren zugeordnet. Eröffnungsverfahren sind solche Methoden, die innerhalb kurzer Zeit und ohne großen Aufwand eine zulässige Lösung finden. Die resultierende Basislösung kann durch mehrfaches Anwenden (Iteration) der Heuristik verbessert werden. Mit diesen Verfahren, die meist auf Erfahrungen bezüglich der Problemlösung beruhen, können sehr gute, aber nicht zwingend die tatsächlich optimale Lösung (globales Optimum) für bestimmte Planungsprobleme gefunden werden (PEARL 1984). Bei der Implementierung von Heuristiken werden problembezogenes Expertenwissen und plausible Vorgehensweisen formalisiert und in Quellcode übersetzt. Dabei wird in der Regel ein Optimalitätskriterium zugunsten einer kürzeren Rechenzeit und einer elementaren Implementierung aufgegeben. Beim Entwurf solcher Verfahren liegt das Hauptproblem in der Identifikation einfacher heuristischer Vorgehensweisen, die möglichst gute Lösungen bezüglich vorgegebener Zielkriterien liefern. Heuristiken basieren häufig auf Greedy-Strategien. Im Rahmen dieser „gierigen“ Verfahren werden Lösungen iterativ konstruiert. Eine Eigenart der Verfahren ist es, dass nur der Folgezustand aus der gesamten aktuellen Nachbarschaft ausgewählt wird, der den größten Gewinn bzw. das beste Ergebnis verspricht (CORMEN et al. 2001). In diesem Zusammenhang wird auch die Bezeichnung „Single Pass

Heuristic“ verwendet (FISHER u. RINNOOY KAN 1988), da sich der Lösungsweg wie ein einzelner Pfad durch den Suchraum bewegt.

2.2.3.2 Metaheuristiken

Metaheuristiken sind Algorithmen, also präzise definierte Handlungsvorschriften zur Lösung eines meist kombinatorischen Optimierungsproblems. Sie definieren eine abstrakte Folge von Regeln und Anweisungen zur Kontrolle und Steuerung heuristischer Methoden, die theoretisch auf beliebige Problemstellungen angewandt werden können. Heuristiken sind hingegen problemspezifisch und können meist nur für ein bestimmtes Optimierungsproblem eingesetzt werden. Wird eine Metaheuristik problemspezifisch implementiert, kann der allgemeine Charakter wieder verloren gehen und eine Heuristik resultieren. Der Begriff „Metaheuristik“ wurde zuerst von Glover (GLOVER 1986) eingeführt. Seitdem hat sich der Begriff zur Bezeichnung von approximativen Algorithmen, welche Basis-Heuristiken steuern und teilweise kombinieren durchgesetzt. Das übergeordnete Steuern führt dazu, dass ein definierter Suchraum effektiver und effizienter durchsucht wird. In der Literatur finden sich verschiedene Definitionen von Metaheuristiken. Im Folgenden seien drei beispielhaft zitiert:

“A metaheuristic is formally defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space, learning strategies are used to structure information in order to find efficiently near-optimal solutions.” (OSMAN u. LAPORTE 1996)

“A metaheuristic is an iterative master process that guides and modifies the operations of subordinate heuristics to efficiently produce high-quality solutions. It may manipulate a complete (or incomplete) single solution or a collection of solutions at each iteration. The subordinate heuristics may be high (or low) level procedures, or a simple local search, or just a construction method.” (VOß et al. 1999)

“A metaheuristic is a set of concepts that can be used to define heuristic methods that can be applied to a wide set of different problems. In other words, a metaheuristic can be seen as a general algorithmic framework which can be applied to different optimization problems with relatively few modifications to make them adapted to a specific problem.” (METAHEURISTICS NETWORK 2009)

Alle drei Definitionen verstehen unter einer Metaheuristik die Kombination von einem übergeordneten, steuernden Prozess („*iterative generation process“, „master process“, „set*

of concepts") und einer untergeordneten Heuristik, um einen gegebenen Suchraum nach optimalen Lösungen zu durchsuchen.

In der Regel ist nicht garantiert, dass mittels einer Metaheuristik die optimale Lösung eines Problems gefunden werden kann (WEGENER 2003). Grundsätzlich können auf Metaheuristiken basierende Verfahren gute Lösungen ermitteln, aber auch beliebig schlecht im Vergleich zu tatsächlichen Optimallösungen sein. Generell hängen die Lösungsgüte und die Laufzeit solcher Verfahren von der Strukturierung und Implementierung der einzelnen Schritte und Komponenten ab.

Metaheuristische Verfahren können auf verschiedene Arten, je nach Betrachtungsweise klassifiziert werden. (BLUM u. ROLI 2003). Eine einfache Systematik besteht darin, die Verfahren nach dem Ursprung ihrer Funktionsweise zu gruppieren. Viele Verfahren sind Prozessen oder Strukturen aus der Natur entlehnt und werden entsprechend betitelt (z. B. Simulated Annealing oder Genetischer Algorithmus). Andere Metaheuristiken sind hingegen rein logisch-theoretischer Natur. Somit liegt es nah, zwei Gruppen zu bilden: naturinspirierte Verfahren und nicht-naturinspirierte Verfahren. Diese Klassifizierung ist jedoch relativ ungenau, so dass oft eine klare Zuordnung einzelner Verfahren schwierig ist. Eine weitere Möglichkeit ist es nach dem Gebrauch eines „Gedächtnis" zu klassifizieren. Es gibt Verfahren, die jeweils nur die aktuelle Lösung mit möglichen Lösungen aus der Nachbarschaft vergleichen und solche, die in einem Speicher (Gedächtnis) Informationen zu einer definierten Anzahl vorhergehender Lösungen aufbewahren und in den Lösungsprozess mit einbeziehen. Neben einer Vielzahl weiterer Klassifizierungsmöglichkeiten unterscheidet man Metaheuristiken, die im Rahmen eines laufzeiteffizienten Suchprozesses auf die sukzessive Ermittlung verbesserter Lösungen durch eine iterative Anwendung bestimmter Suchoperatoren abzielen und hierbei auf dem Grundansatz der lokalen Suche beruhen oder solche, die auf dem evolutionären/rekombinationsbasierten Prinzip aufbauen. Verfahren der ersten Gruppe arbeiten mit einer schrittweisen Variation von Lösungen, Verfahren der zweiten Gruppe generieren neue Lösungsvorschläge primär durch Kombination (Kreuzung) von Elementen von bisherigen guten Lösungen.

Die hier vorgestellten aber auch andere Versuche, die einzelnen Verfahren zu systematisieren, können die Vielzahl an Variationen und Kombination der Basisverfahren nicht vollständig erfassen, so dass eine zusätzliche Klasse für Hybride und andere schwer einzuordnende Algorithmen gerechtfertigt ist. Für beide Gruppen (Lokale Suche bzw.

Rekombinatorische Suche) gilt, dass die Grundzüge entsprechender Verfahren generisch (d. h. unabhängig von einem bestimmten Problem) konstruiert werden können, während die eigentliche Anwendung einer Metaheuristik eine Anpassung, bzw. Vervollständigung hinsichtlich problemspezifischer Eigenschaften des Planungs- bzw. Entscheidungsproblems erfordert (s. o.). Metaheuristiken sind somit allgemeine, im Wesentlichen nicht problemspezifische und damit generische Prinzipien und Schemata zur Entwicklung und Steuerung heuristischer Verfahren. Sie sind daher für strukturierte Planungsprobleme fast universell anwendbar. Metaheuristiken werden in vielen Bereichen erfolgreich eingesetzt. Zu nennen sind die Produktionsfeinplanung (SCHULTZ u. MERTENS 2000), die Erstellung von Zeitplänen (ROSS et al. 2003), die Personaleinsatzplanung (HAASE 1999), die Produktentwicklung (BALAKRISHNAN u. JACOB 1996) sowie Beispiele aus der forstlichen Betriebssteuerung (CHEN u. GADOW 2002, CHEN 2003, SEO 2005, HINRICHS 2006a, SODTKE et al. 2006, CHEN u. GADOW 2008).

Bei der praktischen Anwendung ist die Robustheit von Verfahren eine der wichtigsten Voraussetzungen für einen erfolgreichen Einsatz. Ein Verfahren kann dann als robust angesehen werden, wenn Qualität und benötigte Rechenzeit für resultierende Lösungen nur im geringen Maße von der Verfahrensgestaltung für ein konkretes Problem und entsprechender Initialisierung der Steuerungsparameter abhängig sind. Theoretische Untersuchungen zur Leistungsfähigkeit von Metaheuristiken liegen z. B. bei Wegener (WEGENER 2003) vor. Beim Vergleich genereller metaheuristischer Verfahren darf das so genannte NFL-Theorem (No Free Lunch – nichts ist umsonst) (WOLPERT u. MACREADY 1997) nicht unbeachtet bleiben. Dieses Theorem besagt, dass aus theoretischer Sicht keine generelle Überlegenheit bestimmter Optimierungsverfahren besteht. Zwar kann die Leistungsfähigkeit einzelner Verfahren durch Anpassung an konkrete Probleme gesteigert werden, jedoch geht mit einer problemspezifischen Anpassung mit hoher Wahrscheinlichkeit ein Rückgang der Performance einher (IGEL u. TOUSSAINT 2004).

Einen entscheidenden Einfluss auf den Erfolg der Suche nach einer optimalen Lösung mittels Metaheuristiken haben Strategien zur Intensivierung und Diversifikation des Suchprozesses (BLUM u. ROLI 2003). Unter Intensivierung (Exploitation) versteht man eine meist zeitliche Beschränkung des Suchprozesses auf einen als vielversprechend erachteten Teilbereich des Lösungsraums. Durch Diversifikation (Exploration) wird hingegen eine Ausweitung des Suchprozesses im Lösungsraum angestrebt, so dass verstärkt deutlich

unterschiedliche Lösungsmöglichkeiten in den Optimierungsprozess einbezogen werden. Die Diversifikation kann dabei gerichtet oder zufällig eingesteuert werden. Auch die Kombination einer fixen und einer dynamischen Variation der zu durchsuchenden Nachbarschaft ist denkbar.

In den anschließenden Kapiteln wird ein Überblick zu lokalen und evolutionären/rekombinatorischen Suchverfahren und verschiedenen Intensivierungs- bzw. Diversifikationsstrategien gegeben. Eine detaillierte Darstellung der Wirkungsprinzipien und verschiedener Modifikationen ausgewählter Algorithmen erfolgt in Kap. 3.

2.2.3.2.1 Lokale Suche

Die Strategie lokaler Suchverfahren beruht auf der Suche in direkter Nachbarschaft einer möglichen Lösung nach einer besseren Lösung. Dabei werden schrittweise geringfügige Modifikationen an der aktuellen Lösung vorgenommen und die neuen Lösungen mit der Ausgangslösung verglichen. Die Suche nach einer optimalen Lösung erfolgt im sog. Suchraum, der alle gültigen Lösungsmöglichkeiten beinhaltet. Eine Lösung ist gültig, wenn sie keine der evtl. spezifizierten Restriktionen verletzt. Im Verlauf des Suchprozesses lokaler Suchverfahren ist in jedem Iterationsschritt jeweils nur eine Lösung die aktuelle (vgl. Abb. 3). Die in einem Suchschritt durchzuführende Variation oder Permutation der aktuellen Lösung wird mittels einer definierten Nachbarschaftsrelation zwischen den Lösungen des Suchraums festgelegt.

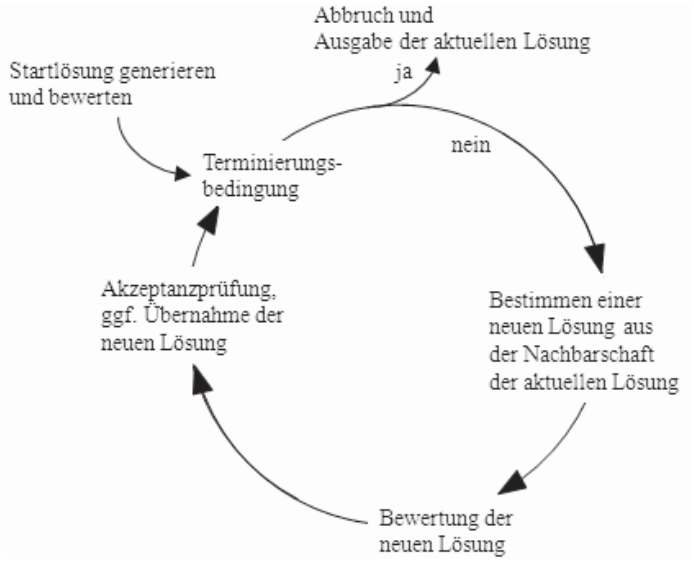


Abb. 3: Genereller Ablauf der lokalen Suche.

Ziel einer Iteration ist es, durch eine geringfügige Variation eine ähnliche Lösung aus der direkten Nachbarschaft der aktuellen Lösung zu generieren und evtl. als neue aktuelle Lösung zu wählen. Ein grundlegender Unterschied zwischen den verschiedenen lokalen Suchverfahren besteht in der Definition der direkten Nachbarschaft und der Strategie zur Bestimmung der neuen aktuellen Lösung. Der Einsatz von Greedy-Strategien führt zu einer starken Intensivierung des Suchprozesses, da bei diesen „gierigen" Suchverfahren die gesamte definierte Nachbarschaft nach einer besseren Lösung durchsucht wird. Bei einigen Verfahren wird über die Suchstrategie die Intensivierung gesteuert. Denkbar ist es z. B. nur eine zufällig gebildete Untermenge der direkten Nachbarschaft zu durchsuchen, oder die Suche in einem Iterationsschritt dann abzubrechen, sobald eine bessere als die aktuelle Lösung gefunden wurde. Wenn in der kompletten Nachbarschaft der aktuellen Lösung keine Variablenkombination entdeckt wird, die zu einem besseren Zielfunktionswert führt, stellt die aktuelle Lösung zumindest ein lokales Optimum dar. In den seltensten Fällen ist dieses lokale Optimum auch das globale Optimum, so dass die Gefahr besteht, dass das Suchverfahren in einem lokalen Optimum „hängen bleibt" und der Suchprozess abgebrochen wird. Um dieser Gefahr entgegenzuwirken, kommen verschiedene Mechanismen zum Einsatz, die es ermöglichen, lokale Optima wieder zu verlassen und die Suche nach dem globalen Optimum fortzusetzen. Die Einsteuerung der Überwindungsstrategien ist entscheidend für den Erfolg des jeweiligen Verfahrens. Die meisten der Überwindungsstrategien beruhen auf einer Diversifikation des Suchprozesses. Dabei wird der Algorithmus „gezwungen" kurzweilig auch schlechtere Lösungen zu akzeptieren (Simulated Annealing (VAN LAARHOVEN u. AARTS 1987), Threshold Accepting (DUECK u. SCHEUER 1990), Tabu Search (TAILLARD et al. 2001), Stochastic Local Search (HOOS u. STUETZLE 2004)), die Suche in andere Regionen des Suchraums zu verlagern (Variable Neighborhood Search (HANSEN u. MLADENOVIĆ 2001), Rollout Algorithm (BERTSEKAS et al. 1997), Pilot Method (DUIN u. VOß 1999)) sowie eine zielgerichtete Variation der Zielfunktion vorzunehmen (Guided Local Search (BALAS u. VAZACOPOULOS 1998)) oder die Suche mit verschiedenen Startlösungen erneut zu starten. (Iterated Local Search, Multi Start Descent (LOURENCO et al. 2003)).

2.2.3.2.2 Rekombinatorische Suche

Rekombinatorische oder evolutionäre Optimierungsverfahren beruhen auf einem gemeinsamen Prinzip, welches Mitte der 1960er Jahre grundlegend erarbeitet wurde (GLOVER 1963, FOGEL et al. 1966, HOLLAND 1975). Darauf aufbauend wurden bekannte Verfahren wie

Genetische Algorithmen, Evolutionsstrategien, Genetische Programmierung und Scatter Search entwickelt.

Bei Optimierungsverfahren dieser Klasse werden mögliche Lösungen (gültige Elemente aus dem Lösungsraum) als Zeichenketten (String) kodiert und durch Anwendung von Rekombinationsoperatoren neue Lösungsmöglichkeiten generiert (vgl. Abb. 4). Im Rahmen der Initialisierung wird eine Population bestehend aus mehreren, meist zufällig erzeugten Individuen (Lösungen) angelegt. Bessere Lösungen werden durch die Kombination der Eigenschaften der aktuell besten Lösungen einer Menge verschiedener Lösungsmöglichkeiten generiert. So ahmen evolutionäre Verfahren die Prinzipien der natürlichen Evolution nach, was sich auch in der im Kontext dieser Verfahren gebräuchlichen Terminologie widerspiegelt. Die Menge möglicher Lösungen wird auch als Population bezeichnet, welche wiederum aus mehreren Individuen (Lösungen) besteht, die durch Chromosomen kodiert werden (WEICKER 2007). Ein wesentlicher Unterschied zu den Verfahren der Lokalen Suche besteht darin, dass eine Population, also mehrere Individuen in einem Iterationsschritt bewertet, kombiniert und verändert werden[3]. Mit Hilfe von Rekombinationsoperatoren werden neue Individuen erzeugt und das Überleben bzw. Fortpflanzen von Individuen durch Selektionsprozesse in Abhängigkeit von deren Lösungsgüte (Fitness) gesteuert.

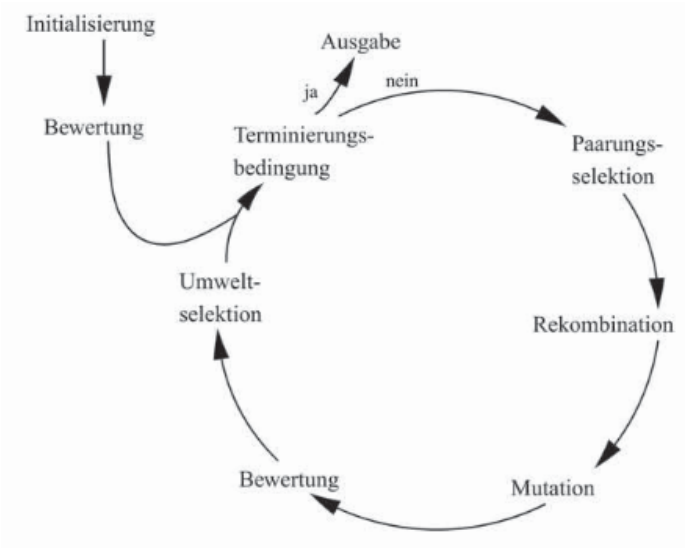


Abb. 4: Genereller Ablauf evolutionärer/rekombinatorischer Algorithmen (WEICKER 2007).

[3] Einige Autoren interpretieren die Verfahren der Lokalen Suche - entgegen der hier vorgenommenen Unterscheidung - als spezielle Fälle der evolutionären Verfahren, bei denen die Population aus nur einem Individuum (einer aktuellen Lösung) besteht (z. B. WEICKER (2007).

Bei der Rekombination werden aus mindestens zwei Individuen durch Kombination ihrer Chromosomen neue Lösungen erzeugt, welche eine neue Generation bilden. Neben der Rekombination oder Vererbung können aber auch zufällige Änderungen der Chromosomen (Mutationen) implementiert werden. Durch diese zufällige Veränderung von Lösungen steht eine Strategie zur Verfügung, welche das Überwinden lokaler Optima ermöglicht. Durch die zufällige Mutation einzelner Lösungen der aktuellen Population wird der Suchverlauf eher diversifiziert. Durch zielgerichtete Veränderungen kann eine Intensivierung der Suche bewirkt werden. Die Fitness oder Lösungsgüte der neu erzeugten Individuen wird über eine vorgegebene Zielfunktion ermittelt.

Anhand der Lösungsgüte werden unter Zuhilfenahme von Auswahlfunktionen selektive Mechanismen eingesteuert, welche Lösungen mit einer hohen Lösungsgüte bevorzugt zur Fortpflanzung bestimmen. Mögliche Abbruchkriterien des Suchprozesses können anhand der Eigenschaften einzelner Individuen oder der Fitness der Individuen einer Generation festgelegt werden. Beispielsweise kann die Rekombination/Mutation solange wiederholt werden, bis die Individuen der aktuellen Generation sich sehr ähnlich werden, die ermittelte Lösungsgüte der jeweiligen Individuen sich auf hohem Niveau einpendelt oder sich einem Grenzwert nähert. Danach würden weitere Iterationen eher zu einer Intensivierung der Suche führen. Bezüglich des Konvergenzverhaltens konnte gezeigt werden, dass bei geeigneter Parametrisierung rekombinationsbasierte Verfahren gegen das globale Optimum konvergieren (RUDOLPH 1997, DAVIS u. PRINCIPE 1991).

2.2.3.2.3 Kombinierte Verfahren

In der Literatur wird eine Vielzahl von kombinierten Verfahren beschrieben, die nicht oder nicht eindeutig in eine der klassischen Verfahrenskategorien *Lokale Suche* und *rekombinatorische Suche* eingeordnet werden können. Es werden Algorithmen vorgestellt, bei denen unterschiedliche Elemente der beiden Verfahrensklassen zu leistungsfähigen Hybridverfahren kombiniert werden. So wird bei der Nachahmung kultureller Evolution die Suche auf Ebene der Population mit der Lokalen Suche auf Ebene der Individuen kombiniert. Nicht nur eine Kombination von Elementen aus den Bereichen der lokalen und der rekombinatorischen Suche führt zu erfolgreichen Hybriden. Auch innerhalb der jeweiligen Verfahrensklassen werden Elemente von verschiedenen Basisverfahren zu Derivaten verschmolzen. Seo (SEO 2005) beschreibt z. B. einen Suchalgorithmus (modified Accelerated

Simulated Annealing, mASA), der Elemente sowohl des Simulated Annealing als auch aus dem Bereich der Tabu-Suche beinhaltet.

2.2.3.2.4 Parallele Suchverfahren

Unter anderem bedingt durch den rasanten technischen Fortschritt im Bereich der parallelen Datenverarbeitung, wurde eine Reihe von Optimierungsverfahren entwickelt, welche auf einer koordinierten Parallelisierung der Suchprozesse basieren. Neben den verschiedenen Formen der Parallelisierung (CRAINIC u. TOULOUSE 2003) der klassischen Optimierungsverfahren aus dem Bereich der lokalen und rekombinatorischen Suche zählen hierzu Ameisensysteme, Partikelschwarmoptimierung sowie Multiagentensysteme. Im Rahmen von Ameisensystemen („Ant Colony Optimization“ (DORIGO u. STÜTZLE 2004)) konstruieren einzelne Suchprozesse (Ameisen) schrittweise vollständige Lösungen für das zu lösende Planungsproblem und markieren dabei qualitativ hochwertige Lösungen oder Elemente entsprechender Lösungen in Analogie zu Pheromonspuren. Diese Informationen werden im weiteren Verlauf der Suche von anderen Ameisen verwendet, um neue Lösungen unter Bevorzugung markierter Lösungen bzw. entsprechender Lösungselemente zu erzeugen.

Ein von Gross (GROSS et al. 1989) durchgeführtes Experiment mit der Ameisenart Iridomyrmex humilis erklärt die Wirkungsweise dieser Mechanismen. Im Rahmen des Experiments wurde einmal das Nest eines Ameisenstaates mit einer Futterquelle über zwei gleich lange Brücken und einmal mit zwei Brücken unterschiedlicher Länge verbunden. Zu Beginn des Experiments schwärmen die Ameisen zur Nahrungssuche aus, wobei einige der Ameisen zufällig eine der beiden Brücken benutzen und ihren Weg mit Pheromonen markieren. Es zeigt sich, dass nach einer gewissen Laufzeit des Experiments die meisten Ameisen nur eine der beiden Brücken benutzen. Nach oftmaligem Wiederholen ergibt sich, dass bei dem Experiment mit den gleich langen Brücken beide Brücken mit gleicher Häufigkeit genutzt werden. Im zweiten Experiment mit unterschiedlich langen Brücken, benutzen die Ameisen zu einem großen Prozentsatz die kürzere Brücke. Dies lässt sich damit erklären, dass die Ameisen, die zu Beginn der Suche zufällig den kürzeren Weg wählen schneller zurück am Nest sind und somit die Pheromonkonzentration auf dem kürzeren Weg deutlich höher wird. Dies führt dazu, dass sich eine Ameise immer häufiger für den kürzeren Weg entscheidet und die Pheromonkonzentration weiter erhöht. Über diese indirekte Kommunikation sind die Ameisen in der Lage die gesamte Futtersuche zu optimieren. Dennoch ist zu beobachten, dass einige Ameisen auch den längeren Weg nutzen, dies führt zu

einer Diversifikation der Suche: Wird eine dritte, noch kürzere Brücke in das Experiment eingeführt, sind die Ameisen dank der Diversifikation in der Lage auf diese noch bessere Route zu wechseln. In der Literatur sind erfolgreiche Anwendungen von Ameisensystemen für diverse Optimierungsaufgaben dokumentiert, wobei der Erfolg dieser Verfahren jedoch zum Teil auf die Integration von Elementen lokaler Suchverfahren zurückzuführen ist (MERKLE u. MIDDENDORF 2003).

Für die Lösung vielschichtiger Planungsprobleme, die eine Koordination teilweise oder völlig autonomer Teilsysteme umfassen, sind zentralistische Planungsansätzen meist nicht geeignet. Dies ist u. a. dann der Fall, wenn zur Problemlösung benötigte Informationen nur dezentral vorliegen und konkurrierende Interessen berücksichtigt werden müssen. Für solche Problemstrukturen bietet sich eine verteilte oder dezentralisierte Form der Problemlösung an. In sog. Multiagentensystemen (WOOLDRIDGE 2002) handeln Softwareagenten im Wesentlichen eigenständig, arbeiten dabei allerdings mit anderen Agenten des Systems zusammen, um eine konkrete Aufgabe gemeinsam zu lösen oder eine Koordination zwischen Systemen zu erreichen. In Abhängigkeit von der gegebenen Problemstruktur können Multiagentensysteme unterschiedlich aufgebaut werden. Kann das Problem generell zentralistisch gelöst werden, basieren die Systeme auf einer Parallelisierung des Suchprozesses mit einer entsprechender Interaktion und Kooperation der einzelnen Agenten. Diesem Ansatz sind auch die Ameisensysteme zuzurechnen. Besteht das Problem hingegen aus mehreren Teilsystemen (Entscheidungsträgern), die größtenteils unabhängig voneinander ablaufen, verlagert sich die Kooperation der Agenten auf eine regel- oder verhandlungsbasierte Koordination verteilter heuristischer Suchprozesse (FINK 2004).

2.2.4 Multikriterielle Bewertung

Viele Planungsprobleme stehen in einem Spannungsfeld verschiedener Interessen und Zielsetzungen. Die Zielsetzungen eines Forstbetriebs lassen sich in drei Ebenen gliedern (WEBER 1995). Die normative Ebene beinhaltet die Grundsätze und Wertvorstellungen (das Leitbild) denen der Betrieb folgt. Sie beschreibt dementsprechend die Motive des forstlichen Handelns. Diese sind zum Teil nicht nur wirtschaftliche bzw. monetär geprägt (Holzproduktion), sondern können auch ideell begründet sein (Liebhaberwert, Naturschutz, Jagd, Tradition). Die strategische Ebene wird durch das Leitbild maßgeblich bestimmt und behandelt die längerfristigen Leistungspotentiale und Gestaltungsmöglichkeiten der waldbaulichen Planungseinheiten. Im Rahmen der strategischen Planung werden betriebliche

Zielkriterien festgelegt, periodische Nutzungsmengen bestimmt und die Prinzipien der Waldbehandlung festgelegt (GADOW 2005). Auf der operativen Ebene wird die Umsetzung der längerfristigen Vorgaben in die waldbauliche Praxis definiert. Aus der Gliederung der Zielsetzungen resultieren für die Forstplanung ebenfalls drei Planungsebenen: die strategische, die taktische und die operationale Planungsebene (PUKKALA 2002). Bei einer mehrkriteriellen Zielausrichtung ergibt sich vor allem auf der operativen Ebene ein Spannungsfeld zwischen den verschiedenen Teilzielen. Wald erfüllt heutzutage mehr als nur eine Funktion. Neben der Holzproduktion spielen verschiedenste Kriterien bei der Waldbewirtschaftung eine Rolle. Naturschutzaspekte kommen ebenso zum Tragen wie etwa die Erholungsfunktion. Besonders Waldflächen in öffentlicher Hand haben einen hohen Anteil an sozialen Funktionen zu erfüllen, die oft im Konflikt mit der ökonomischen Funktion oder anderen Teilzielen stehen. Je nach Leitbild des forstlichen Betriebs verteilen sich folglich die Präferenzen unterschiedlich auf die verschiedenen Waldfunktionen.

Im Rahmen des Einsatzes von Optimierungsverfahren zur Entscheidungsunterstützung ergibt sich daraus ein Bewertungsproblem, da eine mehrdimensionale Zielfunktion (welche minimiert oder maximiert werden soll, Gl. 2) definiert werden muss, welche alle gewünschten Waldfunktionen berücksichtigt. Je nach betrieblichen Vorstellungen müssen Gewichte zu Indikatoren, die die einzelnen Funktionen repräsentieren, ermittelt werden.

$$U_j = \sum_{i=1}^{n} \left(w_i u_i \right)$$

wobei: *Gl. 2*
U_j= Gesamtnutzen der Handlungsalternative j
n= Anzahl Attribute
w_i= Gewicht für Attribut i
u_i= Teilnutzen durch Attribut i

Im Rahmen der Beurteilung von multikriteriellen Szenarien im Bereich der Waldbewirtschaftung existieren exemplarische Ansätze aus verschiedenen Teilbereichen (Wiederaufforstung und Risiko, Verjüngungsplanung, Management und Erholung), wobei die beschriebenen Verfahren meist nicht direkt übertragbar sind. Darüber hinaus müssen für die Lösung der meisten Entscheidungsprobleme bestehende Verfahren modifiziert oder neue Ansätze entwickelt werden (ALBERT 2007).

Im Kontext eines modernen Optimierungssystems zur Entscheidungsunterstützung im forstlichen Betrieb ist somit eine multikriterielle Faktorenanalyse zur Bildung und Bewertung der Zielfunktion für den Optimierungsprozess essentiell (PUKKALA 2002). Multifunktional ausgerichtete Betriebsmodelle unterstützen mehrkriterielle Planungs- und

Entscheidungssituationen dadurch, dass z. B. Gesichtspunkte der Bestandesstabilität, ökologische Kriterien und auch monetäre Kriterien gleichrangig berücksichtigt werden (PRETZSCH et al. 1998). Eine sorgfältige Analyse der Präferenzen ist ein wichtiger Schritt bei der multikriteriellen Planung (PUKKALA 2006). Die Suche nach einer optimalen Lösung eines Planungsproblems ist immer mit den von Entscheidungsträgern definierten Zielen und subjektiven Präferenzen gekoppelt. Präzise formulierte Ziele, Kriterien und Indikatoren sind für die Lösung der vielschichtigen Entscheidungs- und Bewertungsprozesse im Rahmen einer multifunktionalen Forstwirtschaft unerlässlich (SAGL 1995). Um Klarheit über die Ziele und entscheidungsrelevante Einflussfaktoren und deren Gewichtung zu erlangen, sind multikriterielle Bewertungsmethoden geeignet (VACIK et al 2004).

Bereits Anfang der 1950er Jahre wurden theoretische (Vor-)Arbeiten für die mehrkriterielle Optimierung geleistet. Zwanzig Jahre später entwickelte sich die mehrkriterielle Optimierung zu einer eigenständigen Disziplin, die stärker methoden- und anwendungsorientiert war und seit den 1980er Jahren verstärkt im Bereich der Entscheidungsunterstützungssysteme eingesetzt wird. Dieses Forschungsfeld ist heute zumeist unter der Bezeichnung MCDM oder MCDA bekannt. MCDM steht hierbei für Multiple Criteria Decision Making, MCDA für Multiple Criteria Decision Aid bzw. Analysis. Der MCDM-Modellbildung liegt die Annahme zu Grunde, dass bei vielen Entscheidungen mehrere Kriterien zu berücksichtigen sind. Traditionell wurde meist ein hypothetisches oder konstruiertes Gesamtkriterium als Entscheidungsgrundlage angenommen. Prinzipiell kann die multikriterielle Bewertung als Vektor-Optimierungs-Problem charakterisiert werden. Im Zuge der Optimierung wird versucht alle Komponenten des Vektors zu maximieren (ALHO et al. 2002). Zu dieser Problemstellung wurde eine Vielzahl an Methoden entwickelt, die je nach Planungssituation besser oder schlechter zur Lösung geeignet sind. Ausgewählte Beispiele sind das Outranking-Verfahren (ROY 1973), MAUT (multiattributed utility theory) (KEENEY u. RAIFFA 1976), AHP (analytic hierarchy process) (SAATY 1980), die Regime-Methode (HINLOOPEN et al. 1983), der hierarchische interaktive Ansatz (KORHONEN 1986) und AIM (aspiration-level interactive method)(LOTFI et al. 1992).

Je nach Anzahl der berücksichtigten Kriterien und Planungsalternativen lassen sich die Verfahren systematisieren (Tab. 1). Wird die Anzahl der gültigen Alternativen jedoch so groß, dass eine Berechnung aller entscheidungsrelevanten Kriterien nicht mit einem sinnvollen Rechenaufwand zu bewältigen ist, kann auf die in Kap. 2.2.3.2 beschriebenen

metaheuristischen Verfahren zurückgegriffen werden. Die Ansätze zur systematischen Ermittlung der Gewichte der einzelnen entscheidungsrelevanten Kriterien (z. B. Naturschutz, Erholung, wirtschaftlicher Erfolg) können jedoch eingesetzt werden, um die Gewichte für die Nutzenfunktion zur Bewertung der einzelnen Alternativen abzuleiten. Eines der bekanntesten Verfahren, welches ein systematisches Vorgehen zur Bewertung einzelner Entscheidungsfaktoren beinhaltet, ist der Analytische Hierarchieprozesse (AHP). Dieser wurde in den 1970er Jahren in den USA von Thomas Saaty zur Unterstützung von komplexen Entscheidungsproblemen entwickelt.

Tab. 1: Eignung von multikriteriellen Bewertungsansätzen in Abhängigkeit von der Anzahl verwendeter Kriterien und zu prüfender Alternativen.

Anzahl Alternativen	Anzahl Kriterien	
	klein	groß
kein	Outranking Method MAUT Regime Method	AHP HIRMU
groß	Convex cone approach VIMDA AIM	

Der Analytische Hierarchieprozess ist eine Variante der Nutzwertanalyse, wobei der Ausgangspunkt des Verfahrens eine erweiterte Zielhierarchie darstellt. Die letzte Stufe dieser Zielhierarchie wird von den relevanten Planungsalternativen gebildet. Das Ziel des Verfahrens besteht darin, den Beitrag eines jeden Bewertungskriteriums der Hierarchie zu seinen übergeordneten Zielen zu bestimmen. Hierfür vergleicht die Bewertungsperson die Kriterien einer Stufe paarweise bezüglich des direkt übergeordneten Ziels. Aus den Paarbewertungen werden über ein Glättungsverfahren die einzelnen Zielgewichte berechnet, welche additiv und multiplikativ zu einem Gesamtindex für jedes Kriterium aggregiert werden. Nach der Bestimmung der Prioritäten über einen Paarvergleich der einzelnen Attribute mit einer Bewertungsskala ergibt sich eine Bewertungsmatrix, aus welcher der Eigenvektor und der maximale Eigenwert berechnet werden können, welche das Gewicht der einzelnen Attribute ergeben (Tab. 2)

Tab. 2: Paarvergleichsmatrix und aus dem Eigenvektor resultierende Gewichte der einzelnen Attribute (A1-A3).

Entscheidung	A 1	A 2	A 3	**Gewicht**
A 1	1	2	1/2	**0,324**
A 2	1/2	1	4	**0,413**
A 3	2	1/4	1	**0,263**

2.3 Forstliche Entscheidungsunterstützungssysteme

Die Notwendigkeit ein kompaktes und einfach zu bedienendes System zur Optimierung waldbaulicher Prozesse zu erstellen und potentiellen Endanwendern zur Verfügung zu stellen, wurde schon früh in der Geschichte der geregelten forstlichen Planung erkannt. Ein solches System kann die Entscheidungsfindung im Forstbetrieb maßgeblich unterstützen.

Bereits 1826 hat sich Hundeshagen (HUNDESHAGEN 1826, SPEIDEL 1972) mit der Erstellung eines Modells zur nachhaltigen Holznutzung beschäftigt. Das Normalwaldmodell stellt ein einfaches, jedoch nur auf die Holzproduktion bezogenes Nachhaltigkeitskonzept dar. Ein Waldgebiet wird der Umtriebszeit entsprechend in gleich große und gleichaltrige Flächen eingeteilt. Das bedeutet, dass ein Normalwald mit einer Umtriebszeit von 70 Jahren aus 70 gleich großen aber unterschiedlich alten Teilflächen besteht (Abb. 5).

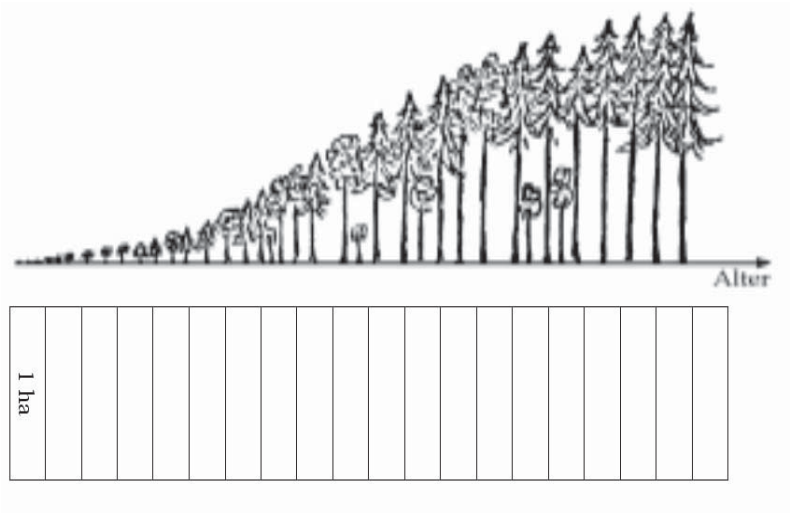


Abb. 5: Normalwaldmodell nach Hundeshagen.

Heute existiert eine Vielzahl deutlich komplexerer Entwürfe forstlicher Managementsysteme, die ein oder mehrere Optimierungsverfahren verwenden, z. T. kombinieren und in Form von Softwarepaketen zur Verfügung stehen (GADOW 2006). Im Folgenden werden ausgewählte Systeme mit integrierten Optimierungsverfahren vorgestellt.

Ein Konzept zur betrieblichen Entscheidungsunterstützung auf Basis des Mehrpfadprinzips stellt Hinrichs (HINRICHS 2006a) vor. Die Abfolge forstlicher Eingriffe wird hierbei als Behandlungspfad oder Pfad bezeichnet. Jeder Bestand bietet eine Vielfalt unterschiedlicher Behandlungspfade, innerhalb eines definierten Handlungsraums. Die Auswahl des optimalen Pfades für einen speziellen Bestand wird vor allem von dessen aktuellen Zustandsgrößen vom sog. „Zwang des Vorhandenen“, von der Gewichtung der

Ziele und von den gesamtbetrieblichen Beschränkungen bestimmt. Zum Aufbau des Entscheidungsunterstützungssystems auf Basis des Mehrpfadprinzips werden neben einem System auf Basis des Simulated Annealing zur Optimierung der Waldentwicklung Modelle zur Erzeugung und Bewertung der möglichen Behandlungspfade der einzelnen Bestände vorgestellt. Die Ergebnisse der Arbeit zeigen die grundlegende Funktionstauglichkeit des Modells in Hinblick auf seine Nutzung als Komponente für ein Entscheidungsunterstützungssystem und zur Optimierung der Waldentwicklung auf Betriebsebene. Für die Weiterentwicklung eines solchen Systems werden, neben der Verbesserung der Prognosegenauigkeit durch fortlaufende Forschung in der Wuchs- und Durchforstungsmodellierung sowie der kosteneffizienten Erfassung von Bestandesinformationen, weitere Untersuchungen zur Bewertung der Pfade und der für die Optimierung der Waldentwicklung einsetzbare Optimierungsverfahren vorgeschlagen. Weitere Beispiele für die praktische Anwendung, des Mehrpfadprinzips, finden sich z. B. in den Arbeiten von Chen u. Gadow (CHEN u. GADOW 2002), Chen (CHEN 2003) und Sánchez Orois (SÁNCHEZ OROIS 2003).

Pukkala (PUKKALA 2006) stellt ein System zur Entscheidungsunterstützung bei der strategischen Forstplanung vor. Hierzu wird eine multikriterielle Nutzenfunktion ermittelt, welche aus mehreren Teilnutzenfunktionen zusammengesetzt ist. Gesucht wird eine Managementalternative, die diese Zielfunktion maximiert. Hierzu werden u. a. ein Wuchs- und Eingriffsmodell verwendet, um die Auswirkungen der Behandlungsalternativen abzuschätzen. Darauf aufbauend werden verschieden Optimierungsverfahren untersucht und verglichen, um aus dem Alternativenraum die beste Lösung zu ermitteln. Der Autor zeigt, dass Verfahren mit Strategien zum Überwinden lokaler Optima gut geeignet sind und durch Kombination verschiedener Verfahren die Lösungsqualität verbessert werden kann.

Das finnische MELA-System (REDSVEN et al. 2007) basiert ebenfalls auf dem Mehrpfadprinzip und ist inzwischen eine wichtige Basis der finnischen Forsteinrichtung geworden und wird in privaten Forstbetrieben, im Staatswald und im Kleinprivatwald regulär eingesetzt und ständig weiterentwickelt (GADOW 2006). Das MELA-System vereinigt ein Waldwachstumsmodell und ein operationales DSS, welches zur Lösung von zwei grundlegenden Problemstellungen konzipiert wurde. Einerseits zielt das System auf die forstlichen Produktionspotentiale ab, andererseits wird eine Entscheidungsunterstützung geboten, wie Bestände zu behandeln sind, um definierte Ziele zu erreichen. Im MELA-System

wird das Management von Wäldern endogen gestaltet, d. h. die Entscheidung wann und wie eine individuelle Planungseinheit zu behandeln ist, hängt von nutzerspezifischen Restriktionen und Zielen ab. MELA besteht aus zwei Komponenten: einem automatisierten einzelbaumbasierten Bestandessimulator und einem Optimierungsmodul (LAPPI 1992) auf Basis der Linearen Programmierung. MELA simuliert eine endliche Anzahl an plausiblen alternativen Management-Plänen für die einzelnen Management-Einheiten über eine definierbare Zeitspanne in Übereinstimmung mit vorgegebenen Simulationsparametern. Die einzelnen Management-Pläne unterscheiden sich beispielsweise durch die zeitliche Struktur der jeweiligen Steuerungsaktivitäten. Anschließend an eine Simulation können ein Produktionsplan für das gesamte simulierte Gebiet und Managementpläne für einzelne Planungseinheiten in Abhängigkeit der nutzerspezifischen Optimierungsstrategie (Ziele und Restriktionen) ermittelt werden.

Seo stellt ein System zur Pfadoptimierung auf Ebene des Forstbetriebs (SEO 2005) vor. Als Optimierungsverfahren wird eine modifizierte Variante des Simulated Annealing verwendet (modified Accelerated simulated Annealing, mAsA). Mittels dieses Verfahrens wird aus einer definierten Anzahl von möglichen Entwicklungspfaden die Pfadkombination ermittelt, die entweder das höchste Holzvolumen oder den höchsten Bestandeserwartungswert liefert.

Sodtke (SODTKE et al. 2006) beschreibt die Entwicklung eines Prototyps zur Entscheidungsunterstützung bei der betrieblichen Planung auf Basis des Simulators SILVA. Zur Identifizierung einer optimalen Managementalternative kommt der Tabu Search-Algorithmus zum Einsatz. Das Optimierungsverfahren greift auf eine mehrkriterielle Bewertungsfunktion zurück, welche die Holzproduktion, die Bestandesstabilität, die Biodiversität, die Schutz- und die Erholungsfunktion und die Jagd berücksichtigt.

Öhman und Eriksson stellen eine Fallstudie vor, in welcher räumliche Aspekte in den Optimierungsprozess mit einbezogen werden (ÖHMAN u. ERIKSSON 2002). Es werden drei Ansätze entwickelt, um unter Einbeziehung räumlicher Aspekte bezüglich durchgehender Flächen von Altbeständen, langfristige Forstplanungsprobleme zu lösen. Zwei der Ansätze integrieren Lineare Programmierung und Simulated Annealing in einem Lösungsansatz. Simulated Annealing wird dabei zum Lösen der räumlichen Komponente (Verminderung von Flächen mit Randeffekten, Förderung von Kernflächen) des Problems herangezogen. Das Verfahren der Linearen Programmierung wird zur Maximierung des Nettobarwertes (NPV)

unter Einhaltung verschiedener Restriktionen verwendet. Der dritte Ansatz greift nur auf die Metaheuristik Simulated Annealing zurück, um den gesamten Problemkomplex zu lösen. Alle drei Ansätze wurden im Rahmen einer Fallstudie auf ein Gebiet in Nordschweden mit 755 Beständen angewendet. Dabei zeigte sich, dass die kombinierten Ansätze effektivere Lösungen produzieren.

Degenhardt (DEGENHARDT 2006) nutzt ein Verfahren aus dem Bereich der Direkten Suche, um für Kiefernreinbestände optimale Behandlungsvarianten abzuleiten. Als Wuchsmodell wird ein Derivat des Modells BWINPro verwendet, welches für die Kiefer in Brandenburg reparametrisiert wurde. In Verbindung mit der Kompass-Suche und einem Modul zur Bestandesbehandlung wird ein Konzept entwickelt, um eine Behandlungsalternative zu bestimmen, aus welcher der höchste Nettoerlös resultiert.

Chen und v. Gadow präsentieren ein Forstplanungsmodell, welches verschiedene räumliche Parameter berücksichtigt. Dabei wird unter Verwendung von Simulated-Annealing eine optimale Pfadkombination für die Bestände des Auswertungsgebiets ermittelt (CHEN u. GADOW 2008). Die verwendete Zielfunktion wird dabei variiert. Es werden eine rein ökonomisch ausgelegte Zielfunktion, eine Zielfunktion mit zusätzlicher Berücksichtigung einer konstanten Holzlieferung (Even-Flow) und eine Zielfunktion konstruiert, welche neben dem ökonomischen Teilnutzen und der konstanten Einschlagsmenge eine räumliche Komponente beinhaltet. Die räumliche Komponente zielt auf eine möglichst hohe Klumpung der in einem Eingriffsintervall zur Nutzung vorgeschlagenen Bestände ab. Hierzu werden separat in einem GIS der Nachbarschaftswert über die Distanz der Centroide der einzelnen Bestände bzw. Polygone und die Länge der gemeinsamen Grenze berechnet und die gespeicherten Ergebnisse in den Optimierungsprozess integriert.

3 Modifikation und Auswahl geeigneter Optimierungsverfahren

Zum Aufbau einer geeigneten Optimierungs-Komponente, werden zunächst verschiedene Optimierungsalgorithmen in ihrer Grundform und in erweiterten Varianten erarbeitet. Bei den Modifikationen liegt der Fokus auf der Parallelisierung der Verfahren und der Steuerung der Diversifikation, bzw. der Intensivierung des Suchprozesses. Anschließend werden die vorgestellten Algorithmen hinsichtlich ihrer Eignung zur Lösung der in Kap. 1.1 definierten Problemstellungen (optimale Bestandesauswahl, Optimierung der Nutzung und Pflege, Optimierung der mittelfristigen Bestandesbehandlung) verglichen. Darauf aufbauend werden die zu verwendenden Verfahren ausgewählt und die konkrete Gestaltung der

Optimierungskomponenten dargestellt. Metaheuristische Verfahren sind besonders gut geeignet, komplexe, modellbasierte Zielsysteme zu optimieren (vgl. Kap. 2.2.3). Darüber hinaus sind sie sehr flexibel einsetzbar. Bei der Implementierung ausgewählter Verfahren zur Lösung konkreter Optimierungsprobleme sind zwei grundsätzliche Anforderungen an die Struktur des vorliegenden Problemkomplexes zu beachten:

- (Zwischen-) Lösungen müssen bezüglich ihrer Güte mithilfe einer Zielfunktion bewertet werden können.
- Es muss möglich sein, Lösungen des Planungsproblems so zu kodieren, dass notwendige Suchoperatoren (z. B. lokale Veränderung, Rekombination) auf das Problem angewendet werden können.

Sind diese Voraussetzungen erfüllt, können metaheuristische Optimierungsverfahren für die Lösung einer Vielzahl unterschiedlicher Problemstellungen eingesetzt werden, welche mit klassischen Verfahren wie z. B. der linearen Programmierung nur eingeschränkt oder nicht zu lösen sind (z. B. nicht-lineare, nicht-stetige sowie NP-schwere Probleme, s. o.). Im Rahmen der Entwicklung von Softwaresystemen mit integrierten Optimierungsmodulen ist es bei Verwendung heuristischer Optimierungsverfahren möglich, das zu lösende Problem sehr realitätsnah abzubilden und komplexe Modelle zur Beschreibung des Planungsproblems zu verwenden. Die Simulation von teilweise zufallsgesteuerten oder regelbasierten Wirkungszusammenhängen kann somit in den Optimierungsprozess eingebunden werden. Aus der Vielfalt und der universellen Anwendbarkeit heuristischer Optimierungsverfahren resultiert ein Entscheidungsproblem bei der Wahl eines konkreten Verfahrens zur Lösung des gegebenen Problems. Im Rahmen der Entscheidung für ein Verfahren kann nur sehr eingeschränkt auf Erfahrungen zurückgegriffen werden, da im Rahmen entsprechender publizierter Untersuchungen meist beschrieben wird, inwieweit ein bestimmtes Verfahren geeignet ist, ein spezielles (nicht übertragbares) Problem zu lösen. Vergleichende Beiträge sind in der Regel nicht geeignet, um generelle Aussagen zur Lösungsgeschwindigkeit und -qualität konkreter Verfahren treffen zu können, da bei einer konkreten Anwendung der Verfahren die zugrunde liegenden Problemstellungen zu unterschiedlichen Lösungsverhalten führen. Umfassende Vergleiche von Lösungsqualitäten und benötigten Rechenzeiten verschiedener Suchverfahren liegen nur für eine geringe Anzahl von Standardproblemen vor (z. B. Traveling Salesman Problem, JOHNSON u. MCGEOCH 2002). Aussagekräftige Vergleichsstudien hinsichtlich der Leistungsfähigkeit unterschiedlicher Verfahren in

Abhängigkeit von Merkmalen realistischer Planungsprobleme existieren im forstlichen Kontext nur in geringem Maße. Um ein oder mehrere geeignete Verfahren für den Einsatz in dem hier vorgestellten Entscheidungsunterstützungssystem zu ermitteln, werden deshalb verschiedene Standardverfahren und modifizierte Varianten implementiert und anhand von zwei konstruierten Optimierungsproblemen verglichen. Die Struktur dieser hypothetischen Probleme richtet sich nach den Problemstrukturen der mittels des vorgestellten Entscheidungs-Unterstützungs-Systems zu lösenden Fragestellungen.

Dennoch können aus den Grundprinzipien metaheuristischer Optimierungsverfahren theoretisch fundierte Erkenntnisse abgeleitet werden, welche die Auswahl von konkreten Verfahren unterstützen können. Beispielsweise sind auf dem Prinzip der Rekombination beruhende Verfahren oder evolutionäre Ansätze dann besonders gut geeignet, wenn das zu lösende Planungsproblem zerlegt werden kann (GOLDBERG 2002). Die guten Ergebnisse sind dadurch zu erklären, dass bei der Lösung hochwertige Teillösungen für einzelne Teilprobleme ermittelt werden und anschließend die Gesamtlösung aus Teillösungen zusammengesetzt wird. Im Rahmen von Scatter Search (LAGUNA u. MARTI 2003, MARTI et al. 2006) wird dieser Lösungsansatz explizit verwendet und hochwertige Einzellösungen (Consistent Chains) kombiniert. Lokale Suchverfahren sind hingegen dann effizient, wenn ähnliche Lösungen eine ähnliche Lösungsgüte aufweisen (CHRISTENSEN u. OPPACHER 2001). Lösungen sind im Kontext der Lokalen Suche dann ähnlich zueinander, wenn hinsichtlich einer verwendeten Nachbarschaftsrelation nur wenige Suchschritte zwischen den jeweiligen Lösungen liegen. Die direkte praktische Anwendung solcher theoretischen Erkenntnisse wird jedoch dadurch erschwert, dass es z. B. notwendig sein kann, den gesamten Suchraum zu untersuchen, um Aussagen darüber treffen zu können, ob das Problem zerlegbar ist, oder ein Problem sowohl gut zerlegbar ist als auch ähnliche Lösungen eine ähnliche Lösungsgüte aufweisen.

Nach der Entscheidung für ein spezielles Verfahren muss dieses dem zu lösenden Problem angepasst und entsprechend implementiert werden. Wie in Kap. 2.2.3 erläutert wird, definieren Metaheuristiken übergeordnete Regeln zur Steuerung heuristischer Verfahren. Diese müssen sinnvoll dem Problem angepasst werden. In dieser Notwendigkeit liegt ein wesentlicher Unterschied zwischen der Anwendung heuristischer Optimierungsverfahren und den meisten klassischen Verfahren. So kann z. B. für LP-Planungsprobleme (LP = Linear Programming) oder in Grenzen auch für ganzzahlige Optimierungsprobleme (MIP: Mixed Integer Programming) beschränkter Größe Standardsoftware verwendet werden (z. B. ILOG

2009). Bei Verwendung von Standardsoftware ist der Nachteil gegeben, dass diese vom Nutzer erlernt werden muss und für die Anschaffung zusätzliche Kosten anfallen. Bedeutender ist jedoch, dass hierbei gegebenenfalls eine starke Vereinfachung von Planungsproblemen und entsprechende Verschlechterungen hinsichtlich der Realitätsnähe vorausgesetzt werden. Eine Vielzahl realer Planungsprobleme kann oft nicht in dem Maße sinnvoll vereinfacht und angepasst werden, dass es möglich wäre, Standardsoftware zu verwenden. Für die erfolgreiche Lösung komplexer und individueller Problemstellungen ist eine problemspezifische Verfahrensgestaltung erforderlich.

Das ausgewählte Optimierungsverfahren ist in einem zweiten Schritt problemspezifisch einzubinden und ggf. zu modifizieren. Die wichtigsten Designentscheidungen sind hierbei die Definition eines geeigneten Suchraums (die Kodierung von Handlungsalternativen), die Festlegung von dazu passenden Suchoperatoren (Nachbarschaftsrelationen), die Initialisierung des Verfahrens (Bestimmung von geeigneten Startlösungen, verfahrensspezifische Parameter) und die Ermittlung einer Zielfunktion, welche den unterschiedlichen Handlungsalternativen (Lösungen) eine Lösungsgüte zuweist.

Bei der Anwendung von Metaheuristiken sind die problemspezifische Gestaltung der Verfahren und die Berücksichtigung von problemspezifischen Informationen für das konkrete Design der Verfahren von essentieller Bedeutung (CHRISTENSEN u. OPPACHER 2001, DROSTE u. WIESMANN 2002). Durch Berücksichtigung charakteristischer Eigenschaften des zu lösenden Planungsproblems bei der Gestaltung von metaheuristischen Verfahren kann die Lösungsqualität der Optimierung erhöht werden. Ist beispielsweise der Zielfunktionswert des globalen Optimums bekannt oder kann dieser zumindest abgeschätzt werden, ist es möglich, diese Information dazu zu nutzen, um festzustellen, ob das Verfahren gegen ein lokales Optimum konvergiert und so entsprechende Gegenmaßnahmen einzuleiten.

Die Gestaltung konkreter Verfahren wird durch anwendungsorientierte Untersuchungen unterstützt, welche einen Zusammenhang zwischen der erzielbaren Lösungsqualität und der Verfahrensgestaltung herstellen. So wurde etwa untersucht, wie die Lösungsqualität von rekombinationsbasierten Verfahren von der verwendeten Problemkodierung beeinflusst wird (ROTHLAUF 2002). Rowe behandelt hingegen den Zusammenhang von Lösungsqualität und den eingesetzten Suchoperatoren (ROWE et al. 2002).

Im Folgenden werden aus dem Bereich der Lokalen Suche die Standardverfahren Simulated Annealing (SA), Kompasssuche (KS), und Tabu Search (TS) dargestellt. Aus der

Kategorie der rekombinationsbasierten Verfahren wird der Genetische Algorithmus (GA) behandelt. Die Modifikationen dieser Verfahren zielen darauf ab, die Lösungsgeschwindigkeit und/oder -qualität zu verbessern. Aus dem Bereich der Greedy-Strategien wird der Dijkstra-Algorithmus beschrieben und ebenfalls modifiziert. Dieser Algorithmus wird als Hilfsverfahren eingesetzt, um z. B. in einem gegebenen Wegenetz die kürzeste Route zwischen zwei Knotenpunkten (Beständen) zu bestimmen. Soll im Rahmen der Optimierung z. B. eine Auswahl an Beständen getroffen werden, welche u. a. dicht beieinander liegen um den Maschineneinsatz zu optimieren, können die minimalen Wegelängen zwischen zwei Beständen mit dem Dijkstra-Algorithmus ermittelt und diese Information in den weiteren Optimierungsprozess mit einbezogen werden.

3.1 Kompasssuche

Die Kompasssuche ist den Verfahren der Direkten Suche (Direct Search) zuzuordnen. Diese Technik wurde in den 1960iger Jahren populär. Aufgrund der oft einfach zu implementierenden Algorithmen oder der teilweise hohen Zuverlässigkeit der Verfahren sind sie bis heute im Gebrauch. Gerade in den letzten Jahren wurde diesen Verfahren im Zusammenhang mit parallelem oder aufgeteiltem Rechnen wieder verstärkte Aufmerksamkeit geschenkt (KOLDA et al. 2003). Dabei sind besonders folgende zwei Punkte deutlich geworden:

1. Direct Search ist nach wie vor eine effektive und teilweise die einzige Möglichkeit zur Lösung komplexer, nicht-differenzierbarer Optimierungsprobleme.
2. Für eine große Anzahl dieser Methoden kann ein Konvergieren garantiert werden.

Der grundlegende Algorithmus im Bereich dieser Verfahren wurde von Davidon (DAVIDON 1959) vorgestellt. Die sog. Kompasssuche ist die Basis für eine Vielzahl an Verfahren wie z. B. Alternating Directions, Alternating Variable Search, Axial Relaxation, Coordinate Search oder Local Variation und liefert verlässliche Ergebnisse. Der Grundgedanke dieses Verfahrens ist ein iteratives Vorgehen, welches aus den Ergebnissen der Zielfunktion und entsprechenden Parametersätzen die Strategie zur Definition der Parametersätze für den nächsten Iterationsschritt ableitet. Die Kompasssuche soll anhand eines zweidimensionalen Optimierungsproblems veranschaulicht werden (Gl. 3). Es wird das Minimum einer Funktion in Abhängigkeit von x_1 und x_2 gesucht.

$$\min \leftarrow f(x_1, x_2) \qquad \text{wobei: } x_1, x_2 \in \mathbb{R} \qquad \textit{Gl. 3}$$

In Abb. 6 ist die Lösung des Problems (Globales Minimum) mit einem Sternchen gekennzeichnet. Die Isolinien beschreiben solche Variablenkombinationen, die zu gleichen Funktionswerten führen. Insgesamt werden sechs einzelne Iterationsschritte (a bis f) dargestellt. Der schwarze Punkt markiert die Ausgangsposition (Variablenkombinationen) in jedem Iterationsschritt. Die grauen Punkte veranschaulichen die „Tastversuche“ in Richtung der Achsen des Koordinatensystems. Dies bedeutet, dass ausgehend von einem Initialpunkt 2n Lösungsmöglichkeiten ($D_{\oplus}$) durch Variieren der Variablenkombination des Ausgangspunktes berechnet werden, wobei *n* die Anzahl der in der Zielfunktion verwendeten Variablen ist. Die Koordinaten des Ausgangspunktes werden um einen definierten Betrag Δ_k reduziert und erhöht. Bei einer zweidimensionalen Zielfunktion werden dementsprechend vier Lösungsversuche je Iterationsschritt (*k*) berechnet.

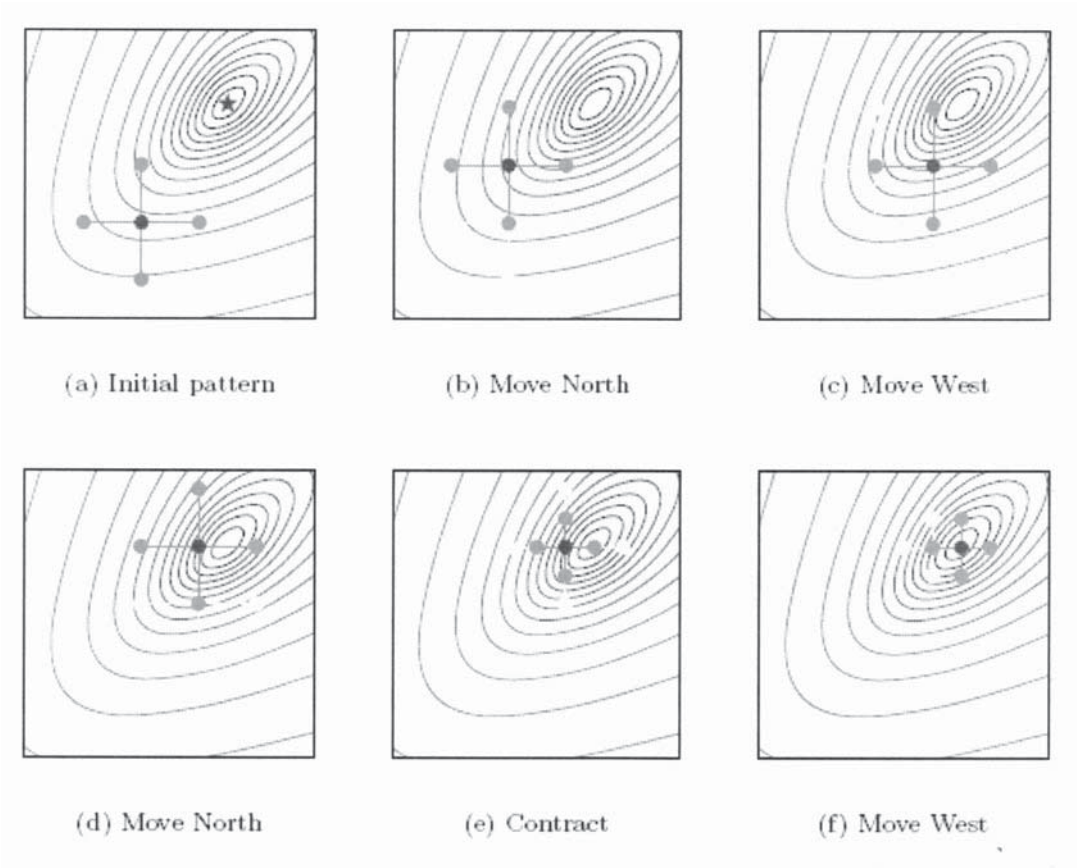


Abb. 6: Kompasssuche in Abhängigkeit von zwei Parametern (DEGENHARDT 2006).

Liefert einer der Lösungsversuche einen kleineren Funktionswert als der Ausgangspunkt, so wird dieser zum Ausgangspunkt für den folgenden Iterationsschritt. Δ_k ändert sich nicht. Liefern alle berechneten Lösungsmöglichkeiten lediglich Werte größer oder gleich dem Ausgangspunkt des aktuellen Iterationsschrittes, wird der Ausgangspunkt für die nächste Iteration nicht variiert. Die Schrittlänge Δ_k wird jedoch halbiert. Unterschreitet Δ_k einen Schwellenwert (Abbruchkriterium) so wird der Algorithmus beendet. In Unterabbildung a sind die vier Lösungsversuche ausgehend vom zuvor initialisierten Startpunkt aufgetragen. Die Lösungsmöglichkeit in „nördlicher" Richtung liefert von allen vier Möglichkeiten den kleinsten Wert. Da dieser kleiner dem Funktionswert an der Ausgangsstelle ist, wird der Ausgangspunkt für den nächsten Iterationsschritt um den Betrag der definierten Schrittweite (Δ_k) nach Norden verschoben. In den Schritten a, b, c und d kann jedes Mal in mindestens einer der Suchrichtungen ein kleinerer Funktionswert entdeckt werden. Es wird somit lediglich der Ausgangspunkt verschoben. In Iterationsschritt d liefern die „Tastversuche" nur größere oder gleiche Funktionswerte. Es wird entsprechend der Zeile 15 in Abb. 7 lediglich die Schrittweite (Δ_k) für die nächste Iteration (e) halbiert und nicht der Ausgangspunkt verschoben.

Initialisierung

$f: \mathbb{R}^n \rightarrow \mathbb{R}$	
$x_0 \in \mathbb{R}$	Startpunkt
$\Delta_{tol} > 0$	Schwellenwert zum Testen auf Konvergenz
$\Delta_0 > \Delta_{tol}$	Initiale Schrittlänge
$D_\oplus$	Satz an Suchrichtungen mit $\{\pm e_i \mid i=1, ..., n\}$, wobei e_i der i-te Einheits-Koordinaten-Vektor in $\mathbb{R}^n$ ist
$d_{min}, d_n \in D_\oplus$	Beste (minimale) Lösung, aktuelle Lösung
$k=0$	Zähler für Iterationsschritte

Algorithmus (min)

```
1 stop=false
2 Min ← f(x0)
3 dmin ← d0
4 Do until stop ← true
5   for i ← 0 to 2n do
6     if f(xk+Δkdi) < f(xk) and f(xk+Δkdi) < Min
7         Min ← f(xk+Δkdi)
8         dmin ← di
9     end if
10  end for
11  if new minimum found do
12    xk+1 ← xk+Δkdmin (neuer Ausgangspunkt)
13  end if
14  else do
```

```
15    Δ_{k+1} ← 0.5Δ_k (ändere Schrittweite)
16    if Δ_{k+1} < Δ_tol stop ← true
17  end else
18  k ← k+1
19 end until
```

Abb. 7: Algorithmus zur Kompasssuche

In Unterabbildung e ist dies graphisch durch die verkürzten Abstände zu den grauen Punkten (Lösungsversuchen) dargestellt. Die Suche mit halbierter Schrittweite führt zu einer Verschiebung in westlicher Richtung. Von diesem Punkt (f) kann bei gegebener Schrittweite keine bessere (kleinere) Lösung gefunden werden, die Schrittweite wird wieder halbiert und unterschreitet das Konvergenzkriterium Δ_{tol}. Somit bricht der Algorithmus (Abb. 7) gemäß Zeile 16 ab und das Minimum wird näherungsweise erreicht. Wie nahe der Schätzer dem tatsächlichen Minimum kommt, hängt somit unter anderem vom Abbruchkriterium (Δ_{tol}) ab.

Im folgenden Abschnitt werden mögliche Modifikationen der Kompasssuche vorgestellt und hinsichtlich der Laufzeit gegenübergestellt. Zur Beurteilung der Laufzeit wird die benötigte CPU-Zeit in Millisekunden (ms) und die Anzahl der Aufrufe der Kostenfunktion verwendet. Da die Kostenfunktion bei komplexen Optimierungsproblemen sehr viel Rechenzeit in Anspruch nimmt, liefert die Anzahl der zur Problemlösung benötigten Aufrufe mehr Information, als die Angabe zur Anzahl der Iterationsschritte. Diesbezüglich bleibt bei einigen Autoren unklar, was unter einer Iteration zu verstehen ist (z. B. Anzahl äußere Schleifendurchläufe oder Anzahl innere Schleifendurchläufe). Ein weiterer Vorteil besteht so auch beim Vergleich mehrerer Algorithmen hinsichtlich ihrer Effizienz, da die Kostenfunktion gleich ist und die Anzahl der Aufrufe dieser Funktion direkt gegenüber gestellt werden kann. Der in Abb. 7 veranschaulichte einfache Algorithmus bietet mehrere Ansatzstellen zur Verbesserung. Eine denkbare Modifikation setzt an der Wahl der Schrittlänge Δ_k an. Bei der einfachen Kompasssuche wird diese nur verändert (halbiert), wenn keine akzeptable Lösungsmöglichkeit bei einem Iterationsschritt gefunden wurde. Denkbar wäre im anderen Fall die Schrittweite zu erhöhen und optional einen Begrenzer für die maximal zulässige Schrittweite einzuführen. Liegt der Startpunkt der Optimumsuche sehr weit vom Optimum entfernt, kann der Algorithmus sich so schneller einem Optimum annähern. In Abb. 8 ist der Optimierungsfortschritt (aktuelles Minimum) über der Anzahl der Zielfunktionsaufrufe aufgetragen. Das gesuchte, wahre Optimum (Minimum) einer zweidimensionalen Kostenfunktion liegt bei 11,0. Bei Verwendung des modifizierten

Algorithmus (Abb. 8 mod) wurde ein Schritterhöhungs-Faktor von 1,5 eingeführt und auf 100 Einheiten begrenzt. Die Gesamtlaufzeit bei ansonsten gleichen Startbedingungen des modifizierten Algorithmus liegt bei 47 ms. Der Optimierungsprozess bei Verwendung des Basisalgorithmus benötigt 344 ms mit 1200 Aufrufen der Kostenfunktion. Die modifizierte Variante produziert 365 Aufrufe. Die beiden Bereiche a und b (Abb. 8) veranschaulichen den oben erwähnten Effekt, dass der Algorithmus eine gewisse Zeit benötigt, um zu „erkennen“, dass das gesuchte Optimum approximativ erreicht ist. Die Länge dieser Phase kann über das Konvergenzkriterium Δ_{tol} beeinflusst werden. Eine weitere Modifikation in Kombination mit der variablen Schrittweitenanpassung besteht darin, nur solche Lösungsversuche zu akzeptierten, die ein Minimum an Verbesserung zum Ausgangspunkt liefern. Bei der Prüfung $f(x_k+\Delta_k d_n) < f(x_k)$ könnte eine Funktion $p(\Delta_k)$ integriert werden, so dass nun $f(x_k+\Delta_k d_n) < f(x_k) - p(\Delta_k)$ erfüllt sein muss, um eine Lösungsmöglichkeit zu akzeptieren. Eine einfache Wahl für p wäre z. B. $p(t) = \alpha t^2$ mit α größer 0. Abb. 9 zeigt die Verbesserung, die durch die Berücksichtigung einer Judge-Funktion erzielt werden kann. *Mod a* beschreibt den Iterationsfortschritt für die Variante der Kompasssuche mit variabler Schrittweite, bei *mod b* wurde zusätzlich die Judge-Funktion $p(x) = ax^2$ verwendet.

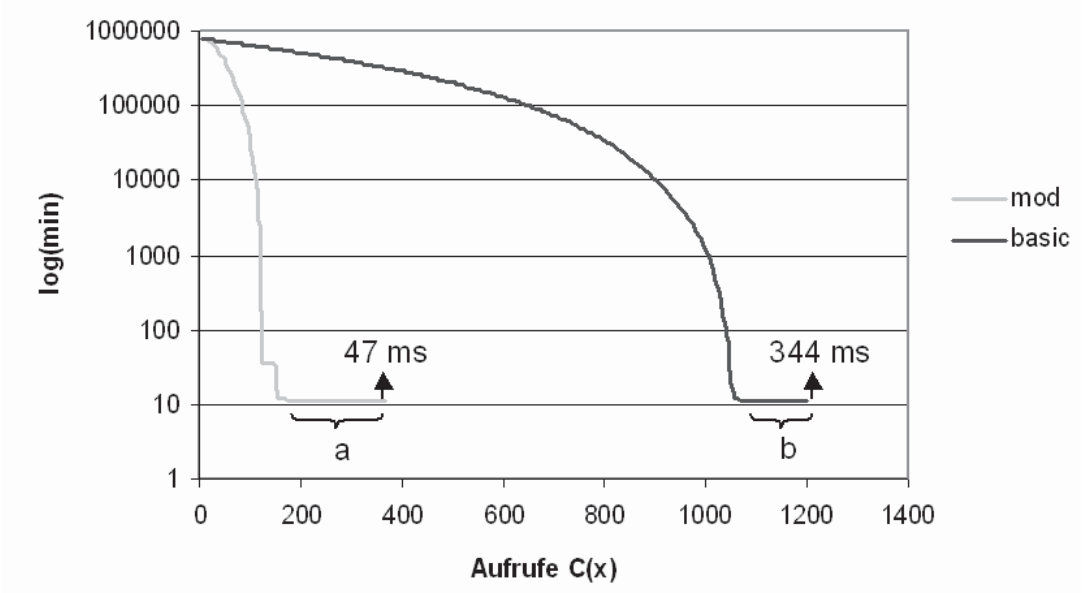


Abb. 8: Optimierungsfortschritt bei einer zweidimensionalen Funktion mit dem wahren, globalen Minimum bei 11,0.

Die Anzahl der Aufrufe von C(x) reduziert sich bei diesem zweidimensionalen Beispiel (s. o.) von 289 auf 265. Auch ist es denkbar, die Suchrichtung nicht nur parallel zu den Achsen des Koordinatensystems zu wählen, sondern noch weitere, systematisch erzeugte Suchrichtungen

zu berücksichtigen. Dies erhöht jedoch die Anzahl der Aufrufe der Kostenfunktion. Deshalb wäre es - auch für den Fall der Suche nur entlang des Koordinatensystems – denkbar, nicht alle Lösungen zu berechnen, sondern die innere Schleife abzubrechen, sobald eine oder eine definierte Anzahl besserer Lösungen gefunden wurde. In Abb. 9 ist der Iterationsfortschritt zusätzlich mit Schleifenabbruch, sobald eine bessere Lösung gefunden wird, aufgetragen (mod c). Im Vergleich zu der Variante mit variabler Schrittlänge und Judge-Funktion wird die Anzahl der Aufrufe von C(x) noch einmal von 265 auf 162 reduziert.

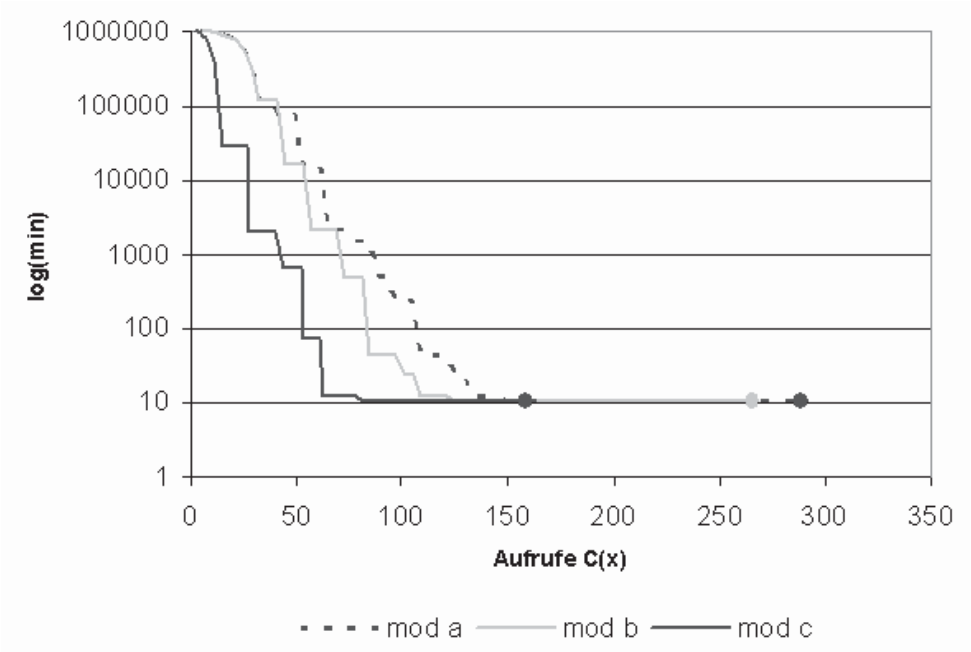


Abb. 9: Optimierungsfortschritt mit variabler Schrittweite, Judge-Funktion und Schleifenabbruch (mod c), mit variabler Schrittweite und Judge-Funktion (mod b) und nur mit variabler Schrittweite (mod a)

Anhand des zweidimensionalen Beispiels konnte gezeigt werden, dass neben einer sorgfältigen Wahl der Startparameter geringfügige Änderungen am Standardalgorithmus zur Kompasssuche den Optimierungsprozess maßgeblich verbessern können. Zur Lösung des zweidimensionalen Optimierungsproblems benötigte der Basisalgorithmus 1200 Aufrufe der Kostenfunktion. Die Variante mit den drei Änderungen *variable Schrittlänge*, *Judge-Funktion* und *Schleifenabbruch* benötigt zur Lösung lediglich 162 Aufrufe. Dabei nimmt die Lösungsqualität nicht ab. Das globale Optimum von 11,0 an der Stelle (8; 0) wurde durch die Variante *mod c* und die Basisvariante approximativ wie folgt gelöst:

y: 11,000000047 (Minimum)

x1: 8,0004882812, x2: 0,0

Nachfolgend ist der modifizierte Algorithmus (Abb. 10) zur Kompasssuche dargestellt. Änderungen zum Standardalgorithmus sind dunkel hervorgehoben.

Initialisierung

$f: \mathbb{R}^n \rightarrow \mathbb{R}$	
$x_0 \in \mathbb{R}^n$	Startpunkt
$\Delta_{tol} > 0$	Schwellenwert zum Testen auf Konvergenz
$\Delta_0 > \Delta_{tol}$	Initiale Schrittlänge
$D_\oplus$	Satz an Suchrichtungen mit $\{\pm e_i \mid i=1, ..., n\}$, wobei e_i der i-te Einheits-Koordinaten-Vektor in $\mathbb{R}$ ist
$d_{min}, d_n \in D_\oplus$	Beste (minimale) Lösung, aktuelle Lösung
ϕ	Expansionsfaktor ≥ 1
p	Judge-Funktion mit $[0, +\infty) \rightarrow \mathbb{R}$, p(t) fällt mit $t \rightarrow 0$ und $p(t)/t \rightarrow 0$ wenn $t \downarrow 0$
λ	Anzahl Verbesserungen bis vorzeitiger Schleifenabbruch $0 < \lambda \leq n$
k=0	Zähler für Iterationsschritte

Algorithmus (min)

```
1 stop ← false
2 Min ← f(x0)
3 dmin ← d0
4 Do until stop=true
5   counter ← 0
6   for i ← 0 to 2n do
7     if f(xk+Δkdi) < f(xk)-p(Δk) and f(xk+Δkdi) < Min
8         Min ← f(xk+Δkdi)
9         dmin ← di
10        counter ← counter+1
11        if(counter = λ) break (Schleifenabbruch)
12    end if
13  end for
14  if(new Min found) do
15    xk+1 ← xk+Δkdmin (neuer Ausgangspunkt)
16    Δk+1 ← ϕΔk (Schrittweite erhöhen)
17  end if
18  else do
19    Δk+1 ← 0.5Δk (verringere Schrittweite)
20    if Δk+1 < Δtol stop ← true
21  end else
22  k ← k+1
23 end until
```

Abb. 10: Modifizierte Kompasssuche

3.2 Simulated Annealing

Simulated Annealing (SA) ist ein heuristisches Optimierungsverfahren aus dem Bereich des Operations Research (OR). Dieses Verfahren wird zur Suche nach einer approximativen Lösung von Optimierungsproblemen eingesetzt, die durch ihre hohe Komplexität das vollständige Bewerten aller Möglichkeiten und einfache mathematische Verfahren ausschließen. SA eignet sich für solche Optimierungsprobleme, bei welchen die zu

optimierenden Parameter nur einzelne, diskrete Werte annehmen können. Je nach Implementierung ist das Verfahren auch für geschlossene Zielfunktionen zu verwenden, bei denen die Parameter aus beliebigen reellen Zahlen stammen können.

Grundidee ist die Nachbildung des Abkühlungsprozesses von Metallen. Nach Erhitzen eines Metalls bedingt das langsame Auskühlen, dass die Moleküle sich ordnen und somit stabile Kristalle bilden können. Dadurch erreicht das Metall einen energiearmen (optimalen) Zustand (Entropie). Überträgt man das Abkühlen auf den Bereich der Optimierung, so entspricht die Temperatur der Wahrscheinlichkeit, mit der sich eine Zwischenlösung des Optimierungsprozesses auch verschlechtern darf. Die Metropolis-Strategie ist die Grundlage der simulierten Abkühlung. Im Gegensatz zu anderen Lokale Suche-Verfahren kann das Verfahren durch Vergleichen der Metropolis-Wahrscheinlichkeit (METROPOLIS et al. 1953) mit einer Zufallszahl zwischen 0 und 1 ohne zusätzliche Mechanismen (z. B. Mehrfachstart des Suchlaufs) ein lokales Optimum auch wieder verlassen und das globale Optimum auffinden. Es ist bewiesen, dass dieses Verfahren für unendlich kleine Änderungen des Temperaturwertes T immer das globale Minimum findet. Beispielhaft kann das Verfahren auf einen Bergsteiger bei der Suche nach dem höchsten Berg in einem Gebirge bei schlechten Sichtbedingungen übertragen werden. Der Bergsteiger läuft mit geringer Sichtweite von einem Ausgangspunkt unterschiedlich weit in verschiedene, zufällige Richtungen und prüft, wo eine höher gelegene Stelle vorliegt. Wird ein höherer Punkt gefunden, wird dieser zum neuen Ausgangspunkt und von dort die Suche fortgesetzt. Hier lassen sich Parallelen zur Kompasssuche erkennen, bei welcher ebenfalls in mehrere „Richtungen“ ausgehend von einer aktuellen Lösung nach einer besseren Lösung gesucht wird. Der wesentliche Unterschied besteht jedoch darin, dass die Kompasssuche lediglich Zielfunktionen mit unabhängigen Variablen aus dem Raum der reellen Zahlen verarbeiten kann. Zu Beginn seiner Suche hat der Bergsteiger noch viel Energie und er nimmt auch häufig Abwärtsbewegungen auf sich. Die Energie entspricht dabei der Temperatur beim Abkühlen von Metallen. Je länger die Suche andauert (die Temperatur sinkt), umso geringer ist die Motivation des Bergsteigers, Abwärtsbewegungen in Kauf zu nehmen. Die Motivation, auch Verschlechterungen (Abwärtsbewegungen) zu akzeptieren entspricht der Metropolis-Wahrscheinlichkeit. Diese nimmt mit sinkender Temperatur ab und mit kleineren Veränderungen bzw. Verschlechterungen zu. Die Veränderung (Δ) entspricht der absoluten Höhendifferenz von Ausgangspunkt und zufällig angesteuertem Punkt. Es werden somit schlechtere Lösungen mit

einer höheren Wahrscheinlichkeit akzeptiert, wenn die Temperatur/Motivation hoch ist und die Verschlechterung zum Ausgangspunkt gering ausfällt.

Der Algorithmus ist leicht auf ein gegebenes Optimierungsproblem zu übertragen. Es müssen lediglich die Kostenfunktion (C) und ein Mechanismus zum Variieren der Variablenkombination (im Gültigkeitsbereich) angepasst bzw. vorgegeben werden. Die Gestaltung der Kostenfunktion unterliegt dabei lediglich der Bedingung, dass die aktuelle Variablenkombination mit einem Zielfunktionswert aus $\mathbb{R}$ bewertet werden kann. Die Mechanismen zur Berechnung des Zielfunktionswertes kann völlig frei gestaltet werden. Es können beispielsweise Zufalls- oder Simulationskomponenten in die Zielfunktion integriert werden. Der dargestellte Algorithmus (Abb. 11) bricht ab, wenn eine definierte Mindesttemperatur (Ω) erreicht wird.

Initialisierung

r	Abkühlungsfunktion
V	Mechanismus zum Variieren der aktuellen Lösung
Ω	Grenzwerte für Abbruch
n	Anzahl „Tastversuche“
C(x)	Kostenfunktion in Abhängigkeit der Lösung x
C_{opt}, x_{opt}	Optimum, Optimale Lösungsstrategie
x_0, x_j	Start-/Ausgangslösung, Lösungsversuch j (Variablenkombination)
K	Boltzmann-Konstante
M	Metropolis-Wahrscheinlichkeit
T	Temperatur

Algorithmus (max)

```
1  while(T>Ω)
2       T ← r(T)
3       for j ← 1 to n do
4           x_j ← V(x_0)
5           Δ ← C(x_j) - C(x_0)
6           if Δ > 0 then
7               x_0 ← x_j
8           else
9               p ← Random [0...1]
10              m ← exp-(abs(Δ) / ( k * T )
11              if p < m then                    // Metropolis
12                   x_0 ← x_j
13              end if
14          end if
15          if C(x_0)> C_opt then
16               C_opt ← C(x_j)
17               x_opt ← x_j
18          end if
19      end for
20 end while
```

Abb. 11: Simulated Annealing.

Denkbar sind aber auch andere Kriterien, wie z. B. der Abbruch nach einer definierten Anzahl von Versuchen, die keine signifikante Verbesserung ergeben oder wenn eine definierte Anzahl von Versuchen mit Verbesserung erreicht ist. Liegen aufgrund der Struktur der Kostenfunktionen Informationen über das globale Optimum vor, kann auf Basis dieser Informationen zusätzlich ein effizientes Abbruchkriterium eingeführt werden. Ist die minimale Temperatur noch nicht unterschritten, aber ein bisher optimaler Wert (C_{opt}) in akzeptabler Nähe des bekannten Optimums ($abs(_{Copt\_bekannt}\text{-}C_{opt}) < x$), kann die Suche abgebrochen werden. Bei den Abkühlungsstrategien wird üblicherweise zwischen den in Tab. 3 dargestellten Grundformen unterschieden (GERDES et al. 2004). Darüber hinaus werden noch weitere erfolgreiche Abkühlungsstrategien beschrieben, die nicht monoton verlaufen (OSMAN 1993). Solche Verfahren führen nicht nur zu einer stetigen Abnahme der Temperatur, sondern temporär auch zu „Wiedererwärmungsphasen“, so dass ein oszillierendes Gleichgewicht zwischen Diversifikation und Intensivierung des Suchprozesses resultiert.

Tab. 3: Abkühlungsstrategien.

Konstant	$T(n) = k$, wobei k eine beliebige Konstante ist und n die Anzahl der Iterationen
Arithmetisch	$T(n) = T(n-1) - k$, wobei k eine beliebige Konstante ist. Allerdings muss darauf geachtet werden, dass die Anfangstemperatur T(0) so groß gewählt wird, dass die Temperatur selbst nach der maximalen Anzahl von Iterationsschritten nicht negativ wird.
Geometrisch	$T(n) = \alpha(n) \cdot T(n-1)$, wobei α typischerweise zwischen 0,8 und 0,99 liegt.

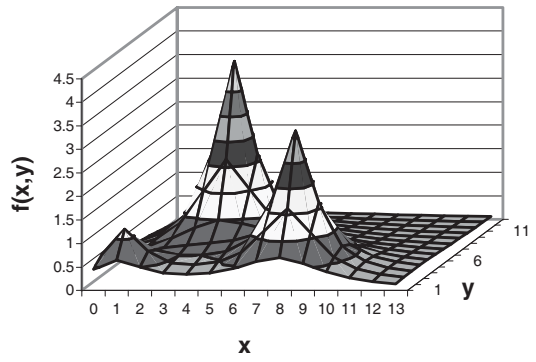
In der Literatur findet man die Prüfung von Δ gegen Null kombiniert mit dem Metropolis-Mechanismus (`if (Δ > 0 or Random[0..1] < m) then...`) (z. B. SEO 2005). Da jedoch nur eine der beiden Bedingungen erfüllt sein muss, sollte der Metropolis-Vergleich erst stattfinden wenn $\Delta \leq 0$ ist. Somit kann die Berechnung von m und das Ziehen der Zufallszahl in Fällen $\Delta > 0$ vermieden werden. Grundsätzlich wird bei diesem Verfahren ausgehend von einem initialen Lösungsvorschlag die Umgebung (Lösungsraum) „blind“ abgetastet. Wird eine bessere Lösung gefunden, wird diese zum Ausgangspunkt für eine erneute Suche. Durch den Metropolis-Mechanismus können aber auch in Abhängigkeit der Temperatur und von Δ Verschlechterungen akzeptiert werden, so dass lokale Optima verlassen werden können.

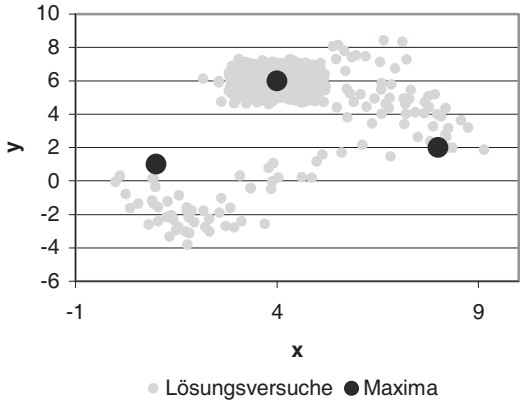
Je nach Optimierungsproblem wird eine Zielfunktion minimiert oder maximiert. Der Algorithmus 3 (Abb. 11) maximiert die gegebene Zielfunktion C. Der zuvor besprochene Algorithmus zur Kompasssuche minimiert die Zielfunktion. Je nach Problemstellung ist also

die Zielfunktion oder der Algorithmus abzuändern. Die Änderungen am Algorithmus oder der Zielfunktion sind dabei trivial. Beim vorstehenden Algorithmus (Abb. 11) müsste lediglich in den Zeilen 6 und 15 das „Größer-Als-Zeichen“ gegen ein „Kleiner-Als-Zeichen“ ausgetauscht werden.

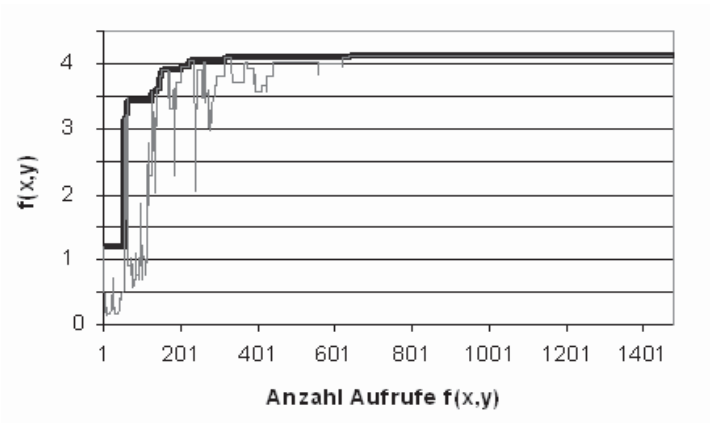
Abb. 12 zeigt eine zweidimensionale Kostenfunktion in Abhängigkeit von x und y (a) und den Verlauf des Lösungsprozesses (b und c). Es liegen neben dem globalen Maximum noch zwei weitere lokale Maxima vor. Die Startkonfiguration von x und y liegt nahe einem lokalen Maximum.

$$f(x,y)=\frac{1}{1+(1-x)^2+(1-y)^2}+\frac{4}{1+(4-x)^2+(6-y)^2}+\frac{3}{1+(8-x)^2+(2-y)^2}$$





a b



c

Abb. 12: Oben: zweidimensionale Zielfunktion, a: Visualisierung der Zielfunktion, b: Lösungsversuche und Maxima, c: Optimierungsverlauf aktuelle Lösungsgüte und bisher gefundenes Optimum über der Anzahl der Iterationen oder Aufrufe von f(x, y).

Die Optimumsuche mit Simulated Annealing ergibt, dass aufgrund der Möglichkeit in Abhängigkeit der Temperatur und dem Grad der Verschlechterung auch zwischenzeitlich weniger optimale Lösungen zu akzeptieren, der Algorithmus den beiden lokalen Maxima entkommen ist (Abb. 12. b). Dies geht auch aus Abb. 12 c hervor. Die schwarze Linie zeigt das aktuell gefundene Maximum, die graue Linie die akzeptierten Lösungsversuche. Während das gefundene Maximum stetig steigt, weist der Verlauf der verwendeten Lösungsmöglichkeiten anfangs hohe und im Verlauf der Optimierung immer schwächer werdende Ausschläge nach unten auf. Wie bei der Kompasssuche, ist auch die Initialisierung beim SA maßgeblich entscheidend für den Verlauf und Erfolg der Optimierung (SEO 2005). Bei ungünstigen Startparametern (hier x u. y) und schlecht gewählter Anfangstemperatur besteht die Möglichkeit, dass der Algorithmus erst bei einer sehr niedrigen Temperatur ein lokales Optimum findet und nicht mehr die „Energie“ besitzt, diesem zu entkommen. Um diesem Problem entgegenzuwirken, kann die Initialisierung entsprechend vorgenommen werden (z. B. grobes Durchsuchen des Lösungsraums) oder der Abkühlungsprozess wird so gesteuert, dass eine oder mehrere Wiedererwärmungen möglich sind.

Die Basisvariante des Simulated Annealing bietet viele Ansatzstellen für Modifikationen, um einerseits die Lösungsgeschwindigkeit und andererseits die Lösungsqualität zu verbessern. Verschiedene Metaheuristiken verwenden ein sog. Kurz- oder Langzeitgedächtnis (eine Speicherstruktur), um Informationen über den Suchverlauf zu sammeln und daraus Strategien zur Verbesserung des Suchprozesses abzuleiten (BETTINGER u. ZUH 2006). Diese Technik wird beispielsweise bei der Tabu-Suche (GLOVER 1990) oder der Ameisen-Kolonie-Optimierung eingesetzt. Ein sehr einfacher Gedächtnismechanismus wurde bereits durch das Speichern der absoluten besten Lösung (C_{opt}/x_{opt}, s. o.) implementiert. Diese Information kann z. B. dazu genutzt werden, ein Abbruchkriterium in der Form

```
Abbruch wenn C_opt seit n Iteration unverändert
```

oder

```
Abbruch wenn seit n Iterationen C_opt-C(x_0) < a  wobei a ∈ ℝ
```

zu konstruieren. Eine andere Möglichkeit, die Optimuminformation zu nutzen besteht darin, die Abkühlungsprozedur in Abhängigkeit von C_{opt} wie folgt zu regulieren:

```
Wenn seit n Iterationen C_opt-C(x_0) < a dann erhöhe T um x
Sonst vermindere T nach der vorgegebenen Abkühlungsstrategie
```

Durch diese Wiedererwärmungsstrategie kann die Robustheit des Algorithmus verbessert werden, da somit eine, in diesem Fall nötige Diversifikation der Suche ermöglicht wird.

Eine dritte denkbare Nutzung der Information zu C_{opt} und $C(x_0)$ ergibt sich am Ende eines Suchlaufs. Überschreitet die Differenz von C_{opt} und $C(x_{0\_end})$ einen Grenzwert ist die Suche suboptimal verlaufen und es kann automatisch ein Neustart unter Berücksichtigung aus dem ersten Lauf gesammelter Erfahrungen initialisiert und ausgeführt werden. Aus diesem Vorgehen leitet sich auch ein generelles Verfahren zur Initialisierung der Startparameter ab. Vor dem Hauptsuchprozess wird eine vereinfachte Form des Simulated Annealing mit einer wesentlich geringeren Laufzeit durchgeführt, um den Suchraum zu sondieren und daraus die Startparameter und andere Initialisierungsinformationen abzuleiten.

Initialisierung

r	Abkühlungsfunktion
V	Mechanismus zum Variieren der aktuellen Lösung
Ω, g_{limit}	Grenzwerte für Abbruch
n, g	Maximale Anzahl „Tastversuche“, Anzahl akzeptierter Versuche
C(x)	Kostenfunktion
C_{opt}, x_{opt}	Optimum, Optimale Lösungsstrategie
x_0, x_j	Start-/Ausgangslösung, Lösungsversuch j
K	Boltzmann-Konstante
M	Metropolis-Wahrscheinlichkeit
T	Temperatur
Mem	Speicher für Lösungsversuche (x_{j1}, ..., x_{jn}, $C(x_{j1})$, ..., $C(x_{jn})$)
A, a	Aspirationsfunktion mit Aspirationsgrenzwert a

Algorithmus (max)

```
1  while(T>Ω)
2      T ← r(T)
3      for j ← 1 to n do
4          x_j ← V(x_0), wobei x_j ∉ Mem oder A(x_j)<a
               updateMem(x_j)
5          Δ ← C(x_j) - C(x_0)
6          if Δ> 0 then
7              x_0 ← x_j
8              g ← g+1
9          else
10             p ← Random [0...1]
11             m ← exp-(abs(Δ) / ( k * T )
12             if p < m then                    // Metropolis
13                 x_0 ← x_j
14                 g ← g+1
15             end if
16         end if
17         if C(x_j)> Copt then
18             C_opt ← C(x_j)
19             x_opt ← x_j
```

```
20         end if
21         exit when g > g_limit
22       end for
23   end while
```

Abb. 13: SA mit Gedächtnisspeicher und erweitertem Abbruchkriterium.

Eine komplexere Gedächtnisstrategie besteht darin, eine definierte Anzahl der Vorgängerlösungen zu speichern und diese für eine über die Größe des Speichers festgelegte Zeitspanne für die Verwendung als neue Lösungsversuche zu sperren (vgl. Abb. 13, Zeile 4). Die Sperrung kann zusätzlich durch Einführen einer Aspirationsfunktion verhindert werden, um beispielsweise besonders gute Lösungen zuzulassen. Durch diese Gedächtnisstrategie können Rechenzeit verlängernde Zyklen und somit unnötige Aufrufe der Kostenfunktion vermieden werden. Weiterhin kann über die Dauer der Sperrung von Lösungen die Diversifikation der Suche beeinflusst werden.

3.3 Tabu Search

Tabu Search (TS) ist eine der am häufigsten verwendete Metaheuristik für kombinatorische Optimierungsprobleme. Die Grundidee stellte Glover 1986 vor (GLOVER 1986). Seit dem wird das Verfahren ständig erweitert und erfolgreich auf verschiedene Probleme wie dem QAP (Quadratic Assignment Problem), dem JSS-Problem (Job Shop Scheduling) oder dem VR (Vehicle Routing) angewendet (TAILLARD 1991, NOWICKI u. SMUTNICKI 1996, CORDEAU u. LAPORTE 2002, CORDEAU et al. 2008).

Der Algorithmus benutzt einen Gedächtnisspeicher zum Verlauf der bisherigen Suche, um einerseits lokalen Minima/Maxima zu entkommen und andererseits um eine Strategie zur Erforschung des Suchraums zu implementieren. Die Basisvariante des TS-Algorithmus (Abb. 14) verwendet ein lokales Suchverfahren zur Identifikation der bestmöglichen Verbesserung in direkter Nachbarschaft der aktuellen Lösung. Hinzu kommt ein sog. Kurzzeitgedächtnis, welches zur Vermeidung des „Hängenbleibens“ in lokalen Minima/Maxima und zur Vermeidung von Endlosschleifen dient. Das Kurzzeitgedächtnis wird auch als Tabu-Liste bezeichnet und speichert Informationen zu den kürzlich aufgesuchten Lösungen. Bei der Suche nach einer besseren Lösung in der Nachbarschaft sind die in der Tabu-Liste gespeicherten Lösungsvorschläge „tabu“, d. h. sie dürfen nicht als neue Lösungsvariante herangezogen werden. Die Liste erlaubter Lösungsversuche wird bei jeder Iteration nach der besten Lösung durchsucht, welche dann zur neuen aktuellen Lösung wird. Zusätzlich wird die neue Lösung in die Tabu-Liste aufgenommen.

Initialisierung

s	Startlösung / Aktuelle Lösung
N	Nachbarschaft der aktuellen Lösung s
k	Iterationszähler
TL, l	Tabuliste mit der Länge l
AB	Aspirations-Bedingung
GL	Liste gültige Lösungen

Algorithmus

```
s ← generiere Initiallösung()
initialisiere Tabu-Liste(TL1, ..., TL1)
k ← 0
while kein Abbruchkriterium erfüllt do
      GL(s, k) ← {s' ∈ N(s) | s nicht in TL, oder erfüllt mindestens eine AB}
      s ← WähleBesteLösungAus(GL(s, k))
      aktualisiere TL und AB
      k ← k + 1
end while
```

Abb. 14: Tabu Search.

Da der Speicher der Liste begrenzt ist, wird im Gegenzug eine Lösung aus der Tabu-Liste entfernt. Gewöhnlich wird dabei nach dem FIFO-Verfahren (**f**irst **i**n, **f**irst **o**ut) vorgegangen, so dass die älteste Lösung durch die neue Lösung ersetzt wird. Der Algorithmus stoppt, wenn das Abbruchkriterium erfüllt ist oder die Liste der erlaubten Lösungsversuche leer ist. Dies ist dann der Fall, wenn alle Lösungen in der Nachbarschaft der aktuellen Lösung tabu sind. Die Verwendung der Tabu-Liste unterbindet das Aufsuchen vorhergehender bester Lösungen. So wird auch das Auftreten von Endlosschleifen verhindert und der Algorithmus unter Umständen zu Auf- bzw. Abwärtsbewegungen (Verschlechterungen) gezwungen.

Über die Länge *l* der Tabu-Liste wird die Tabu-Dauer und somit das Suchverhalten beeinflusst. Durch eine kurze Tabu-Dauer wird die Suche auf kleinere Gebiete des gesamten Suchraums beschränkt. Eine längere Tabu-Dauer (größere Tabu-Liste) führt zu einer Exploration größerer Regionen des Suchraums. Neben der einfachen Variante einer fixen Listengröße werden auch Ansätze beschrieben, welche eine dynamische Gedächtnisgröße implementieren, was ein robusteres Suchverhalten bewirkt (BLUM u. ROLI 2003). Beispielsweise stellt Taillard (TAILLARD 1991) eine dynamische Steuerung der Tabu-Dauer vor, welche die Listengröße periodisch in gegebenen Schranken zufällig variiert. Eine andere Variante beschreiben Battiti und Tecchiolli (BATTITI u. TECCHIOLLI 1994) sowie Battiti und Protasi (BATTITI u. PROTASI 1997). Die Speichergröße wird erhöht, wenn Anzeichen für sich wiederholende Lösungsstrategien auftreten, da in diesem Fall eine höhere Diversifikation des Suchsets benötigt wird. Die Tabu-Dauer wird verkürzt (die Suche intensiviert), wenn keine

Verbesserung erreicht wird. Eine detaillierte Beschreibung zur Entwicklung weiterer dynamischer Prozeduren wie die ATL (active tabu list) zum Management der Tabu-Liste stellt Glover (GLOVER 1990) vor.

Neben dem Management der Speichergröße der Tabu-Liste, kann auch die Struktur der Liste den Suchprozess beeinflussen. Als Alternative zum Speichern der jeweiligen kompletten Lösung in der Tabu-Liste, können auch einzelne Attribute der jeweiligen Lösungen in einer oder mehreren Listen organisiert werden. Attribute einer Lösung sind z. B. Veränderungen zur aktuellen Lösung oder Unterschiede zwischen zwei Lösungen. Die Listen mit den einzelnen Attributen werden bei der Auswahl gültiger Lösungen herangezogen, um aus der direkten Nachbarschaft der aktuellen Lösung ungültige Varianten zu identifizieren. Das Speichern einzelner Attribute ist wesentlich effizienter, als eine Tabu-Liste mit den vollständigen Lösungen zu verwenden (BLUM u. ROLI 2003). Dabei besteht jedoch der Nachteil, dass Informationen verloren gehen und eventuell durch ein Attribut oder eine Attributkombination mehr als eine Lösung tabuisiert werden. Es könnten so auch bisher nicht besuchte, qualitativ gute Lösungen ausgeschlossen werden. Dieses Problem kann mit dem Einführen eines oder mehrerer sog. Aspirations-Kriterien umgangen werden. Aspirations-Kriterien erlauben es, tabuisierte Lösungen auszuwählen, wenn diese definierte Eigenschaften besitzen. Ein übliches Kriterium ist die Lösungsqualität. Liefert eine laut Tabu-Liste gesperrte Lösung einen besseren Zielfunktionswert als die aktuell beste Lösung, wird sie mit in den Satz der erlaubten Varianten aufgenommen.

Eine weitere Erweiterungsmöglichkeit besteht in der Einführung eines Langzeitgedächtnisses. In einem Langzeitspeicher werden Informationen zum gesamten Lösungsvorgang gespeichert, die dazu genutzt werden können, den weiteren Prozess strategisch zu leiten und die Suche dadurch effektiver zu gestalten. Ein qualitativer Langzeitspeicher dient der Identifikation und Extraktion guter (qualitativ hochwertiger) Lösungsversuche. Informationen über Lösungsversuche hoher Qualität können bei der Konstruktion neuer Lösungsversuche verwendet werden. In anderen Strategien wie der Ant Colony Optimization ist ein qualitativer Langzeitspeicher ein essenzieller Bestandteil des Verfahrens. Neben der Qualität von Lösungsversuchen, kann auch die Häufigkeit aufgesuchter Lösungen gespeichert werden. Die Information über Regionen im Suchraum, in denen der Algorithmus lange verweilt, können genutzt werden, um die Suche gezielt zu diversifizieren.

3.4 Genetischer Algorithmus

Der Genetische Algorithmus ist ein klassischer Vertreter der rekombinatorischen Verfahren. Im Vergleich zu den Algorithmen der Lokalen Suche werden bei dieser Verfahrenskategorie neue Lösungen nicht über eine Nachbarschaftsdefinition ermittelt, sondern durch die Kombination von Eigenschaften von mindestens zwei Lösungen der aktuellen Generation (endliche Menge mehrerer Lösungen). Da aus mehreren Lösungen eine neue Lösung generiert wird, muss zu jedem Iterationsschritt ein Pool verschiedener Lösungen vorhanden sein. Hier besteht ein weiterer Unterschied zur Lokalen Suche, dessen Vertreter in einem Iterationsschritt jeweils immer nur eine Lösung verwenden.

Die Funktionsweise des Genetischen Algorithmus richtet sich stark nach dem Grundprinzip der biologischen Evolution. Zu Beginn einer Optimumsuche wird zunächst eine ausreichend große Menge unterschiedlicher Individuen (Lösungen) meist zufällig erzeugt (vgl. Abb. 15 oben links) und die Fitness (Lösungsgüte) aller Individuen bestimmt. Diese bilden die erste Generation. Die Größe der Generation ist abhängig von der Größe des Suchraums.

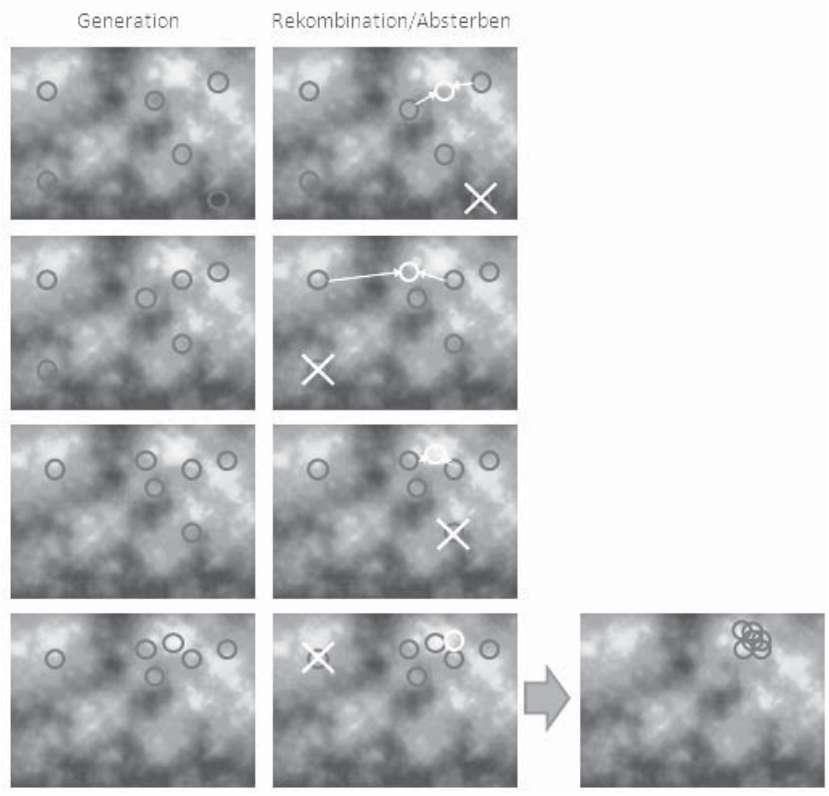


Abb. 15: Prinzip des Genetischen Algorithmus mit einer Population aus sechs Individuen (Kreise). Die Graustufe an der Position eines Individuums entspricht der Fitness (je heller umso höher die Fitness). In diesem Beispiel stirbt (x) und entsteht (Pfeile) je Generation jeweils nur ein Individuum.

Je größer der Suchraum ist, umso größer sollte auch der Generationsumfang gewählt werden, um einen ausreichend großen Genpool (Varianz der Lösungen) zu erzeugen, der die Grundlage für den Erfolg der Optimumsuche bildet. Diese Anfangspopulation wird anschließend über verschiedene Verfahren solange manipuliert, bis ein geeignetes Abbruchkriterium erreicht wird. Wie auch bei den Verfahren der Lokalen Suche sind zur Steuerung des Abbruchs verschiedene Strategien denkbar. Die einfachste besteht darin, eine definierte Anzahl neuer Generationen zu durchlaufen und dann das beste Individuum auszusuchen. Eine andere Möglichkeit die Laufzeit zu steuern, liegt in der Betrachtung der Fitness (Lösungsgüte) aller Individuen der Population. Haben alle Individuen eine Fitness erreicht, die nahe der bislang erreichten maximalen Fitness liegt, kann davon ausgegangen werden, dass eine qualitativ hochwertige Lösung gefunden wurde (vgl. Abb. 15 unten rechts). Im Rahmen der Manipulation der Individuen einer Generation werden drei grundlegende Schritte durchlaufen: Selektion, Rekombination und evtl. Mutation.

Durch die sog. Selektion erfolgt eine zufällige Auswahl von Lösungskandidaten aus der aktuellen Generation. Dabei werden Lösungskandidaten mit besseren Zielfunktionswerten mit einer höheren Wahrscheinlichkeit für die Rekombination ausgewählt. Aus der Rekombination entsteht ein neues Individuum, welches ein anderes, abgestorbenes Individuum ersetzt. Die Wahrscheinlichkeit, mit der ein Individuum stirbt, hängt wiederum von dessen Fitness ab. Es werden dementsprechend so viele Individuen neu gebildet, wie absterben, so dass der Populationsumfang über alle Generationen konstant bleibt.

Mittels der Rekombination werden die Gene (Lösungsparameter der ausgewählten Lösungen) guter Individuen kombiniert. Hierzu sind verschiedene grundlegende Verfahren anwendbar, die jedoch vor allem in Abhängigkeit des gegebenen Lösungsproblems ausgewählt und ggf. angepasst werden müssen. Eine Möglichkeit der Rekombination beruht darauf, einzelne Sequenzen aus dem Erbgut eines Individuums gegen die eines anderen Individuums auszutauschen (Abb. 16).

Abb. 16: Rekombination der Individuen A und B.

Eine Modifikation dieses Verfahrens besteht darin, die Länge der auszutauschenden Sequenzen mit der Fitness der beteiligten Individuen zu gewichten. Somit würde ein Individuum mit einer besseren Fitness mehr Gene vererben, als das Individuum mit der schlechteren Fitness. Zur Lösung des Problems des Handlungsreisenden[4] kann dieses einfache Verfahren nicht verwendet werden, da eine Stadt nur einmal ausgewählt werden darf und durch das einfache Austauschen von Sequenzen verschiedene Gene mehrfach vorkommen können. Es muss entsprechend auf problemspezifische Verfahren zur Erzeugung von Permutationen zurückgegriffen werden.

Durch Rekombination entstandene Individuen können anschließend einer zufälligen Veränderung (Mutation) ihrer Lösungsparameter unterzogen werden. Über die Mutationshäufigkeit und -intensität kann die Diversifikation der Suche gesteuert werden. Auch bei der Gestaltung der Mutation sind verschiedene Verfahren möglich. Es sind Ansätze beschrieben, die überhaupt keine Mutation vorsehen. Wird eine Mutationsfunktion integriert, kann diese so gestaltet werden, dass die Mutationsrate (Anteil der neuen Individuen, die mutieren sollen) und/oder die Mutationsintensität (Anteil der Gene eines einzelnen Individuums, die verändert werden) über alle Generationen konstant bleiben, oder variiert werden. Hier besteht eine Parallele zur Temperatursteuerung im Rahmen des Simulated Annealing. Der Verlauf der Temperatur beeinflusst den Grad der Diversifikation der Suche. Gleiches gilt für die Mutation bei den rekombinatorischen Verfahren. Starke und häufige Mutationen bedeuten eine hohe Diversifikation, wenige, schwache Mutationen erhöhen hingegen die Intensivierung des Suchprozesses.

Nach der Rekombination und der Mutation wird für jeden neuen Lösungskandidaten der aktuellen Generation anhand der Zielfunktion die Fitness bestimmt. Anschließend wird geprüft, ob die neue Generation ein Abbruchkriterium erfüllt. Ist dies nicht der Fall, wird die nächste Generation generiert und ausgewertet. Durch diese Vorgehensweise sind gleich mehrere Verfahren integriert, die dazu beitragen können, lokale Optima zu überwinden. Jede Generation besteht aus vielen Lösungen, so dass eine hohe Anfangsdiversität - ähnlich wie der Mehrfachstart Lokaler Suche-Verfahren - die Wahrscheinlichkeit erhöht, lokale Optima verlassen zu können. Durch die Mutation in Kombination mit der Rekombination, entstehen

[4] Das Problem des Handlungsreisenden besteht darin, dass eine endliche Menge von Städten aufgesucht werden muss und der Handlungsreisende am Ende zur Ausgangsstadt zurückkehren soll. Die dabei zurückgelegte Route soll möglichst kurz sein. Es gilt also die Reihenfolge der aufzusuchenden Städte so festzulegen, dass die Gesamtstrecke minimal wird und die Reise an der Ausgangsstadt endet.

im Verlauf der Optimumsuche immer wieder Individuen, die ebenfalls dazu beitragen, ein lokales Optimum überwinden zu können. Wie Abb. 17 zeigt, ist der Genetische Algorithmus grundsätzlich sehr einfach aufgebaut und somit einfach zu implementieren. Komplizierter ist die Auswahl und Anpassung geeigneter Rekombinations- und Mutationsstrategien, welche den Sucherfolg maßgeblich beeinflussen (WEICKER 2007). Der Algorithmus durchläuft zunächst zwei Selektionsabschnitte. Hier werden Individuen ausgewählt, die sich fortpflanzen bzw. absterben. Danach erfolgt die Rekombination der ausgewählten Individuen und eine anschließende Mutation der neu entstandenen Individuen. Diese werden abschließend mit der Fitnessfunktion bewertet und anschließend das beste Individuum (höchste Fitness) der neuen Generation ($G^{(n)}$) ermittelt.

Initialisierung

$G^{(n)}$	Startpopulation bzw. aktuelle Generation mit dem Umfang n
F	Fitnessfunktion
S	Selektionsfunktion
R	Rekombinationsfunktion
M	Mutationsfunktion
Is, Id	Menge an Individuen für Rekombination/Auslese

Algorithmus

```
1   s ← generiere Initialpopulation()
2   while kein Abbruchkriterium erfüllt do
3     Is ← S(G)
4     Id ← S(G)
5     Id← R(Is)
6     Id← M(Id)
7     for j ← 1 to Anzahl(Id) do
8           fitness_Idj ← F(Id_j)
9     end for
10 end while
11 return Max(G)
```

Abb. 17: Genetischer Algorithmus.

3.5 Parallelisierung von Metaheuristiken

Die Parallelisierung von Metaheuristiken ist im Bereich der kombinatorischen Optimierung ein interessantes Forschungsfeld, da durch parallele Ansätze sowohl die Lösungsqualität verbessert als auch die Rechenzeit ohne Qualitätseinbußen verringert werden kann. Gerade im Bereich der kombinatorischen Optimierung ist die Reduktion der Rechenzeit von Bedeutung, da der iterative Charakter und der meist hohe Komplexitätsgrad der zu lösenden Probleme viele Systemressourcen benötigen. Hinzu kommt, dass die Entwicklung paralleler Hardware

rasant voranschreitet, so dass auch softwareseitig entsprechende Implementierungen vorgenommen werden müssen, um den vollen Leistungsumfang auszuschöpfen. Crainic und Toulouse (CRAINIC u. TOULOUSE 2003) stellen ein Klassifizierungsschema zu Parallelisierungsmethoden für Metaheuristiken vor. Die Autoren definieren drei Typen zur Abgrenzung verschiedener Verfahren zur Parallelisierung: *Type1*, *Type2* und *Type3*. Die Klassifizierung basiert auf der Strategie zur Aufteilung von Berechnungen auf mehrere Prozessoren (und/oder Prozessorkerne).

Type1-Parallelisierung

Bei Verfahren der Type1-Parallelisierung wird der rechenintensive Teil eines heuristischen Verfahrens auf mehrere Prozessoren verteilt und durch einen Master-Prozessor koordiniert. Der einzige Zweck dieses Parallelisierungstyps ist es, die Gesamtrechenzeit herabzusetzen (JAMES et al. 2009). Einen guten Parallelisierungsansatz bietet die Exploration der direkten Nachbarschaft der aktuellen Lösung bei den Verfahren der Lokalen Suche. Es liegt nahe, die Nachbarschaft in so viele Unter-Suchräume aufzuteilen, wie Prozessoren zur Verfügung stehen, und von jedem Prozessor nur den zugeteilten Suchraum nach der besten Lösung durchsuchen zu lassen (Abb. 18). Hat jeder Prozessor seine Suche beendet, wird unter den jeweils besten Teil-Lösungen die beste ausgewählt und zur neuen aktuellen Lösung erklärt, von welcher die aufgeteilte Suche von neuem startet. Nach diesem Prinzip arbeitende Low-Level-Parallelisierungsstrategien stellen z. B. Taillard (TAILLARD 1991), Chakrapani und Skorin-Kapov (CHAKRAPANI u. SKORIN-KAPOV 1993) und James, Rego sowie Glover (JAMES et al. 2005) vor.

Type2-Parallelisierung

Strategien der Type2-Parallelisierung werden auch Zerlegungs-Strategien genannt, da bei diesen Strategien der gesamte Suchraum zerlegt wird. Die Optimierung in Abhängigkeit von nur einer Untermenge der Entscheidungsvariablen verringert den Suchraum. Die Optimierung in den Teil-Suchräumen wird wiederum auf mehrere Prozessoren verteilt und die Kombination der einzelnen Entscheidungsvariablen zu einer gültigen Lösung und das Auffinden der besten Lösung übernimmt ein Masterprozess. Bei diesem Vorgehen wird jedoch die Grundstruktur der sequentiellen Algorithmen abgeändert, wodurch die Lösungsqualität maßgeblich beeinflusst werden kann (JAMES et al. 2009).

Type3-Parallelisierung

Bei den Type3-Strategien werden grundsätzlich zwei oder mehr identische Suchalgorithmen auf mehreren Recheneinheiten gestartet. Die einzelnen Suchen nach einem Optimum können dabei völlig unabhängig erfolgen. Oder es wird eine Schnittstelle implementiert, die Interaktionen zwischen den einzelnen Suchläufen während des Optimierungsprozesses zulässt (z. B. der Austausch der jeweils aktuellen besten Lösungen). Am Ende des Gesamtsuchlaufs (nachdem alle einzelnen Prozesse beendet sind) wird bei beiden Verfahren, der Unabhängigen und der Kooperativen Suche, die beste Lösung aus den von jedem Prozess gelieferten Einzellösungen gewählt. Bei diesen Verfahren steht weniger die Beschleunigung der Gesamtsuche, sondern vielmehr die Verbesserung der Lösungsqualität im Vordergrund.

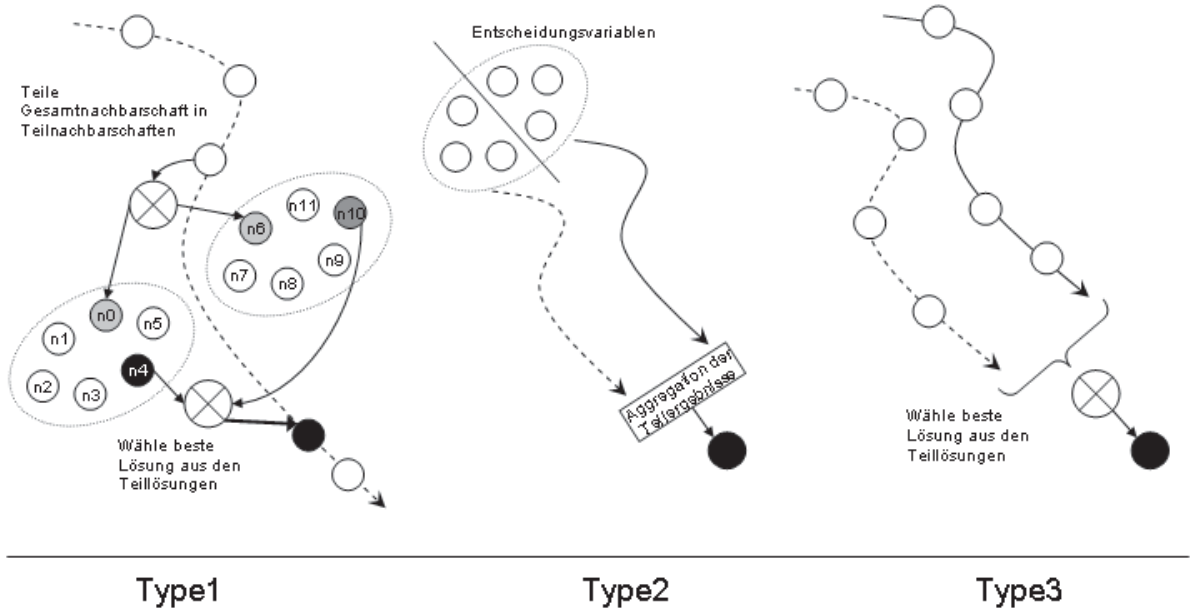


Abb. 18: Parallelisierungs-Strategien nach Crainic und Toulouse (CRAINIC u. TOULOUSE 2003)

Auch für die drei Basistypen zur Parallelisierung metaheuristischer Algorithmen gilt, dass Hybride und Kombinationen, je nach Problemstellung und Hardwareausstattung möglich und sinnvoll sind. Denkbar wäre z. B. eine Type1-Parallelisierung wiederum simultan im Sinne der Type3-Parallelisierung zu starten. Bei der Implementierung und Anwendung paralleler Verfahren ist jedoch immer zu prüfen, ob der durch den Parallelisierungs-Mechanismus auftretende Overhead sich nicht insgesamt negativ auf die Gesamtlaufzeit des Suchprozesses auswirkt. Die Verwaltung und Synchronisation mehrerer Threads benötigt ebenfalls Rechenzeit, welche nicht für die eigentliche Lösungssuche zur Verfügung steht und diese verlangsamt.

3.5.1 Parallele Kompasssuche

Das Verfahren der Kompasssuche bietet mehrere Möglichkeiten, die Berechnungen der Zielfunktion bei gegebener Hardwarestruktur simultan durchzuführen. Somit kann eine Verbesserung der Geschwindigkeit des Optimierungsprozesses erreicht werden. Dabei bieten sich grundsätzlich zwei Strategien an. Zum einen könnte man so viele Threads erzeugen, dass für jede Suchrichtung ein Thread die Berechnung der Zielfunktion übernimmt. Oder es wird eine feste Anzahl an Threads erzeugt, die die Berechnung für mehrere Suchrichtungen übernehmen. Die Ergebnisse zu jeder Suchrichtung müssen zentral gespeichert und ausgewertet werden. Bei der Nutzung paralleler Lösungen ist jedoch zu beachten, dass die Threadstruktur einen Overhead erzeugt, der im Verhältnis zur Komplexität der Zielfunktion bewertet werden muss. Zur Initialisierung und Synchronisierung einzelner Threads wird Rechenzeit benötigt. Die Nutzung eines threadbasierten Optimierungssystem ist nur dann zu empfehlen, wenn die benötigte CPU-Zeit zur Threadverwaltung durch eine deutlich verbesserte Lösungsqualität und/oder eine insgesamt verbesserte Lösungsgeschwindigkeit gerechtfertigt ist. Ein weiterer Einsatzbereich des parallelen Rechnens bietet sich bei Optimierungsproblemen, von denen bekannt ist oder vermutet wird, dass neben einem globalen Minimum weitere lokale Minima existieren. Eine gängige Strategie, um dieses Problem zu bewältigen, ist ein Mehrfachstart des Optimierungsprozesses mit zufällig oder systematisch gewählten Startkoordinaten. Die einzelnen Optimierungsläufe können wieder auf einzelne Threads ausgeteilt und somit simultan gerechnet werden (Type3).

3.5.2 PSA - Paralleles Simulated Annealing

In diesem Abschnitt wird eine Variante des Simulated Annealing entwickelt, welche um eine Type1-Parallelisierung erweitert wird. Hierzu wird die innere Schleife parallelisiert, in dem sie auf mehrere Threads (InnerLoopThread, vgl. Abb. 19) aufgeteilt wird. Die partiellen Lösungen der einzelnen Nachbarschaftssuchen werden nach jedem Durchlauf aller Threads synchronisiert, so dass die folgende Suche bei der bis dahin besten Lösung aus allen parallelen Nachbarschaftssuchen startet. Nach der Synchronisation der Teilergebnisse wird die Temperatur verringert und die Threads erneut gestartet. Dieser Vorgang wird solange wiederholt, bis beispielsweise die spezifizierte Minimumtemperatur (Ω) erreicht wird oder ein anderes Abbruchkriterium greift. Je nach verfügbarer Prozessor(-kern)-Anzahl kann so die absolute Laufzeit der inneren Schleife reduziert werden. Denkbar ist auch die Iterationsanzahl der inneren Schleife (*nlimit*) zu erhöhen, so dass bei gleicher Laufzeit im Vergleich zur

sequentiellen Variante die Nachbarschaft intensiver durchsucht wird. Innerhalb der Threads (InnerLoopThread) erfolgt der Durchlauf einer For-Schleife. Bei Durchlauf dieser Schleife werden neue Lösungen aus der Nachbarschaft der aktuellen Lösung generiert und geprüft, ob sie besser sind als die aktuelle Lösung. Ist dies der Fall oder greift der Metropolis-Mechanismus, wird die neue Lösung zur aktuellen Lösung innerhalb des Threads (x_{0_t}). Weiterhin merkt sich jeder Thread seine beste Lösung (x_{max_t}) und nach jedem Durchlauf aller Threads wird geprüft, ob die globale beste Lösung (x_{max}) durch eine bessere aus den jeweiligen Threads ersetzt werden kann. Die letzte aktuelle Lösung muss nicht zwingend die ermittelte, beste Lösung eines Thread-Durchlaufs sein. Die neue Startlösung (x_0) für alle Threads ist die beste aktuelle Lösung aus allen Threads der vorherigen Iteration.

Initialisierung

x_0, x_{max}, x_{0_t}, x_{max_t}, x_j	Start-/Ausgangslösung, beste Lösung, Ausgangslösung eines Threads, beste Lösung eines Threads, aktuelle Lösung in der Schleife eines Threads
r(T)	Abkühlungsfunktion in Abhängigkeit der aktuellen Temperatur T
Ω	Grenzwerte für Abbruch bzw. minimale Temperatur
n	Anzahl „Tastversuche“
t_i	i-ter InnerLoopThread
p	Grad der Parallelisierung
C(x)	Kostenfunktion in Abhängigkeit einer Lösung x
N(x)	Nachbarschaftsfunktion zu einer Lösung x
K	Boltzmann-Konstante
T	Temperatur
M(T, Δ)	Metropolis-Wahrscheinlichkeit abhängig von Verschlechterung (Δ) T sowie K
Δ	Differenz von $C(x_j)$ und $C(x_{0\_t})$

Algorithmus (max)

```
1  x0 ← generiere Startlösung
2  xmax ← x0
3  while(T>Ω)
4       for i ← 1 to p do
5           ti.start(T,x0)
6       end for
7       wait() //warte bis alle Threads die start-Methode beendet haben
8       x0 ← Max(x0_t_i, i ←1..p) //beste aktuelle Lösung aus allen Threads
9       if C(Max(xmax_t_i, i ←1..p))> C(xmax) then
10          xmax ←Max(xmax_t_i, i ←1..p)
11      end if
12      T ← r(T)
13 end while

InnerLoopThread{
1  start(T,x0)
2       x0_t ← x0
3       xmax_t ← x0
4       for j ←1 to n do
5           xj ← N(x0_t)
```

```
6           Δ ← C(x_j)-C(x_0_t)
7           if Δ>0 then
8               x_0_t ← x_j
9           else
10              if Random[0..1] < M(Δ,T) then
11                  x_0_t ← x_j
12              end if
13          end if
14          if C(x_0_t)>C(x_max_t) then
15              x_max_t← x_0_t
16          end if
17      end for
18  end start
```

Abb. 19: Paralleles Simulated Annealing.

Konvergiert der Thread, sollte die bis dahin ermittelte beste Lösung sich kaum von der aktuellen Lösung unterscheiden (s. o.). Die optimale Lösung liegt nach Durchlauf der while-Schleife dementsprechend mit x_{max} vor. Ist die Differenz von $C(x_0)$ und $C(x_{max})$ am Ende des Suchprozesses sehr hoch, deutet dies darauf hin, dass der Suchlauf nicht optimal verlaufen ist (z. B. trotz Metropolis-Mechanismus ein lokales Minimum nicht verlassen werden konnte). Ist dies der Fall, sollte der Algorithmus erneut mit veränderten Startparametern (z. B. x_{max}) gestartet werden.

Eine weitere Modifikation des Simulated Annealing kann dadurch erfolgen, dass den einzelnen Threads nicht die gleiche Funktion (N) zum Generieren neuer Lösungen übergeben wird, sondern verschiedene Funktionen implementiert werden und in einem oder mehreren Threads zum Einsatz kommen. Dadurch wird der Suchraum von einem Thread mit einer intensivierenden Nachbarschaftsfunktion einmal sehr gründlich um eine aktuelle Lösung durchsucht. In einem zweiten Thread, dessen Nachbarschaftsfunktion einen größeren Teil des Suchraums abdeckt, erfolgt eine gröbere, aber umfassendere Suche, so dass in bislang unberücksichtigten Bereichen des Suchraums neue evtl. bessere Lösungen ermittelt werden können. Durch diese Strategie kann die Robustheit des Verfahrens verbessert werden. Darüber hinaus können weitere Modifikationen, so auch die in Kap. 3.2 vorgestellten Veränderungen, mit in den parallelen Ansatz integriert werden.

3.5.3 PGA - Paralleler Genetischer Algorithmus

Die simultane Verwendung und Kombination mehrerer Individuen einer Population in einem Iterationsschritt des Genetischen Algorithmus führt zu einer hochgradigen Parallelisierbarkeit. Gerade bei komplexen, rechenaufwendigen Zielfunktionen kann die Parallelisierung der Berechnung des Zielfunktionswertes die Suchgeschwindigkeit deutlich steigern. Die

technische Realisierung eines parallelen Genetischen Algorithmus (PGA, vgl. Abb. 20) gestaltet sich durch das Verwenden einer Population relativ einfach und ist der Type1-Parallelisierung zuzuordnen. Im Rahmen der Rekombination und Mutation werden neue Individuen erzeugt, welche anschließend mittels einer Zielfunktion bewertet werden müssen. Die Rekombination und die Bewertung der neuen Individuen kann parallel in mehreren Threads durchgeführt werden. Aufgrund der Struktur des Genetischen Algorithmus ist kein zusätzlicher Aufwand notwendig, um die gleichzeitig berechneten Ergebnisse zu synchronisieren. Beim PSA müssen die threadweise produzierten Teillösungen abgeglichen werden, um die insgesamt beste Lösung zu bestimmen. Dieses Vorgehen ist beim Genetischen Algorithmus schon implementiert, da bei der Generierung einer neuen Generation mit hoher Wahrscheinlichkeit die besten Lösungen für die Rekombination ausgewählt werden.

Initialisierung

$G^{(n)}$	Startpopulation bzw. aktuelle Generation mit dem Umfang n
F	Fitnessfunktion
S	Selektionsfunktion
R	Rekombinationsfunktion
M	Mutationsfunktion
Is, Id	Menge an Individuen für Rekombination, Auslese
t_i	i-ter Thread zur Berechnung der Fitness (FitnessThread)

Algorithmus

```
1  s ← generiere Initialpopulation()
2  while kein Abbruchkriterium erfüllt do
3     Is ← S(G)
4     Id ← S(G)
5     Id ← R(Is)
6     Id ← M(Id)
7     for j ← 1 to Anzahl(Id) do
8           tj.start(F,Idj)
9     end for
10    wait() //warte bis alle Threads die start-Methode beendet haben
11 end while
12 return Max(G)

FitnessThread
1     start(F,Idi)
2           fitnessIdj ← F(Idj)
3     end start
```

Abb. 20:Paralleler Genetischer Algorithmus (PGA).

Sind alle Threads zur Berechnung der Fitness (Zielfunktionswerte) beendet, kann der Algorithmus sofort mit der nächsten Generation fortfahren. Algorithmus 8 verdeutlicht den Ablauf des parallelen Genetischen Algorithmus.

3.6 Implementierung der Algorithmen

Die Implementierung der vorgestellten Algorithmen erfolgt sowohl für den anschließenden Verfahrensvergleich als auch zur Einbindung in das Entscheidungsunterstützungssystem in der Programmiersprache Java, da diese aufgrund der objektorientierten und paralleles Rechnen unterstützenden Architektur zu diesem Zweck gut geeignet ist. Denkbar wäre auch das Einbinden bestehender Optimierungssysteme. Die Simulationssprache SIMUL_R (RUZICKA 1988, RUZICKA 1989, RUZICKA 1993) bietet hervorragende Möglichkeiten, komplexe Systeme abzubilden und in Abhängigkeit definierter Parameter vom Modell abhängige Zielfunktionen zu optimieren. Die Firma SAS bietet eine Software SAS/OR an, welche unter Einbeziehung der Modellierungssprache Optmodel ebenfalls ein leistungsfähiges Optimierungstool darstellt (SAS 2008). Cahon stellt die Bibliothek ParadiseEO mit verschiedenen Implementierungen paralleler Methaheuristiken vor (CAHON et al. 2004). Das Nutzen externer Bibliotheken erfordert jedoch z. T. komplexe Interface-Strukturen, die die Laufzeit des Gesamtsystems negativ beeinflussen und einen größeren Installations- und Wartungsaufwand verursachen. Nicht zu vernachlässigen sind auch die Kosten der meist kommerziellen Softwareprodukte. Darüber hinaus sind in den genannten Produkten oft nur Standardverfahren abrufbar, so dass die hier vorgestellten Modifikationen teilweise nicht umgesetzt werden könnten.

Die notwendigen Klassen werden in einer Klassenbibliothek zusammengefasst. Der Aufbau dieser Bibliothek wird in Abb. 21 verdeutlicht. Die eigentlichen Algorithmen werden strikt von den benötigten Zusatzklassen (Common) getrennt, welche teilweise von mehreren Algorithmen verwendet werden. Die Zusatzklassen müssen größtenteils ein vorgegebenes Interface implementieren, da die jeweiligen Algorithmen dieses im Methodenaufruf zur Minimierung bzw. Maximierung voraussetzen. Dieses Vorgehen ermöglicht es, auf verschiedenste Optimierungsprobleme angepasste Klassen zur Variation der aktuellen Lösung oder zur Beschreibung der Kosten- (bzw. Ziel-) Funktion nachträglich zu erstellen und somit eine hohe Flexibilität der Bibliothek zu erreichen (CAHON et al. 2004). Alle Klassen, welche die Funktionalität eines Optimierungsalgorithmus beinhalten, implementieren das Interface *OptimizationAlgorithm.* Dieses Interface „erzwingt“ in der implementierenden Klasse die im

Folgenden beschriebenen, öffentlichen (außerhalb der Bibliothek aufrufbaren) Methoden. Die Klasse *ParallelSA* benötigt zusätzlich die Hilfsklasse *InnerLoopThread*. Diese Klasse erweitert die Klasse Thread und stellt die Suchmethodik zur Verfügung, welche nach dem parallelen Simulated Annealing in der inneren Schleife abläuft.

Optimization (B)		
	Algorithms (P)	
		ParallelSA (C ← *OptimizationAlgorithm*)
		InnerLoopThread (C ← Thread)
		CompassSearch (C ← *OptimizationAlgorithm*)
		*ParallelGA (*C ← *OptimizationAlgorithm)*
		FitnessThread(C ← Thread)
		GA(C ← *OptimizationAlgorithm*)
		SAM (C ← *OptimizationAlgorithm*)
		TabuSearch (C)
		SimulatedAnnealing (C ← *OptimizationAlgorithm*)
	Common (P)	
		OptimizationAlgorithm (I) *CostFunction (I)* *VariationFunktion (I)* *RecombinationFunktion (I)* *MutationFunction (i)* *TabuListElement (C)* *TabuList (C)* *SimpleVariationFunction (C ← VariationFunktion)* *VisualisationPanel (I)* *SimpleVisualisationPanel (C ← VisualisationPanel)* *OutputWriter (C)* *NeighbourSet (I)* *SimpleNeighbourSet(C ← NeighbourSet)* *AnnealingFunction (I)* *ExponentialAnnealingFunction (C ← AnnealingFunction)* *LinearlAnnealingFunction (C ← AnnealingFunction)* *CosExpAnnealingFunction (C ← AnnealingFunction)* *RestrictionChecker (I)*

B= Bibliothek, P= Package, C= Class, I= Interface, ← = implementiert nachstehendes Interface oder erweitert die Klasse

Abb. 21: Aufbau der Java-Klassenbibliothek zur Kapselung der notwendigen Klassen zum Aufbau eines Optimierungssystems.

3.6.1 Interface OptimizationAlgorithm

Die Methode *optimize* stellt in den konkreten Klassen die eigentliche Suchlogik zur Verfügung und liefert keinen direkten Rückgabewert (**void**). Hierzu erwartet sie einen

Parameter (type) vom Type **int**, der vorgibt, ob die spezifizierte Kostenfunktion (cf) maximiert oder minimiert werden soll.

```
void optimize(int type, double[] startparam, CostFunktion cf,
      VariationFunction vf, VisualisationPanel vp, OutputWriter out){}

int getNCallsCF(){}

double[] getBestSolution(){}

double getOptimum(){}

long getTimeUsed(){}
```

Die Variation oder Auswahl eines direkten Nachbarn der aktuellen Lösung erfolgt über eine Implementierung des Interfaces *VariationFunction* (vf). Die Konfiguration der Startparameter (startparam) wird über ein Array des Typs **double** spezifiziert. Beinhaltet das Optimierungsproblem ganzzahlige, binäre, ordinal- oder nominal-skalierte Parameter, werden diese ebenfalls als **double** übergeben und entsprechend codiert. Lediglich bei der Implementierung der Variations-Klasse ist der Parameter-Typ zu beachten und die Variation oder Rekombination entsprechend vorzunehmen. So ist es möglich, kombinatorische und stetige Optimierungsprobleme gleichermaßen dem jeweiligen Algorithmus zu übergeben, ohne an der Implementierung der Algorithmen etwas zu ändern. Wenn eine visuelle Veranschaulichung während des Suchprozesses gewünscht ist, kann der Methode eine Implementierung des Interfaces *VisualisationPanel* übergeben werden. Wird **null** übergeben erfolgt keine visuelle Ausgabe. Gerade für das Debugging und den Vergleich von verschiedenen Implementierungen ist es notwendig, Informationen zum Suchprozess zu jedem Iterationsschritt (nach jeder Variation der aktuellen Lösung) auszugeben. Dies übernimmt der *OutputWriter* (out), welcher Informationen zum aktuellen Iterationsschritt in eine ASCII-Datei schreibt. Wird out mit einem Null-Pointer initialisiert erfolgt keine Ausgabe. Die Methode *getNCallsCF* gibt die Anzahl der Aufrufe der Kostenfunktion nach einem Optimierungslauf an. Der Aufruf der Methode *getBestSolution* liefert ein Array des Typs **double,** welches die beste ermittelte Lösung kodiert. Der Aufruf von g*etTimeUsed* gibt die für den Optimierungsprozess benötigte Rechenzeit in Millisekunden zurück.

Die Implementierung des parallelen Simulated Annealing weist hinsichtlich der Methode *optimize()* den Unterschied zu den anderen Implementierungen auf, dass von der Kostenfunktion (*CostFunktion*) und der Variationsfunktion (*VariationFunction*) jeweils so viele Instanzen in einem Array des jeweiligen Typs übergeben werden, wie Threads gestartet

werden, um die simultane Suche in einem Iterationsschritt durchzuführen. Alle konkreten Implementierungen der jeweiligen Algorithmen beinhalten somit die Methoden des Interfaces *OptimizationAlgorithm*. Darüber hinaus werden noch eine Reihe von *get*- bzw. *set*-Methoden implementiert, die zum Großteil zur Einstellung der algorithmus-spezifischen Parameter (Metropolis-Wahrscheinlichkeit, Abbruchbedingungen etc.) dienen oder Modifikationen am Algorithmus aktivieren oder deaktivieren.

3.6.2 Zusatzklassen (Common)

Die Implementierung der beiden Interfaces *CostFunction* und *VariationFunction* sind mitentscheidend für den Erfolg der Suchalgorithmen. Gerade bei der *VariationFunction* sind die Modellparametereigenschaften und die Definition der Nachbarschaft zu einer aktuellen Lösung zu beachten und die Implementierung unter Berücksichtigung des Algorithmus entsprechend vorzunehmen. Das Interface *VariationFunction* gibt lediglich eine Methode vor:

```
double[] vary(double[] actualsolution){}
```

Diese Methode erwartet ein Array vom Type **double**, welches die aktuelle Lösung im Suchprozess kodiert und variiert diese entsprechend der konkreten Implementierung. Für ein stetiges Optimierungsproblem mit zwei Parametern könnten diese z. B. um einen zufälligen Wert aus einem definierten Intervall verändert werden. Im nachstehenden Beispiel (Abb. 22) ist eine einfache Implementierung der Methode *vary* dargestellt. Diese Methode variiert zufällig alle Parameter des übergebenen Array *actualsolution* mit dem Wert *value* innerhalb des Intervalls $\{\text{value} \in \mathbb{R} | \text{ factor} \leq \text{value} \leq \text{ factor; value} \neq 0\}$.

```
public double[] vary(double[] actualsolution){
      double dir;
      double value=0;
      double[] result=new double[param.length];
      for(int i=0; i<param.length; i++){
            value=0.0;
            if(Math.random()<0.5) dir=-1.0;
            else dir=1.0;
            while(value==0.0){
                value=dir*Math.random()*factor;
            }
            result[i]=param[i]+value;
      }
      return result;
}
```

Abb. 22: Implementierung der Methode vary.

Soll ein kombinatorisches Optimierungsproblem gelöst werden, muss die Methode eine Parameterkombination finden, welche der aktuellen Lösung ähnlich ist (in der direkten

Nachbarschaft liegt). Unterliegt das Optimierungsproblem gegebenen Restriktionen, muss die Methode *vary* prüfen, ob eine Parameterkombination diese verletzt und ggf. eine neue Parameterkombination ermitteln. Wurde eine gültige Kombination gefunden, wird diese als Array (**double**) an die aufrufende Methode zurückgegeben.

Das Interface *CostFunction* gibt vor, die drei folgenden Methoden zu implementieren:

```
void init(){}

int getNCalls(){}

double getCosts(double[] actualsolution){}
```

Die Methode *init* wird zu Beginn der Methode *optimize* aufgerufen und setzt vor allem den internen Aufrufzähler auf 0. Der aktuelle Wert dieses Zählers wird von der Methode *getNCalls* zurückgegeben, welche in allen Implementierungen von *optimize* nach einem Suchdurchlauf aufgerufen wird. Wird eine *OutputWriter*-Instanz der Methode *optimize* übergeben, ruft diese nach jedem Aufruf von *vary* ebenfalls die Methode *getNCalls* auf.

Die Ermittlung der Kosten oder des Nutzens einer Lösung erfolgt durch den Aufruf der Methode getCosts. Der Methode wird die als double-Array kodierte aktuelle Lösung übergeben. Wenn es das gegebene Optimierungsproblem ermöglicht, sollte das Ergebnis der Funktion zwischen 0 und 1 normiert werden, da so die Parametrisierung der Algorithmen und vor allem der Abbruchkriterien vereinfacht wird.

Die Klasse *TabuList* wird von den Implementierungen der Algorithmen TS (TabuSearch) und SAM (Simulated Annealing with Memory) benötigt. Diese Liste stellt eine Speicherstruktur zur Verfügung, die Informationen zu Lösungen beinhaltet, welche im Laufe des Suchprozesses vom jeweiligen Algorithmus als gültige aktuelle Lösung ermittelt werden. Diese Lösungen sind für eine definierbare Zeitspanne tabuisiert. Die Liste arbeitet nach dem FIFO-Prinzip, welches besagt, dass das zuerst aufgenommene Element auch als erstes wieder aus der Liste entfernt wird. Die Einträge in die Tabuliste müssen Objekte vom Typ *TabuListElement* sein oder dieses erweitern.

Für alle Varianten des Simulated Annealing sind das Interface und die abgeleiteten Klassen *AnnealingFunction* besonders hervorzuheben. Wie vorstehend (Kap. 3.2) erläutert, hat die Abkühlungsstrategie einen entscheidenden Einfluss auf das Lösungsverhalten des jeweiligen Verfahrens. Das Interface *AnnealingFunction* beinhaltet lediglich die Methode `double anneal(double t).` Diese Methode erwartet die aktuelle Temperatur (t) und liefert

entsprechend der Abkühlungsstrategie die neue Temperatur. In die Bibliothek wurden drei Klassen des Typs *AnnealingFunction* aufgenommen:

ExponentialAnnealingFunction

LinearAnnealingFunction

CosExpAnnealingFunction

Die Temperaturen werden durch die exponentielle und lineare Funktion stetig abgesenkt. Die Funktion *CosExpFunction* hingegen führt zu einer Oszillation um ein exponentiell fallendes Mittel (Abb. 23). Das Interface *RestrictionChecker* beinhaltet nur die Methode *isValid.* Diese Methode erwartet als Übergabeparameter ein Array vom Typ **double** und liefert als Ergebnis einen booleschen Wert. Das Ergebnis zeigt an, ob die zu prüfende Parameterkombination (aktuelle Lösung) eine oder mehrere Restriktionen verletzt. Bei der Implementierung konkreter Checker-Klassen ist somit eine große Flexibilität hinsichtlich der Berücksichtigung einer oder mehrerer Restriktionen gegeben.

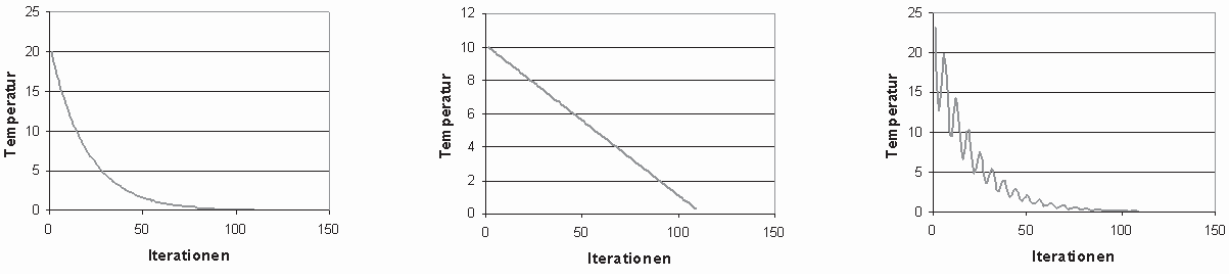


Abb. 23: Temperaturverlauf der drei implementierten Abkühlungsfunktionen für eine Starttemperatur von 10; links exponentiell, Mitte linear, rechts exponentiell fallend und oszillierend.

3.7 Vergleich ausgewählter Metaheuristiken

Zum Vergleich verschiedener metaheuristischen Verfahren werden zwei hypothetische Optimierungsprobleme mit bekannter optimaler Lösung konstruiert, deren Struktur den Problemen entspricht, die im Rahmen des Entscheidungsunterstützungssystems gelöst werden sollen. Die Optimierungsprobleme werden mit ausgewählten metaheuristischen Verfahren gelöst. Auf Basis der erreichten Lösungsqualitäten und des dazu benötigten Rechenaufwandes wird dann eine Entscheidung getroffen, welche oder welches der Verfahren in das Entscheidungsunterstützungssystem integriert werden sollen.

Das erste konstruierte Optimierungsproblem besteht darin, aus einer endlichen Menge von Beständen eine Untermenge zu identifizieren. Die ausgewählten Bestände sollen insgesamt eine definierte Mindestmenge eines Zielsortiments bereitstellen und räumlich möglichst dicht beieinander liegen. Darüber hinaus soll das Mittel der Pflegedringlichkeit der selektierten Bestände möglichst hoch sein. Es liegt dementsprechend ein Auswahlproblem vor. Das Problem umfasst insgesamt 800 Bestände mit einer definierten räumlichen Anordnung zueinander. Da eine Auswahlentscheidung getroffen werden soll, kann das Problem binär kodiert werden. Wird ein Bestand gewählt, erhält der entsprechende Parameter den Wert 1 ansonsten den Wert 0. Theoretisch ergeben sich maximal 2^{800} mögliche Kombinationen. Die Größe des Suchraums wird dadurch eingeschränkt, dass nicht alle Bestände jedes gesuchte Zielsortiment beinhalten. Ist das Zielsortiment in lediglich der Hälfte aller Bestände vorhanden, beträgt die Suchraumgröße 2^{400}.

$$\min \leftarrow f(D,P) \qquad \text{wobei:} \quad \textit{Gl. 4}$$
$$\text{D = kürzester Weg zwischen allen ausgewählten Beständen}$$
$$\text{P = Pflegedringlichkeit}$$

Zur Lösung des Problems wird zunächst eine Zielfunktion aufgestellt (Gl. 4), in welche die beiden skalierten und gewichteten Zielgrößen Gesamtabstand (*D*) und mittlere Pflegedringlichkeit (*P*) eingehen. Es wird unterstellt, dass die Nutzung pflegedringlicher Bestände dem Entscheidungsträger doppelt so wichtig ist, wie der Abstand zwischen den ausgewählten Beständen. Die Pflegedringlichkeit liegt bereits normiert vor (vgl. Kap. 4.5.4). Der Abstand der Bestände zueinander (*D*) wird über den Quotienten aus der Fläche der Bounding Box[5] um alle ausgewählten Bestände und die Fläche der Bounding Box um alle potentiellen Bestände berechnet. Der Wert von *D* liegt somit ebenfalls im Intervall $\{D \in \mathbb{R} \mid -1 < D \leq 1$. Die beiden skalierten und gewichteten Werte *P* und *D* werden additiv verknüpft. Da Bestände mit einer möglichst hohen Pflegedringlichkeit gesucht werden, ist diese mit -1 zu multiplizieren, so dass durch Minimieren der Funktion hohe Pflegedringlichkeiten der ausgewählten Bestände resultieren.

[5] Bounding Box: Fachbegriff aus dem Bereich der Geo-Informationssysteme. Beschreibt ein Rechteck, welches eine oder mehrere Polygone so umschließt, dass diese gerade in der Bounding Box liegen, das Rechteck also so klein wie möglich ist und alle Punkte der Polygone beinhaltet.

Das zweite Optimierungsproblem besteht darin, allen Beständen eine Behandlungsalternative zuzuweisen. Die ausgewählten Optimierungsverfahren sollen die Behandlungsalternativen den einzelnen Beständen so zuweisen, dass eine gesamtbetriebliche Zielfunktion (Gl. 5) maximiert wird. Somit liegt in diesem Fall ein Zuweisungsproblem vor. Jedem der 800 Bestände werden vier verschiedene Behandlungsalternativen (Suchraumgröße = 4^{800}) zugewiesen, welche verschiedene Werte für die Teilnutzen Gewinn (*g*) und Naturschutz (*nat*) unterstellen. Weiterhin wird eine Simulationsdauer von zwanzig Jahren angenommen, so dass für jede Behandlungsvariante aller Bestände eine Nutzungsmatrix vorgegeben wird, die angibt, in welchem Jahr welche Mengen in einem Bestand bei einer der Varianten entnommen wurden.

$$\max \leftarrow w_1 \sum_{i=1}^{n} g_i + w_2 \sum_{i=1}^{n} nat_i + w_3 k \qquad \textit{Gl. 5}$$

wobei:

w_1, w_2, w_3 = Gewichte der Teilnutzen g, n, k ($w_1+w_2+w_3=1$)

i = Bestand i

n = Anzahl vorhandener Bestände

g_i = Teilnutzen *Gewinn* von Bestand i

nat_i = Teilnutzen *Naturschutz* von Bestand i

k = Teilnutzen *konstante Nutzung* über alle Bestände

Die Berechnung des Teilnutzens *konstante Nutzung* erfolgt nach dem in Kap. 4.5.1 beschriebenen Vorgehen. Die einzelnen Teilnutzen werden mit Werten zwischen 0 und 1 versehen, so dass auch die Zielfunktion Werte zwischen 0 und 1 liefert. Die bestandesweisen Teilnutzen und die Nutzungsmengen und -Zeitpunkte wurden so festgelegt, dass bei einer ungefähr gleichverteilten Zuweisung der verschiedenen Handlungsalternativen zu den entsprechenden Beständen ein maximaler Zielfunktionswert von 1 resultiert.

Verglichen werden die Verfahren Simulated Annealing (SA), Tabu Search (TS) und der genetische Algorithmus (GA) sowie die modifizierten Varianten des Simulated Annealing SAM (mit Gedächtnis), SAG (mit Verkürzung der inneren Schleife), SAMG (mit Gedächtnis und Verkürzung der inneren Schleife) und PSA (paralleles Simulated Annealing), sowie die parallelisierte Variante des Genetischen Algorithmus (PGA). Die Kompasssuche ist für die vorgestellten Problemstellungen ungeeignet. Dieses Verfahren besitzt einerseits keine immanenten Mechanismen zum Überwinden lokaler Optima, andererseits kann es nur Zielvariablen aus dem Bereich der reellen Zahlen verarbeiten.

Die Parametrisierung der jeweiligen Verfahren hat einen großen Einfluss auf den Erfolg der Optimumsuche (CHEN 2003, HINRICHS 2006a). Werden mehrere Verfahren verglichen, sollte sichergestellt sein, dass die Parametrisierung für alle Verfahren bestmöglich vorgenommen wird. Andererseits besteht die Gefahr, dass ein Verfahren einem überlegenen Verfahren auf Grund einer schlechten Parametrisierung vorgezogen wird. Die Parametrisierung eines metaheuristischen Verfahrens kann ebenfalls als ein Optimierungsproblem interpretiert werden. Die Parameter zur Steuerung des Verfahrens sind so zu wählen, dass eine gute Lösungsqualität erreicht wird und die dazu benötigte Rechenzeit oder die Anzahl der Aufrufe der Kostenfunktion möglichst minimiert wird. Dabei muss jedoch bedacht werden, dass die Allgemeingültigkeit und Übertragbarkeit des Verfahrens auf (geringfügig) anders strukturierte Probleme durch eine zu spezielle Parametrisierung verloren gehen kann.

Ein solcher Fall ist in Abb. 24 beispielhaft anhand von drei Optimumsuchen mit dem Verfahren Simulated Annealing dargestellt. Für alle drei Suchläufe wird die gleiche Parametrisierung verwendet und es liegt das gleiche Optimierungsproblem (Bestandesauswahl) zugrunde. Das globale Optimum besitzt in allen drei Fällen einen Funktionswert von 1. Die einzelnen Läufe unterscheiden sich nur darin, dass die Suchraumgröße variiert. Im ersten Lauf (a) beträgt die Anzahl der potentiellen Bestände 50. In Lauf b können 200 Bestände das Zielsortiment bereitstellen. Im dritten Lauf (c) sind es 800 potentielle Bestände. Die verwendete Parametrisierung wurde für den Lauf b optimiert, ohne einen möglichen Verlust der Allgemeingültigkeit zu beachten. Der Lösungsprozess des Problems b weist erwartungsgemäß einen optimalen Verlauf auf. Nach einer kurzen explorativen Phase (bis ca. 450 Aufrufe der Kostenfunktion) ist eine stetige Verbesserung des Funktionswertes zu beobachten. Die Spanne, die keine weiteren Verbesserungen bringt, ist relativ kurz, da die Abbruchfunktion optimal auf das Problem angepasst ist. Die Auswirkungen der Übertragung dieser stark problemspezifischen Parametrisierung auf geringfügig anders strukturierte Probleme ist in den Verläufen a und c deutlich zu erkennen. Der Verlauf der Optimumsuche zu Problem a zeigt, dass das globale Optimum mit der gegebenen Parametrisierung nicht gefunden werden kann. Die Suche wird nach ca. 500 Aufrufen der Zielfunktion abgebrochen und liefert keine befriedigende Lösung. Es wird lediglich ein Zielfunktionswert von etwas mehr als 0,3 erreicht. In Fall c wird hingegen eine bessere Lösungsqualität erreicht (ca. 0,82). Das globale Optimum wird aber ebenfalls nicht gefunden. Das Verfahren bleibt in einem lokalen Optimum hängen und es werden ca. 1000

unnötige Aufrufe der Kostenfunktion durchgeführt, die zu keiner Verbesserung der Lösungsgüte führen.

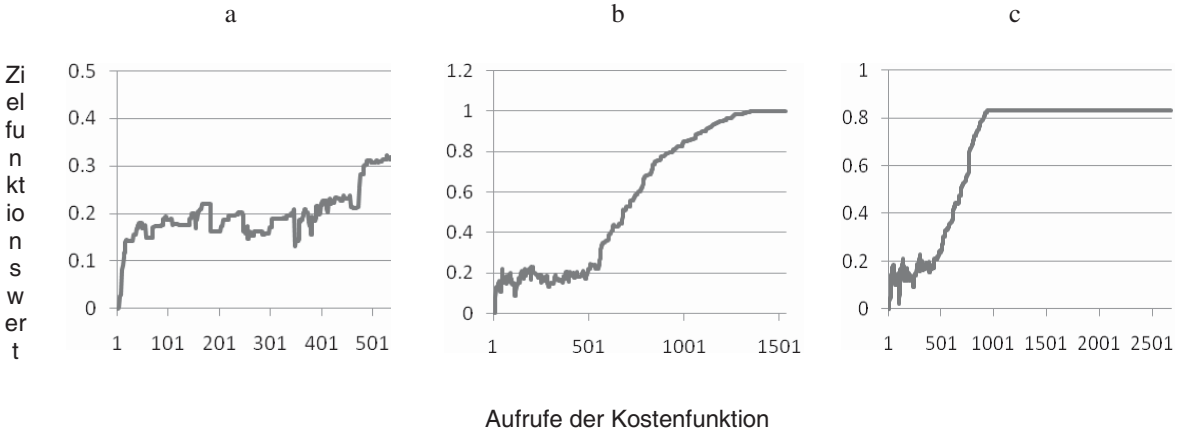


Abb. 24: Auswirkungen einer zu problemspezifischen Parametrisierung.

Da die Strukturen der Optimierungsprobleme, die das im Rahmen dieser Arbeit entwickelte Entscheidungsunterstützungssystem lösen soll, bekannt sind und hauptsächlich die Suchraumgröße variiert, sind vor allem die Parameter sorgfältig zu wählen, welche das Abbruchverhalten beeinflussen. Die Abbruchparameter müssen so eingesteuert werden, dass bei steigender Suchraumgröße die Anzahl der einzelnen Suchschleifen automatisch angepasst wird. Metaheuristische Verfahren unterliegen grundsätzlich dem No-Free-Lunch-Theorem (s. o.). Sie können zwar durch geschickte Modifikation und Parametrisierung effizienter gestaltet werden, jedoch ist bei zunehmender Suchraumgröße auch ein größerer Suchaufwand (Anzahl Aufrufe der Zielfunktion) nötig, um vergleichsweise gute Lösungen zu ermitteln. Abb. 25 zeigt diesen Zusammenhang am Beispiel von Simulated Annealing. Das Verfahren wurde für die Lösung des Bestandesauswahl-Problems mit zunehmender Anzahl potentieller Bestände (solcher Bestände, die das Zielsortiment beinhalten) verwendet und dann abgebrochen, wenn ein Lösungsniveau von 0,95 erreicht wurde. Für die Spannweite der betrachteten Suchraumgrößen ist ein linearer Zusammenhang zwischen der benötigten Anzahl der Aufrufe der Zielfunktion und der Anzahl potentieller Bestände zu erkennen. Um bei wechselnder Suchraumgröße nicht zu viele oder zu wenige Iterationen zu durchlaufen, müssen die Suchverfahren mit einem flexiblen Abbruchsystem ausgestattet werden.

Alle zu vergleichenden Verfahren werden mit einem Abbruchmechanismus ausgestattet, der eine zweistufige Prüfung vornimmt. Da die Zielfunktionen so konstruiert wurde, dass die

Zielfunktionswerte nur Werte zwischen 0 und 1 bzw. -1 annehmen können, kann in der ersten Stufe ein einfaches aber sehr effektives Abbruchkriterium geprüft werden.

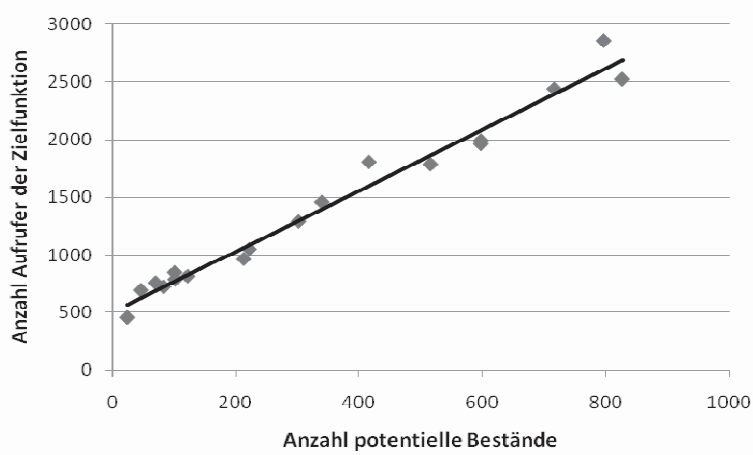


Abb. 25: Benötigte Aufrufe der Kostenfunktion, um ein einheitliches Lösungsniveau von 0.95 zu erreichen in Abhängigkeit von der Suchraumgröße (Anzahl potentielle Bestände).

Es wird ein zu erreichendes Mindestlösungsniveau festgelegt. Ist dieses Niveau erreicht, wird geprüft, ob es sich lohnt, noch weiter zu suchen oder eine weitere Verbesserung nur mit einem hohen Suchaufwand zu erreichen ist. Bei dieser Abschätzung sind Verfahren im Vorteil, die den Verlauf der Suche in einem „Gedächtnis“ speichern. Je weniger Informationen zum Lösungsverlauf zur Verfügung stehen, umso unsicherer kann abgeschätzt werden, ob noch weitere Verbesserungen möglich sind. In diesem Fall erfolgt der Abbruch nach einer definierten Anzahl weiterer Iterationen nach Erreichen des vorgegebenen Lösungsniveaus. Bei alleinigem Anwenden dieses Abbruchkriteriums besteht die Gefahr, dass das gewünschte Lösungsniveau nicht erreicht werden kann, und die Suche endlos fortgesetzt wird. Um dies zu vermeiden, wird in der zweiten Stufe geprüft, wie groß der Unterschied der aktuellen Lösung zu einer definierten Anzahl vorhergehender Lösungen ist. Wird keine ausreichende Verbesserung erzielt, wird der Suchvorgang abgebrochen. Dieses Abbruchkriterium sollte jedoch nur in Kombination mit Verfahren eingesetzt werden, welche eine effektive Strategie zur Überwindung lokaler Optima besitzen, sonst besteht die Gefahr, dass das Verfahren zu früh (in einem lokalen Optimum) abbricht.

Jeder der in diesem Kapitel betrachteten Algorithmen wird 10.000 mal auf ein ähnliches Optimierungsproblem angewendet, wobei lediglich die gesuchten Sortimente (Auswahlproblem) bzw. die Anzahl der möglichen Behandlungsvarianten (Zuweisungsproblem) variiert werden. In dem konstruierten Auswahlproblem sind 4 Baumarten zu gleichen Vorratsanteilen vertreten. Die Zielsortimente werden so definiert, dass

alle 2500 Läufe eine weitere Baumart mit in die Zielsortimentsdefinition aufgenommen wird. So werden mit jedem untersuchten Verfahren jeweils 2500 Suchläufe mit einer Suchraumgröße von 2^{200}, 2^{400}, 2^{600} und 2^{800} durchgeführt. Liefern die Verfahren trotz der variierenden Suchraumgröße qualitativ gute Lösungen, kann davon ausgegangen werden, dass das Verfahren robust parametrisiert wurde. Für die Prüfung auf Eignung für das kombinatorische Problem wird die Anzahl der Bestände, welche in die Optimumsuche mit einbezogen werden sollen, ebenfalls schrittweise erhöht. Die Suchraumgröße beträgt jeweils 4^{200}, 4^{400}, 4^{600} und 4^{800}.

3.7.1 Ergebnisse

Zum Vergleich der Lösungsgeschwindigkeit werden die mittlere Rechenzeit und die mittlere Anzahl benötigter Aufrufe der Zielfunktion herangezogen. Die Lösungsqualität wird über den mittleren Zielfunktionswert, dem Anteil der Suchläufe, die das globalen Optimum finden, sowie über die Standartabweichung aller (10.000) ermittelten Minima bzw. Maxima beurteilt.

In Tab. 4 und Tab. 5 sind die Lösungsqualität und die zur Lösung benötigte Rechenzeit bzw. Anzahl der Aufrufe der Zielfunktion für alle verglichenen Algorithmen dargestellt. Insgesamt ist zu erkennen, dass die Suchverfahren für das Zuweisungsproblem bessere Lösungsqualitäten liefern. Vor allem der Genetische Algorithmus (GA, PGA) schneidet deutlich besser ab. Eine Ausnahme stellt das Verfahren SAG dar. Dieses liefert geringfügig schlechte Lösungsqualitäten. Die zur Lösung benötigte Anzahl der Aufrufe der Kostenfunktion und somit auch der absoluten Rechenzeit sind aufgrund des größeren Lösungsraums deutlich höher.

Tab. 4: Ergebnis des Vergleichs ausgewählter Algorithmen auf Basis des Auswahlproblems; (+)= bester Wert, (-) = schlechtester Wert.

	SA	**SAM**	**SAG**	**SAMG**	**PSA**	**TS**	**GA**	**PGA**
Mittleres Optimum	-0,966	-0,996	**-0,955**$^{(-)}$	-0,996	**-0,999**$^{(+)}$	-0,976	-0,978	-0,978
Standardabweichung Optimum	0,085	0,038	**0,091**$^{(-)}$	0,007	**0,002**$^{(+)}$	0,042	0,040	0,040
globales Minimum gefunden [%]	**40**$^{(-)}$	93	41	95	**97**$^{(+)}$	50	82	81
Mittlere Anzahl Aufrufe C	1613,2	1551,3	1364,1	1306,4	**1181,8**$^{(+)}$	1566,2	**1738,1**$^{(-)}$	1734,7
Mittlere Zeit [ms]	244,9	**1204,9**$^{(-)}$	210,8	1075,2	180,04	117,38	94,91	**42,51**$^{(+)}$

Tab. 5: Ergebnis des Vergleichs ausgewählter Algorithmen auf Basis des Zuweisungsproblems; (+)= bester Wert, (-) = schlechtester Wert.

	SA	SAM	SAG	SAMG	PSA	TS	GA	PGA
Mittleres Optimum	0,970	0,997	**0,948**$^{(-)}$	0,998	**0,999**$^{(+)}$	0,981	0,989	0,990
Standardabweichung Optimum	0,081	0,012	**0,094**$^{(-)}$	0,009	**0,002**$^{(+)}$	0,035	0,004	0,004
globales Maximum gefunden [%]	46	95	**40**$^{(-)}$	96	**98**$^{(+)}$	63	96	97
Mittlere Anzahl Aufrufe C	**12103**$^{(-)}$	9639	10241	9807	**8865**$^{(+)}$	11750	11038	11018
Mittlere Zeit [ms]	1846	9039	1590	**9064**$^{(-)}$	1357	1090	842	**447**$^{(+)}$

Bis auf die mittlere Rechenzeit weist PSA die besten Resultate in beiden Anwendungsbereichen (Auswahlproblem, Zuweisungsproblem) auf. In 97 bzw. 98 Prozent der durchgeführten Suchläufe wurde das globale Optimum mit den geringsten Aufrufen der Kostenfunktion gefunden. Die im Mittel erreichten Optima von -0,999 und 0,999 deuten ebenfalls auf ein sehr hohes Lösungsniveau hin. Die bekannten globalen Optima liegen bei 1 bzw. -1. Die gute Lösungsqualität des Verfahrens PSA ist auf den wesentlichen Vorteil der Parallelisierungsstruktur zurückzuführen. Neben der höheren Lösungsgeschwindigkeit (180 bzw. 1357 ms), bietet die Parallelisierung die Möglichkeit mehrere Instanzen der Variationsfunktion zum Erzeugen neuer aktueller Lösungen zu nutzen. Diese können so gestaltet werden, dass alle Instanzen die gleiche Nachbarschaftsbeziehung verwenden, oder es werden verschiedene Nachbarschaftsbeziehungen verwendet. Im vorliegenden Vergleich wurde das Verfahren mit zwei verschiedenen Nachbarschaftsfunktionen ausgestattet, wobei eine die Intensivierung der Suche begünstigt und die andere eine höhere Diversifikation bewirkt. Auf diese Weise wird der Suchprozess so gesteuert, dass ein hoher Diversifikationsanteil den gesamten Suchprozess über beibehalten wird. Es werden lokale Optima besser überwunden und der Suchraum insgesamt und nicht nur in der Nähe guter Lösungen gründlicher durchsucht. Dies führt zu einer hohen Robustheit und einer guten Lösungsqualität des Verfahrens. An dieser Stelle ist ein fließender Übergang zu naturinspirierten Verfahren wie Ameisen-Algorithmen oder Multiagenten-Systemen gegeben. Das Basisverfahren SA weist deutlich schlechtere Ergebnisse auf. In lediglich 40 bis 46 Prozent der durchgeführten Suchläufe wird das globale Optimum gefunden, obwohl die hohe Anzahl der Aufrufe der Kostenfunktion auf einen hohen Suchaufwand hindeutet. Dieses Verfahren arbeitet mit einer exponentiellen Abkühlungsfunktion und nur einer

Variationsfunktion, so dass die Suche häufig in einem späten Stadium (bei geringen Temperaturen) in einem lokalen Optimum hängen bleibt, da notwendige kurzzeitige Verschlechterungen, um das lokale Optimum überwinden zu können, mit nur noch sehr geringer Wahrscheinlichkeit akzeptiert werden. Die erweiterte Variante SAG benötigt zwar weniger Aufrufe der Kostenfunktion, bewirkt aber keine Verbesserung der Lösungsqualität. Die erreichten Qualitäten liegen auf dem Niveau des einfachen Simulated Annealing. Das Verwenden eines Gedächtnisses ist eine effiziente Möglichkeit, unabhängig vom Suchfortschritt lokale Optima zu überwinden. Dieser Mechanismus wird von den Verfahren SAM, SAMG und TS verwendet. Durch die Kombination der Metropolis-Wahrscheinlichkeit mit einem Gedächtnisspeicher wird eine effiziente Steuerung der Diversifikation und Intensivierung des Suchprozesses bei SAM und SAMG bewirkt. In 93 bis 96 Prozent der Optimierungsläufe finden diese Verfahren das globale Optimum. Dementsprechend gering ist auch die Standardabweichung (0,038 - 0,009). Die beiden Erweiterungen des Simulated Annealing mit einem Gedächtnisspeicher SAM und SAMG weisen bei beiden Optimierungsproblemen jedoch die höchsten Rechenzeiten auf. Dies ist dadurch zu erklären, dass für das Abgleichen einer potentiellen, aktuellen Lösung mit allen im Gedächtnis gespeicherten Lösungen entsprechend viel Rechenzeit benötigt wird. Dieser Vorgang kann zwar durch Verwendung schneller Speicherstrukturen mit konstanter Zugriffszeit (Hashtable, Hashmap) effizient gestaltet werden, die benötigte zusätzliche Rechenzeit führt dennoch dazu, dass die Verfahren ohne Gedächtnisspeicher schneller durchlaufen. Tabu Search (TS) arbeitet im Vergleich zu SAM und SAMG nur mit einem Gedächtnisspeicher. Dadurch ist die Lösungsgeschwindigkeit deutlich schneller (117 und 11750 ms). Die dabei erreichten Lösungsqualitäten liegen vergleichsweise im mittleren Bereich. Das Verfahren findet in 50 bzw. 63 Prozent der Suchläufe das globale Optimum.

Bemerkenswert ist, dass der Genetische Algorithmus für das Auswahlproblem deutlich schlechtere Ergebnisse liefert (GA 82 Prozent, PGA 81 Prozent) als für das Zuweisungsproblem. Dies kann damit erklärt werden, dass die verwendeten Kreuzungsmechanismen die Auswahl weniger Bestände aus einer großen Anzahl nicht effizient leisten können, da durch das Kreuzen zweier guter Individuen leicht zwei sehr schlechte Individuen entstehen können. In Abb. 26 werden zwei relativ gute Individuen (Auswahlsituationen) gekreuzt.

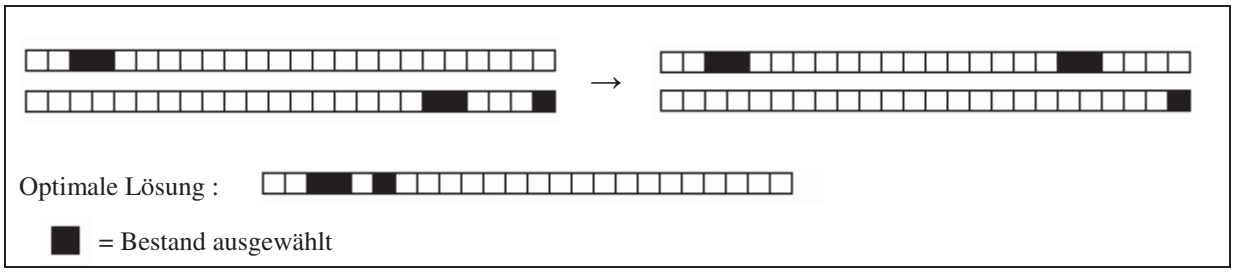


Abb. 26: Kreuzung zweier guter Individuen mit dem Ergebnis zweier schlechter Individuen.

Die neuen Individuen (rechts) weisen eine schlechte Fitness (Zielfunktionswert) auf. Die obere neue Auswahlsituation beinhaltet fast alle in den beiden Ausgangsindividuen ausgewählten Bestände, so dass sich z. B. die Aggregation deutlich verschlechtert. Das zweite neue Individuum beinhaltet lediglich einen ausgewählten Bestand, so dass diese Lösung nicht gültig ist oder als sehr schlecht eingestuft wird, da die geforderte Holzmenge nicht erreicht wird.

Bei Anwendung auf das Zuweisungsproblem liefern die Verfahren GA und PGA deutlich bessere Ergebnisse. Hier erreichen die Verfahren ähnliche Lösungsqualitäten wie sie durch die modifizierten Varianten des Simulated Annealing SAMG und PSA erzielt werden. Hinsichtlich der Rechenzeit sind in dem durchgeführten Vergleich der Genetische Algorithmus und dessen parallelisiertes Derivat die schnellsten Verfahren. Die parallelisierten Varianten PSA und PGA arbeiten mit einer Aufteilung der Suche auf zwei Threads und benötigen erwartungsgemäß sehr wenig Rechenzeit. Die ohne Parallelisierungsmechanismen implementierten Verfahren GA und auch TS arbeiten ebenfalls sehr schnell. Dies ist vor allem auf die einfache Struktur dieser Verfahren zurückzuführen.

3.8 Dijkstra-Algorithmus

Nach der Vorstellung der für eine Vielzahl unterschiedlicher Optimierungsprobleme einsetzbaren metaheuristischen Verfahren, wird in diesem Kapitel ein Algorithmus behandelt, welcher speziell für den Bereich sog. Routing-Probleme entwickelt wurde. Im Rahmen der Suche nach einer optimalen Bestandesauswahl zur Bereitstellung definierter Zielsortimente soll u. a. der Umsetzaufwand möglichst gering gehalten werden. D. h. die Wegstrecken, die zurückzulegen sind, um von einem ausgewählten Bestand zum nächsten zu gelangen, sollen minimiert werden. Liegen keine Geoinformationen zum Wegenetz vor, werden die

euklidischen Distanzen verwendet. Steht ein Shapefile[6] zur Erschließung zur Verfügung, wird der Dijkstra-Algorithmus verwendet, um zwischen zwei ausgewählten Beständen die kürzeste Route im gegebenen Wegenetz zu berechnen.

Gerade bei Einbeziehung räumlicher Komponenten in den Optimierungsprozess spielt die Wegeoptimierung eine große Rolle. Wobei, je nach Problemstellung, die Zielfunktionen sehr unterschiedlich ausfallen können. So kann z. B. einmal gefordert sein, den kürzesten Weg zwischen mehreren Zielorten zu finden (Traveling Salesman Problem) oder die kürzeste Route zwischen einem Start- und einem Zielpunkt zu ermitteln. Im vorliegenden Fall ist es notwendig zwei Optimierungsverfahren zu kombinieren. Ein Verfahren aus dem Bereich der Lokalen Suche kommt zum Einsatz, um die optimale Reihenfolge festzulegen, in welcher die ausgewählten Bestände abzuarbeiten sind, um die Gesamtroute zu minimieren. Der Dijkstra-Algorithmus wird benötigt, um die kürzeste Route zwischen zwei Beständen zu ermitteln. Er wird also in die Zielfunktion eines metaheuristischen Verfahrens integriert. Neben dem Dijkstra-Algorithmus gibt es noch eine Reihe anderer Verfahren zur Lösung dieses Problems (Bellman-Ford-Moore, Floyd-Warschall (BANG-JENSEN u. GUTIN 2007), A*-Algorithmus (HART et al. 1968)).

Um die Rechenzeit zu minimieren, ist die Verwendung eines speziellen Algorithmus zur Lösung von Routing-Problemen sinnvoll. Würde man bei der Suche nach dem kürzesten Weg zwischen zwei Punkten alle möglichen Pfade berechnen und anschließend den kürzesten auswählen, ergibt sich das Problem, dass die Anzahl der zu berechnenden Pfade exponentiell mit der Anzahl an Knoten oder Kreuzungen im gegebenen Wegesystem wächst.

Algorithmen, die geeignet sind, das Problem zu lösen, den kürzesten Pfad zwischen allen Punkten oder einer Untermenge aller Punkte zu berechnen, wurden in Kap. 3 vorgestellt. In diesem Kapitel sind vor allem solche Algorithmen von Interesse, die den kürzesten Weg von einem Startpunkt zu einem oder allen anderen Punkten im Wegesystem berechnen. In dieser Arbeit wird der Algorithmus von Dijkstra (DIJKSTRA 1959) in seiner ursprünglichen Form erläutert, eine modifizierte Variante abgeleitet und beide hinsichtlich der Laufzeit zur Routenoptimierung verglichen.

[6] Unter einem Shapefile versteht man ein Dateiformat zum Speichern von Geodaten. Das Format wurde von ESRI ursprünglich für ArcView entwickelt und umfasst mindestens drei Dateien (Sachdaten (dbf), Geometrien (shp), Index zur Verknüpfung von Sachdaten und Geometrien (shx)).

Der Dijkstra-Algorithmus ist den Greedy-Strategien zuzuordnen. Diese Verfahren beruhen auf der Annahme, dass durch Auswahl der besten Lösung in einem Iterationsschritt und darauf aufbauende Verbesserung der Lösung die optimale Lösung gefunden wird. Hierzu wird in jedem Iterationsschritt jede mögliche Lösung geprüft (→ greedy/gierig, s. o.).

3.8.1 Graphentheorie

Die gängige Darstellung von Routing-Problemen und notwendiger Datenstrukturen erfolgt mittels der Graphentheorie. Unter einem Graphen im Sinne der Graphentheorie versteht man ein Konstrukt aus Knoten oder auch Ecken oder Punkten, welche über Kanten miteinander verbunden sind (JUNGNICKEL 1987, BANG-JENSEN u. GUTIN 2007, DIESTEL 2011). Diese Relation von Objekten kann beispielsweise ein Stammbaum oder eine Straßenkarte sein. Beim Stammbaum sind die Familienmitglieder die Objekte oder Knoten und die Kanten stellen die direkten verwandtschaftlichen Beziehung der Personen dar. Bei der Straßenkarte sind Ortschaften und andere eingezeichnete Orte sowie Kreuzungen die Knoten und die Straßen die Kanten.

Ein Graph G ist somit ein Paar zweier endlicher Mengen V und E.

$$G = (V, E)$$

V (engl.: vertices) bezeichnet die Menge der im Graphen enthaltenen Knoten und E (engl.: edges) die Menge der Verbindungen zwischen den Kanten. Ein sog. Nullgraph ist auszuschließen, es gilt:

$$V \cap E \neq \emptyset$$

$$V \neq \emptyset$$

Weiterhin besteht eine injektive Abbildung auf die Kantenmenge E eines Graphen:

$$\Psi : E \mapsto V^2$$

Jede Kante eines ungerichteten Graphen entspricht somit genau einem ungeordneten Eckenpaar, so dass für jede Kante e aus E mit den Knoten v und w gilt:

$$\Psi(e) = \{v, w\} = \{w, v\}$$

In einem Digraphen können sowohl gerichtete als auch ungerichtete Kanten auftreten (BANG-JENSEN u. GUTIN 2007). Es ist möglich, das zwei adjazente[7] (benachbarte) Knoten v und w in beide Richtungen (w ↔ v), im gerichteten Fall aber z. B. nur eine Verbindung von Knoten v zu w (v → w) aber nicht von w zu v möglich ist. In solchen Fällen werden die Kanten eines Graphen oft durch Vektoren dargestellt.

Werden die Kanten E eines Graphen G mit Kosten oder einer Kostenfunktion c versehen, liegt ein gewichteter Graph vor.

$$c : E \rightarrow \mathbb{R}$$

$$G = (V, E, c)$$

Ein Weg oder eine Kantenfolge W in einem Graphen ist eine Teilmenge von V wobei v_x und v_{x+1} durch eine Kante verbunden sein müssen.

$$W = \{v_1, ..., v_n\}$$

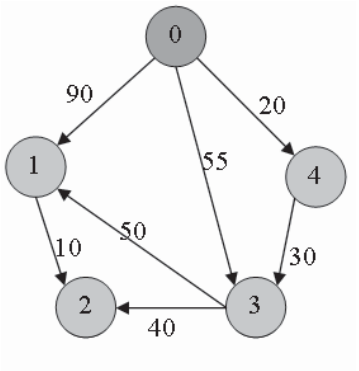
Eine geeignete Form der Darstellung und Datenstruktur für einen gerichteten und gewichteten Graphen ist eine Adjazenzmatrix (BANG-JENSEN u. GUTIN 2007). Diese Datenstruktur gibt in einem Zeilen-Spaltensystem an, ob für die Knoten v_i und v_j überhaupt eine Kante besteht. Liegt keine Verbindung zwischen einem Knotenpaar vor, wird dies durch ein definiertes Element gekennzeichnet. Die Größe der Matrix resultiert aus der Anzahl der Knoten. Hat ein Graph 4 Knoten, ist eine 4x4 Adjazenzmatrix aufzustellen. Diese Datenstruktur hat den Nachteil, dass auch bei Graphen mit wenig Kanten und vielen Knoten die Größe der Matrix (=O(n²)) nicht verringert werden kann und viele Einträge gespeichert werden, die das Nichtbestehen einer Verbindung zwischen zwei Knoten anzeigen. Vorteil dieser Darstellung ist unter anderem, dass die Existenz einer Kante mit einer Laufzeit von O(1) festgestellt werden kann.

3.8.2 Der Algorithmus

Der Dijkstra-Algorithmus (DIJKSTRA 1959) berechnet den kürzesten Weg (den Weg mit den geringsten Kosten) zwischen einem Startknoten und allen anderen, erreichbaren Knoten in einem attributierten, gerichteten Graph G = (V, E). Ein attributierter Graph beinhaltet neben

[7] Einige Autoren definieren, dass v nur dann adjazent zu w ist, wenn eine gerichtete Kante von v zu w existiert, also $e_{v,w}$ = (v, w). Hier wird durch die Adjazenz zweier Knoten lediglich vorausgesetzt, dass eine Kante $e_{v,w}$ = (v, w) oder $e_{w,v}$ = (w, v) existiert. Die Richtung ist für die Adjazenz nicht relevant.

allen Knoten z. B. in einem Wegenetz auch die Kosten zum Erreichen direkter Nachbarknoten (vgl. Abb. 27, links). Es gibt verschiedener Datenstrukturen, um einen gerichteten oder ungerichteten, attributierten Graphen effizient abzubilden. Oft bietet es sich an, den Graphen in Form einer Kostenmatrix (*K*) mit n Zeilen (i) und Spalten (j) zu konstruieren (Abb. 27, rechts). Wobei n der Anzahl der Knoten entspricht. Im Vergleich zu einer Adjazenzmatrix (s. o.) beinhaltet eine Kostenmatrix anstelle eines Elements, welches das Bestehen einer Verbindung anzeigt, die Kosten der Verbindung (z. B. Entfernung in Metern). Die Vor- und Nachteile entsprechen denen der Adjazenzmatrix.



$$\begin{bmatrix} 0 & 90 & \infty & 55 & 20 \\ \infty & 0 & 10 & \infty & \infty \\ \infty & \infty & 0 & \infty & \infty \\ \infty & 50 & 40 & 0 & \infty \\ \infty & \infty & \infty & 30 & 0 \end{bmatrix}$$

Abb. 27: Graph und Matrixdarstellung

Wie in Abb. 27 zu erkennen, wird an den Stellen ij, an denen es keine direkte oder nur eine gerichtete Verbindung von j zu i aber nicht von i zu j gibt, das Symbol ∞ (= unendlich) eingetragen. Damit wird verhindert, dass der gesuchte, kürzeste Weg über diese (nicht existenten) Kante führt.

Die Funktionsweise des Dijkstra-Algorithmus (Abb. 28) kann folgendermaßen beschrieben werden: In zwei Vektoren werden für jeden Knoten *k* die kürzeste Distanz bzw. die geringsten Kosten (Vektor *d*) zum Startknoten und der direkte Vorgängerknoten von *k* (Vektor *p*) im kürzesten Pfad gespeichert. In einem weiteren Vektor *v* werden alle durchlaufenen Knoten gesammelt. Im Rahmen der Initialisierung wird zunächst nur der Startknoten in *v* gespeichert. Der Vektor *d* wird mit der ersten Zeile der Kostenmatrix gefüllt. In *p* wird für jeden Knoten $K_{0,0}$ (der Startknoten) eingetragen. Der Algorithmus iteriert solange, bis alle Knoten in *v* enthalten sind. In jeder Iteration wird der Knoten (k_{min}) bestimmt, der die geringste Distanz zum Startknoten hat und kein Element von *v* ist. Dann wird für alle verbleibenden Knoten geprüft, ob über k_{min} ein kürzerer Pfad (als bislang in *d* gespeichert) möglich ist. Ist dies der Fall, wird für den jeweiligen Knoten in *p* k_{min} und in *d* die kürzere

Distanz eingetragen. Abschließend wird k_{min} in v aufgenommen. Nach Durchlauf des Algorithmus liegt die Lösung mit den beiden Vektoren d und p vor. Der Vektor d enthält die minimalen Distanzen der einzelnen Knoten zum Startknoten. Mit dem Vektor p liegen die Pfade zu allen Knoten implizit vor, d. h. sie können mit Hilfe der in p gespeicherten Vorgängerknoten rekursiv ermittelt werden.

```
1  function shortestPath(X)
2    while one or more nodes are not used do
4     k_min ← the unused node with the shortest d[k_min]
5     set k_min used
6     for all distances from k_min to a neighbour node m with the length L do
7           if d[k_min]+L < d[m]    then
8                 d[m] ← d[k_min]+L  {a shorter way to m found, includes k_min}
9                 p[m] ← k_min
10    end for
11   end while
12 end
```

Abb. 28: Dijkstra- Algorithmus.

Für die in Abb. 27 dargestellte Matrix ergeben sich die in Tab. 6 veranschaulichten Iterationsschritte mit den Vektoren v, p und d. Als Startknoten wurde Knoten 0 gewählt. Nach der ersten Iteration sind die kürzesten Distanzen zu 1, 3 und 4 die Distanzen zum Startknoten. Der Vektor p beinhaltet für 1, 3 und 4 jeweils eine Null. Als k_{min} wird Knoten 4 ermittelt. Dieser hat nur Knoten 3 als direkten Nachbarknoten. Die Distanz von 0 zu 3 über 4 ist 20+30=50. Der aktuelle Wert von d[3] ist 55. Folglich wird d[3] von 55 auf 50 geändert.

Tab. 6: Iterationsschritte und die resultierenden Vektoren p und d.

Iteration	v	d[0],p[0]	d[1],p[1]	d[2], p[2]	d[3], p[3]	d[4], p[4]
1	[0]	0, -	90, 0	∞, 0	55, 0	**20**, 0
2	[0,4]	0, -	90, 0	∞, 0	**50**, 4	20, 0
3	[0,4,3]	0, -	90, 0	**90**, 3	50, 4	20, 0
4	[0,4,3,2]	0, -	**90,**0	90, 3	50, 4	20, 0
Abbruch	[0,4,3,2,1]	0, -				

Der nächste noch nicht in v gespeicherte Knoten mit dem kleinsten Wert in d ist Knoten 3 mit dem einzigen direkten Nachbarknoten 2. Der kürzeste Weg von 0 zu 2 ist bis zu diesem Iterationsschritt unendlich (keine direkte Verbindung). Über den Knoten 3 ist nach dieser Iteration der kürzeste Weg 50+40=90. Das Ergebnis liegt nach vier Iterationen vor. Der kürzeste Distanz zu Knoten 3 beträgt 50. Durch Rekursion der Vorgängerknoten kann der

kürzeste Pfad ermittelt werden. Der in *p*[3] gespeicherte Vorgängerknoten von 3 ist 4, der Vorgängerknoten für 4 ist 0. Somit führt der kürzeste Weg von 0 zu 3 über den Knoten 3.

3.8.3 Modifikation des Dijkstra-Algorithmus

Es ist eine Eigenart des Algorithmus von Dijkstra, dass alle minimalen Pfade zu allen erreichbaren Koten berechnet werden. Wenn alle Pfade oder eine große Untermenge davon von Interesse sind, besteht der Vorteil, dass nur ein Durchlauf des Algorithmus durchgeführt werden muss. Eine weitere Eigenschaft des Verfahrens ist, dass, wenn ein Knoten als besucht markiert wurde (vgl. Tab. 6), es keinen kürzeren Weg mehr zu diesem Knoten geben kann. Diese Eigenschaft kann bezüglich der Pfadsuche zum nächstgelegenen Wegesegment für einer Modifikation genutzt werden, um die Performance erheblich zu steigern. In einem zusätzlichen Vektor *w* werden die IDs der Wegeknoten von Interesse abgespeichert und dem Algorithmus übergeben. Wird ein Knoten als bearbeitet markiert und ist er gleichzeitig ein Element von *w*, wird er aus *w* entfernt. Die notwendigen Modifikationen sind in Form von Pseudocode dargestellt (Abb. 29). In Zeile 11 wird überprüft, ob der letzte als genutzt markierte Knoten (k_{min}) in *w* gespeichert ist. Ist dies der Fall, wird er aus *w* entfernt. Zeile 12 prüft ein zusätzliche Abbruchkriterium. Der Algorithmus stoppt, wenn *w* keine Knoten mehr beinhaltet.

```
1 function shortestPath(X)
2  while one or more nodes are not used do
4     k_min ← the unused node with the shortest d[k_min]
5     set k_min used
6     for all distances from k_min to a neighbour node m with the length L do
7           if d[k_min]+L < d[m] then
8                 d[m] ← d[k_min]+L {a shorter way to m found, includes k_min}
9                 p[m] ← k_min
10    end for
11    if k_min element of w then remove k_min from w
12    if w is empty break {stop the algorithm}
13  end while
12 end
```

Abb. 29: Modifizierter Dijkstra-Algorithmus

Mit dieser Modifikation kann ohne Qualitätsverlust des Ergebnisses die Rechengeschwindigkeit gesteigert werden. In Tab. 7 sind die benötigten Rechenzeiten für 3 Optimierungsläufe unter Verwendung des ursprünglichen Dijkstra-Algorithmus und der modifizierten Variante dargestellt. Die Segmentanzahl (Knotenzahl) beträgt im ersten Durchlauf 100 und wird auf 1000 bzw. 10000 gesteigert. Zur Ermittlung der mittleren

Rechenzeit werden jeweils 1000 Optimierungsläufe mit einem zufällig ausgewählten Zielknoten durchgeführt.

Tab. 7: Vergleich durchschnittliche Rechenzeit in Millisekunden für Dijkstra und Dijkstra modifiziert. Das Mittel wird jeweils über 1000 Optimierungsläufe gebildet.

Variante	100	1000	10000
Dijkstra	0,31 ms	31,71 ms	708,97 ms
DijkstraMod	0,15 ms	18,01 ms	450,58 ms

Liegt ein Zielknoten dicht (im Graphen) beim Startknoten, ist ein deutlicher Geschwindigkeitsvorteil bei der modifizierten Variante zu erwarten, da der Algorithmus sehr früh abbrechen kann. Liegt ein Startknoten relativ weit vom Zielknoten entfernt, ist fast der gesamte Graph zu durchsuchen, und es sind kaum Unterschiede hinsichtlich der Lösungsgeschwindigkeit festzustellen. Im Mittel benötigt dementsprechend die modifizierte Variante im Vergleich zur Basisvariante für kleine Graphen ungefähr die Hälfte der Rechenzeit. Bei Verwendung größerer Graphen (10000 Knoten) ist die Berechnung noch ca. 1,6mal schneller.

3.8.4 Implementierung

Zur Implementierung des einfachen und modifizierten Dijkstra-Algorithmus werden insgesamt sechs Klassen bzw. Interfaces erstellt und in einer Bibliothek mit zwei Packages (Algorithms, Common) zusammengefasst (Abb. 30).

Dijkstra (B)		
	Algorithms (P)	
		Dijkstra (C ← *OptimizationAlgorithm*) *DijkstraMod* (C ← *OptimizationAlgorithm*)
	Common (P)	
		OptimizationAlgorithm (I) Graph (C) Node (C) Point(C)

B = Bibliothek, P = Package, C = Class, I= Interface, ← = implementiert nachstehendes Interface

Abb. 30: Aufbau der Klassenbibliothek zum Dijkstra-Verfahren

Die beiden Klassen *Dijkstra* und *DijkstraMod* beinhalten die eigentliche Optimierungssystematik. Die Optimierungsmethode (`void shortestPath(Graph g)`) erwartet in beiden Klassen den zu durchsuchenden Graphen, wobei der Startknoten an erster Stelle stehen muss. Die modifizierte Variante erwartet noch zusätzlich ein oder mehrere spezifizierte Zielknoten. Beide Klassen besitzen die Methoden

```
int[] getBestNodes(int target)
```

und

```
double getMinDist(int target).
```

Diese Methoden liefern zu einem Zielknoten (target) die Indizes der Knoten, über die der kürzeste Weg verläuft und die Länge des kürzesten Weges. Die Klasse *Point* identifiziert eindeutig einen geographischen Punkt, welcher als Knoten im Graph fungieren kann. Die Klasse *Node* besteht aus einem *Point* und einer Liste weiterer Punkte, die direkte Nachbarknoten darstellen. Die Klasse *Graph* stellte die gesamte Funktionalität zur Verfügung, um aus Geoinformationen (Line-Shape) einen Graph aufzubauen und überflüssige Punkte bzw. Knoten aus dem Graph zu entfernen. Besitzt ein Knoten nur zwei Verbindungen (Kanten) zu zwei verschiedenen Knoten, kann dieser aus dem Graph entfernt werden und eine neue Verbindung zwischen den Nachbarknoten hergestellt werden.

4 Implementierung des DSS

Eingangs der vorliegenden Arbeit wird ein erweitertes Schema zum Aufbau eines forstlichen Entscheidungsunterstützungssystems abgeleitet (Abb. 2). Darauf aufbauend wird die konkrete Implementierung des Softwaresystems vorgenommen. Darüber hinaus wird ein Katalog mit Anforderungen aufgestellt, welche bei der Erstellung des Entscheidungsunterstützungssystems berücksichtigt werden sollen. Die Anforderungen sind in die drei Bereiche *Funktionsumfang*, t*echnische Anforderungen* und *Kosten* gegliedert. Es wird gefordert, dass das Softwaresystem kostenneutral distributiert und angewendet werden kann. Daher werden bei der Konzeption der Software keine Komponenten und Bibliotheken verwendet, die einem kostenpflichtigen Lizenzmodell unterliegen. Ferner wird das DSS so implementiert, dass der Anwender alle notwendigen Funktionen aus einer Oberfläche heraus bedienen kann und nicht auf zusätzliche Software zurückgreifen muss. Da die Software auch verschiedene räumliche Auswertungen und kartographische Darstellungen leisten soll, müssen die entsprechenden Funktionen in das Programm integriert werden (Kap. 4.4.5). Ein

weiterer Schwerpunkt liegt auf dem Aufbau eines Systems, welches die Simulation der Bestandesentwicklung unter Berücksichtigung forstlicher Eingriffe für eine große Anzahl von Beständen automatisiert durchführen kann. Der rechenintensive Simulationsvorgang wird durch entsprechende Programmiertechniken multi-threading-fähig gemacht, so dass besonders Multiprozessor- oder Mehrkernsysteme optimal ausgelastet werden können. Eine optimale Auslastung der zur Verfügung stehenden Rechenkapazität verkürzt deutlich die benötigte Laufzeit rechenintensiver Operationen (Kap. 4.4.7). Die Implementierung des Optimierungssystems erfolgt aufbauend auf den Ergebnissen des Vergleichs verschiedenerer z. T. modifizierter Suchverfahren anhand erreichter Lösungsgeschwindigkeiten und -qualitäten. Dabei werden ebenfalls parallele Berechnungsstrukturen integriert (Kap. 3).

Weiterhin soll das System technisch so konzipiert werden, dass es eine hohe Benutzerfreundlichkeit aufweist, gängige Datenbanksysteme unterstützt und die Anbindung spezieller (nicht standardmäßig vorgesehener) Datenbanken ohne großen Aufwand implementiert werden kann. In diesem Abschnitt wird auf die wichtigsten Schritte zur Implementierung des eingangs spezifizierten DSS eingegangen.

4.1 Wahl der Programmiersprache

Eine wichtige Entscheidung bei der Umsetzung eines Softwareprojektes ist die Wahl der Programmiersprache. Denkbar wären die klassischen Sprachen wie C/C++, Pascal oder Basic. Zur Implementierung des Entscheidungsunterstützungssystems wurde die Programmiersprache Java gewählt, da diese mehrere Vorteile bietet.

Java basiert auf dem JIT-Prinzip (Just-In-Time Compiler). Dies bedeutet, dass beim Erstellen eines Java-Programms zunächst der Quellcode in einen maschinennahen Java-Bytecode übersetzt wird. Erst auf der Zielmaschine wird dieser Bytecode in der JRE (Java Runtime Environment) dem System angepasst. Der wichtigste Bestandteil der Java-Laufzeitumgebung (JRE) ist die Java Virtual Maschine (JVM), die die Programme ausführt, indem sie den Bytecode interpretiert und bei Bedarf kompiliert (Hotspot-Optimierung) (ULLENBOOM 2007). Dies hat zwar den Nachteil, dass das Starten eines Java-Programms länger dauert, bietet aber den großen Vorteil, dass ein Java-Programm grundsätzlich auf jedem Rechner mit einer JRE läuft und somit plattformunabhängig ist. Weiterhin kann die Java-Runtime Umgebung Fehler während der Laufzeit abfangen und behandeln. Der Vorteil der Hotspot-Optimierung liegt darin, dass die JVM immer den gesamten Bytecode kennt und anhand dessen Optimierungsentscheidungen treffen kann, welche die

Laufzeitgeschwindigkeit des Programms erhöhen und gerade bei langlaufenden Anwendungen die längere Startzeit wettmachen kann.

Ein weiterer Vorteil von Java ist die Umsetzung und Unterstützung der objektorientierten Programmierung (OOP). Die objektorientierte Programmierung ist ein Programmierparadigma, welches die Flexibilität und Wiederverwendbarkeit von Programmen verbessert. Daten und Funktionen werden möglichst eng in einem so genannten Objekt zusammengefasst und nach außen hin gekapselt, so dass Methoden fremder oder ausgeschlossener Objekte diese Daten nicht versehentlich manipulieren können. Neben dem OOP wird von Java das sog. Multithreading unterstützt und die Entwicklung parallel-verarbeitender Programme mit verschiedenen vordefinierten Funktionen vereinfacht. Gerade moderne PC-Systeme mit Multi-Core-Prozessoren machen den Einsatz dieser Technologie interessant und notwendig, um die volle Leistung des Systems ausnutzen zu können.

Ein weiterer Aspekt, welcher für den Einsatz von Java spricht, ist die Front-End-Konzeption. Die hier vorgestellte Software, soll u. a. mit einer webbasierten Nutzeroberfläche versehen werden. Zur Realisierung eines solchen servergebundenen Systems sind verschiedene kostenlose, Java-basierte Softwarelösungen verfügbar (Tomcat (THE APACHE SOFTWARE FOUNDATION 2009), Glassfish (SUN 2009b) etc.). Diese Softwarelösungen stellen eine Architektur zur Verfügung, um dynamisch Web-Inhalte zu erstellen und über einen Web-Server zur Darstellung an einen clientseitigen Browser zu senden.

4.2 Softwarekonzeption

Die in den folgenden Kapiteln beschriebenen Modelle und Module müssen zum Aufbau des vollständigen Entscheidungsunterstützungssystems miteinander verknüpft und mit einer Datenbankschnittstelle versehen werden. Zur Ansteuerung der einzelnen Funktionen sowie zur Ergebnispräsentation sollte das System über eine graphische, nutzerfreundliche Nutzerschnittstelle verfügen.

Das vorgestellte DSS bietet dem Anwender drei verschiedene Möglichkeiten zur Entscheidungsunterstützung. Das System soll in der Lage sein, verschiedene Datenquellen (Inventur, FoE) einzulesen und entscheidungsrelevante Indikatoren zu den verschiedenen Waldfunktionen zu berechnen. Darüber hinaus soll die Möglichkeit zur Variantenbewertung und zur simulationsbasierten Optimierung bestehen. Dadurch wird es dem Nutzer ermöglicht, einerseits verschiedene Behandlungskonzepte zu definieren und andererseits die Auswirkungen auf die Bestände anhand der berechneten Indikatoren zu bewerten sowie zu

vergleichen. Auf Basis der Ergebnisse können dann Behandlungsstrategien abgeleitet werden. Eine weitere Anwendungsmöglichkeit besteht darin, unter Zuhilfenahme der simulationsbasierten Optimierung einen Vorschlag für die Lösung eines Planungsproblems zu generieren. Dementsprechend müssen verschiedene Teilmodelle sowie eine Simulations- und Optimierungskomponente kombiniert und mit entsprechenden Auswertungsverfahren versehen werden. Dem vorgestellten System liegt der in Abb. 31 dargestellte Aufbau zugrunde.

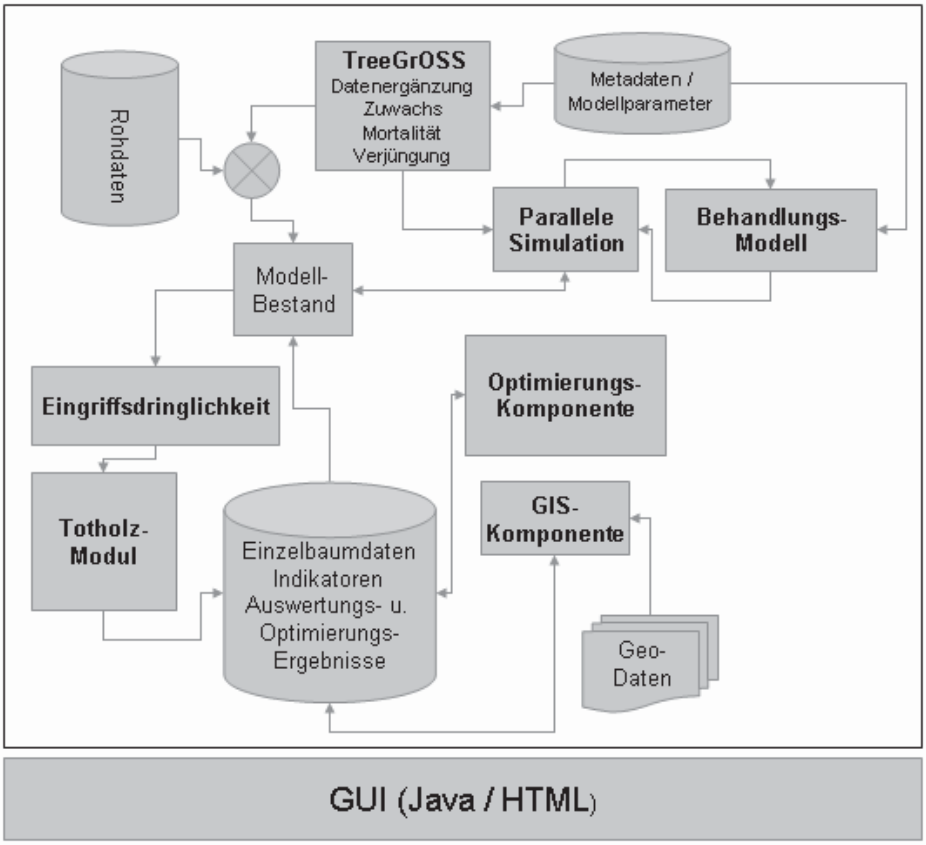


Abb. 31: Vereinfachte Darstellung des Software-Konzepts zur Integration verschiedener Einzelkomponenten.

Das Konzept sieht vor, dass die Benutzeroberfläche strikt vom funktionalen System getrennt implementiert wird. Dadurch ist die Wiederverwendbarkeit des Funktionspaketes in verschiedenen Softwareprojekten mit verschiedenen Front-End-Konzepten leichter umzusetzen. Eine zentrale Rolle im Funktionspaket spielt der Modellbestand. Diese Klasse (*stand*) ist Teil des TreeGrOSS-Paketes (s. u.) und beinhalten die modellrelevanten Einzelbaumdaten. Alle in dem hier vorgestellten System verwendeten Teilkomponenten greifen auf diese Datenstruktur oder auf davon abgeleitete Informationen zu. Aus den sog. Rohdaten (vgl. Kap. 4.4.6) baut das System sequentiell Modellbestände auf und speichert die

teilweise ergänzten Informationen (Stammfußkoordinaten, Baumhöhen, Eingriffsdringlichkeit usw.) in eine angebundene Datenbank. Bei diesem Vorgang werden alle Modellbestände mit einer konstanten Größe von 0,25 ha generiert und die tatsächliche Flächengröße in einer Metatabelle gespeichert. Dadurch wird der Simulationsaufwand deutlich verringert und eine flächenscharfe Berechnung von Mittel- oder Absolutwerten ist weiterhin möglich. Auf diese Daten greifen alle weiteren Komponenten zu. Neben der Auswertung des aus den Rohdaten erzeugten Ausgangszustands können mit den Komponenten Wuchsmodell, Behandlungsmodell und Totholzmodell verschiedene waldbauliche Szenarien simuliert werden. Hierzu wird die Modellkopplung und -anwendung von der Simulationskomponente gesteuert. Das Wuchs- und Behandlungsmodell wird synchronisiert und die komplexen Rechenoperationen parallelisiert, so dass auf verschiedene Threads aufgeteilt, mehrere Modellbestände gleichzeitig aus der Datenbank eingelesen und verarbeitet werden. Die Daten der fortgeschriebenen Bestände werden ebenfalls in der Datenbank gespeichert und können so analysiert und mit Ergebnissen weiterer Szenarien oder dem Ausgangszustand verglichen werden. Beim Speichern der Bestandesdaten werden verschiedene entscheidungsrelevante Parameter ergänzt. Auf den Szenarioergebnissen aufbauend können wiederum weitere Szenarien gerechnet und entsprechend komplexe Behandlungsvarianten auf Betriebs- und Bestandesebene generiert werden. Somit ist es möglich, die verschiedensten Entwicklungspfade für eine große Anzahl einzelner Bestände (z. B. ein Forstamt) aufzubauen, auszuwerten und in Optimierungsprozesse mit einzubeziehen. Zur Darstellung der Ergebnisse kommt u. a. eine GIS-Komponente zum Einsatz. Neben der reinen Visualisierung beinhaltet diese Komponente auch Methoden, um Beziehungen zwischen einzelnen räumlichen Objekten abzuleiten und verschiedene räumliche Mischungs- und Aggregationsindikatoren zu berechnen. Die Optimierungskomponente setzt auf die Modellbestandsdaten auf und leitet auf Basis mehrerer simulierter Entwicklungspfaden eine optimale Kombination ab, so dass eine gesamtbetriebliche Nutzenfunktion maximiert wird. Oder es wird im Rahmen der kurzfristig ausgerichteten Optimierung eine Auswahl an Beständen ermittelt, welche ebenfalls eine konkrete Zielfunktion optimiert. Alle Zielfunktionen werden dialogbasiert nutzerspezifisch aufgestellt. Die Ergebnisse eines Optimierungslaufs werden in dem angebundenen Datenbanksystem gespeichert und können anschließend aufbereitet und dem Anwender über das GUI zugänglich gemacht bzw. präsentiert werden.

4.3 GUI

Für den Anwender ist die Benutzeroberfläche eine der wichtigsten Softwarebestandteile. Sie bildet die Schnittstelle (Benutzerschnittstelle, engl. User Interface, UI, GUI = graphical user interface) zum Programm und soll helfen, Aufgaben leichter zu bewältigen und die Arbeit mit dem Programm effizienter zu gestalten. Demnach hat die Programmoberfläche starken Einfluss auf die Akzeptanz der jeweiligen Software. Ziel der Oberflächengestaltung ist es, eine möglichst hohe Benutzerfreundlichkeit, also die Gebrauchstauglichkeit (*Usability*) der Software für den Anwender zu bieten. Hierzu sollte die Benutzerschnittstelle eines Softwareprogramms dem Anwender einen leichten Zugang zur Programmsteuerung und der Funktionalität der Software ermöglichen. Die Gebrauchstauglichkeit ist in DIN EN ISO 9241 Teil 11 als Produkt von Effektivität, Effizienz und Zufriedenheit definiert. Die Effektivität beschreibt in diesem Zusammenhang den Zielerreichungsgrad. Der Aufwand, welcher betrieben wurde, um das definierte Ziel zu erreichen, ist als Effizienz zu verstehen. Die Einordnung dieser drei Punkte in den Kontext der zu erstellenden Software sollte vor Beginn des Designprozesses erfolgen und festgelegt werden, wie diese Größen quantifiziert werden können.

Neben diesen allgemeinen, problemspezifischen Anforderungen an die Gebrauchstauglichkeit einer Software werden in Teil 10 der Norm sieben Grundsätze zur Dialoggestaltung spezifiziert. Demnach muss eine interaktive Schnittstelle unten stehende Merkmale aufweisen, die jedoch in Abhängigkeit der Zielgruppe (Endanwender) unterschiedlich zu gewichten sind. Die Lernförderlichkeit spielt z. B. bei gut eingearbeiteten Nutzern eine eher untergeordnete Rolle.

- Aufgabenangemessenheit
- Selbstbeschreibungsfähigkeit
- Steuerbarkeit
- Erwartungskonformität
- Fehlertoleranz
- Individualisierbarkeit
- Lernförderlichkeit

In der Literatur werden verschiedene Ansätze vorgestellt, welche grundlegende Prinzipien zur Gestaltung einer effizienten Benutzerschnittstelle aufzeigen (HANSEN 1971, HECKEL 1984, MAYHEW 1992, SCHNEIDERMANN 2001). Einen der ersten Beiträge mit dem Titel „User Engineering Principles for Interactive Systems“ zu diesem Thema lieferte Hansen (HANSEN

1971). Die dargestellte Liste von Prinzipien ist bis heute grundlegend für die Programmgestaltung und findet sich in moderneren Arbeiten zu dem Thema in ähnlicher Form wieder:

- know the user
- minimize memorization
- optimize operations
- engineer for errors

Wie bei Hansen (HANSEN 1971) angeführt, spielt es nach wie vor eine große Rolle, für welche Nutzerkategorie eine Oberfläche entwickelt wird. Es ist kaum möglich, für unterschiedliche Nutzergruppen ein Design zu entwickeln, welches bei allen Anwendern zu einer gleich hohen Gebrauchstauglichkeit führt. Zu beachten sind Hintergrund, Erfahrungen, Training, Bildung, Alter, Kulturkreis, Motivation, Ziele und Persönlichkeit des Nutzers. Weiterhin spielt die Häufigkeit der Softwarenutzung seitens des Endusers (mehrmals täglich, wöchentlich) eine große Rolle bei der Softwareentwicklung. Es ist nahezu unmöglich, eine Gruppe von Nutzern optimal zu bedienen. Wichtig für die Akzeptanz und die Erleichterung der Bedienung ist die Konsistenz einer Anwendung. Darunter ist zu verstehen, dass die Oberfläche und Bedienung sowohl innerhalb eines Programms gleichbleibend gestaltet wird und das Design der Software sich dem Look-And-Feel der graphischen Oberfläche des jeweiligen Betriebssystems anpasst. Nutzern, die die Software regelmäßig nutzen und innerhalb der Software bestimmte Aktionen häufig aufrufen, sollten durch Shortcuts (Tastenkombinationen) und Möglichkeiten, Aktionsfolgen in Makros zusammenzufassen, unterstützt werden. Darüber hinaus tragen auch schnelle Reaktionszeiten dazu bei, die Arbeit für Vielnutzer effizienter und angenehmer zu gestalten. Ein weiterer wichtiger Aspekt bei der Softwaregestaltung sind Rückmeldungen. Diese sollen den User z. B. darüber informieren, ob angeforderte, lang andauernde Aktionen ausgeführt werden und wie der Fortschritt dieser Aktion einzuschätzen ist. Dabei ist es sinnvoll, ein angemessenes Maß an Informationen darzustellen. Die logische Gliederung und Zusammenfassung von Bedienelementen (Buttons, Eingabefelder etc.) zu Dialogen sollen Teilaufgaben sinnvoll abgrenzen und dadurch dem Anwender die Fokussierung auf die momentane Teilaufgabe erleichtern. Selbst erfahrene Nutzer machen Fehler, was die Produktivität merklich senken und die Frustration beim Anwender steigern kann. Eine wichtige und relativ einfache Maßnahme ist es, auf Fehler mit aussagekräftigen Fehlermeldungen hinzuweisen, so dass der Nutzer den Bedienungsfehler erkennt und eine

Korrektur vornehmen kann. Weitaus komplexer ist es, Fehler softwareseitig zu verhindern (fehlerhafte Eingaben automatisch zu korrigieren). Nicht zuletzt soll die Software auch das Ausführen von korrekten Aktionen unterstützen, indem beispielsweise eine automatische Syntaxvervollständigung erfolgt und Menüs kontextsensitiv angepasst werden. Die Möglichkeit ungewollte oder fehlerhafte Aktionen rückgängig zu machen, verhindert Unzufriedenheit des Nutzers und ermutigt darüber hinaus verschiedene Funktionen auszuprobieren, da Aktionen mit unerwünschten Folgen schnell umgekehrt werden können. Um einen leichten Einstieg in ein Softwaresystem zu ermöglichen, sollte das Kurzzeitgedächtnis des Nutzers nicht überlastet werden. Es sollten nicht erst mehrere Befehle erlernt werden müssen, um die Software produktiv einzusetzen. Hierzu ist eine einfache, selbsterklärende Bedienungsführung notwendig. Das Bewältigen von Aufgaben sollte so einfach wie möglich gestaltet werden (z. B. drag and drop, strukturierter überschaubarer Aufbau). Die Verwendung von Metaphern (Schreibtisch, Papierkorb) und Icons (Symbole, Werkzeugleiste) ist dabei sehr beliebt. Der Nutzer kann sich an Dingen aus der realen Welt orientieren und somit Funktionen schneller und leichter wiedererkennen. Die Schreibtischmetapher hat sich bislang am besten behaupten können.

Da bei der Implementierung des vorgestellten Systems die Funktionspakete separat implementiert werden, können problemlos verschiedene Front-End-Konzepte realisiert werden. Im Rahmen der vorliegenden Arbeit werden zwei Benutzeroberflächen entworfen. Es wird eine Einzelplatzanwendung erstellt, welche auf der SWING-Bibliothek (WALRATH et al. 2004) basiert. Diese strikt objektorientierte Bibliothek stellt Methoden und Werkzeuge zur Verfügung, um Bedienelemente und das dazugehörige Verhalten (Reaktion auf Interaktion) zu implementieren. Die Bibliothek ist Bestandteil der Java-Runtime. Unter Windows, Linux und MacOS steht neben dem Standarddesign (Ocean-Look-And-Feel) auch ein System-Look-And-Feel zur Verfügung, welches das betriebssystemtypische Design aufgreift.

Das GUI der Einzelplatzversion (WaldPlaner) richtet sich nach dem Look-And-Feel des jeweiligen Betriebssystems, so dass der Anwender nicht durch ein abweichendes Design der Steuerelemente irritiert wird.

Das Hauptfenster des WaldPlaners gliedert sich in drei Bereiche (Abb. 32). Unter dem Hauptmenü, welches in fast allen fensterbasierten Anwendungen vorhanden ist, sind an der linken Fensterseite der Projektbaum und die Bestandesliste angeordnet.

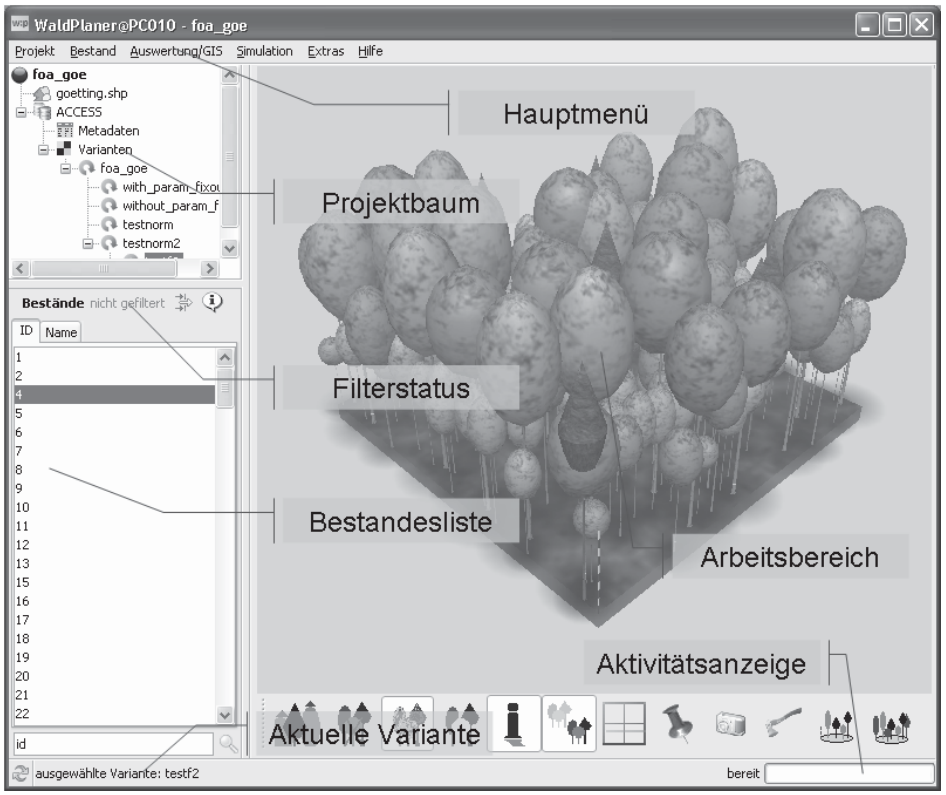


Abb. 32: Hauptfenster des einzelplatzbasierten WaldPlaners.

Der Projektbaum strukturiert alle wichtigen Elemente eines WaldPlaner-Projekts. Wichtige Elemente sind z. B. eingebundene Geodaten, die Metadatentabelle oder alle vorhandenen Varianten. Unter dem Begriff einer Variante wird im Kontext des WaldPlaners eine Menge an Modellbeständen verstanden, welche zu einer Stichprobe oder einem Betrieb gehören. Die Modellbestände können aus Rohdaten generiert werden oder im Rahmen einer Simulation entstehen. In der Regel wird ein Waldzustand zu einem Stichjahr einmal aufgebaut und darauf basierend mehrere Simulationsrechnungen durchgeführt. Die Anordnung innerhalb der Baumstruktur richtet sich nach den Abhängigkeiten der Varianten. Wurden beispielsweise die Varianten B und C durch Simulation von zwei verschiedenen Waldentwicklungsszenarien ausgehend von Variante A gerechnet, so erscheinen die Varianten B und C eine Ebene unterhalb der Variante A. Durch Anklicken der Elemente des Projektbaums wird entweder im Arbeitsbereich eine entsprechende Seite angezeigt (z. B. die Kartendarstellung oder die Metadatentabelle) oder es wird durch einen Klick auf eine Variante diese zur aktuellen Variante, so dass die Bestandesliste und alle Auswertungs- und Simulationsmöglichkeiten sich auf diese beziehen.

Unter dem Projektbaum befindet sich die Bestandesliste. Diese listet alle zu einer Variante gehörenden Bestände mit ihrer ID oder einer individuellen Bezeichnung auf. Durch

Anklicken wird der jeweilige Bestand geladen, so dass sich alle bestandesbezogenen Auswertungen und Darstellungen auf diesen beziehen. Da die Möglichkeit besteht, die Liste auf Bestände einzuschränken, welche bestimmte Kriterien erfüllen, wird über der Bestandesliste der Filterstatus angezeigt (nicht gefiltert/gefiltert). Durch Anklicken des daneben befindlichen Symbols gelangt man direkt zu den Filtereinstellungen. Ein Klick auf das zweite Symbol öffnet ein Fenster, welches Details zum aktuellen Bestand anzeigt. Unter der Liste befindet sich ein Textfeld zum Angeben einer zu suchenden ID.

Auf der rechten Seite des Hauptfensters befindet sich der Arbeitsbereich. In diesem werden je nach Benutzeraktion Informationen, Ergebnisse oder Eingabemasken dargestellt. In Abbildung 32 wird im Arbeitsbereich eine 3D-Ansicht des aktuellen Bestands angezeigt. Da das verwendete Wuchsmodell einzelbaumorientiert ist, können Informationen für die gesamte betrachtete Einheit (z. B. ein Forstamt), für eine oder mehrere Untereinheiten (Bestände) bis hin zum Einzelbaum (z. B. modellhaft ausgehaltene Sortimente) ausgewertet und dargestellt werden.

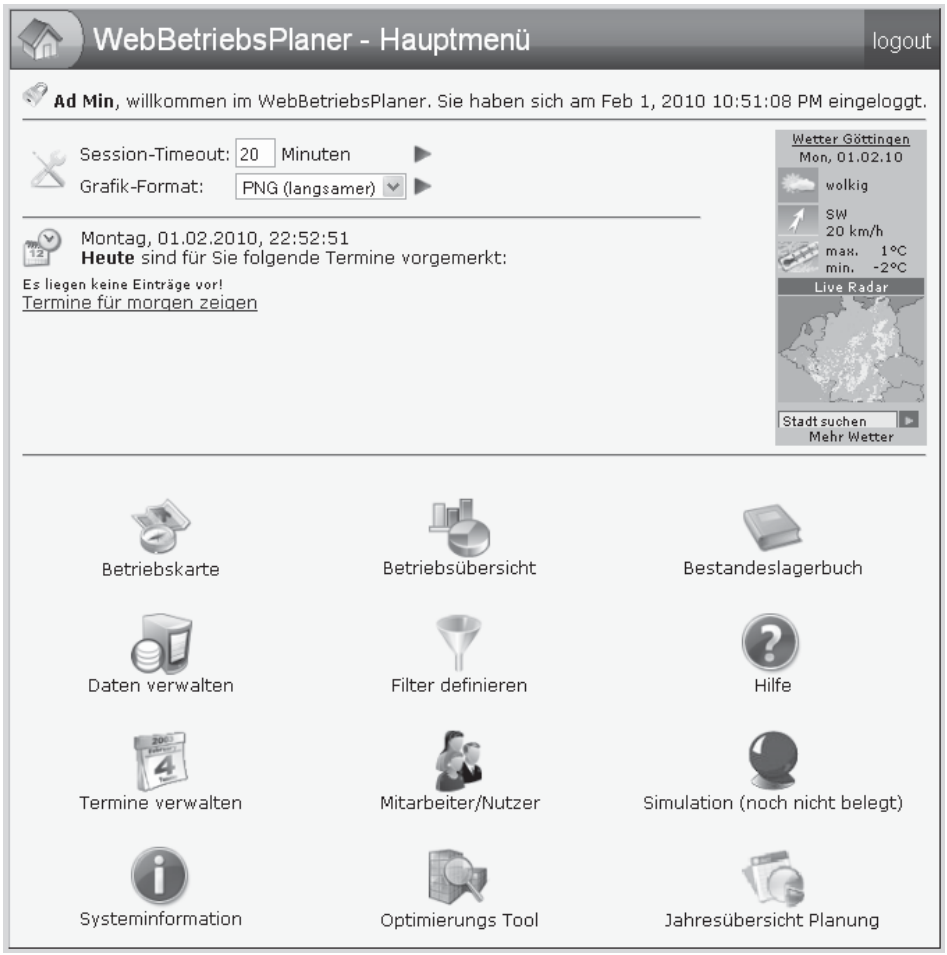


Abb. 33: Hauptfenster des WebBetriebsPlaners.

Die webbasierte Front-End-Variante (WebBetriebsPlaner) stellt durch interaktiv gestaltete HTML-Seiten ebenfalls eine graphische Benutzeroberfläche zur Verfügung Die HTML-Seiten werden serverseitig durch Servlets oder in Form von JavaServerPages (JSP) dynamisch generiert und an den Client-Browser gesendet. Durch die dynamische Erstellung der angefragten Seiten können diese mit variablem Inhalt gestaltet werden. Der Funktionsumfang der webbasierten Variante ist im Vergleich zu der Einzelplatzversion geringer. Letztere richtet sich an erfahrene Anwender, welche z. B. über selbst zusammengestellte Abfragen (SQL) zusätzliche Ergebnisse aggregieren müssen. Die webbasierte Version richtet sich an Anwender, welche wiederholt die gleichen Auswertungsvorgänge abrufen. Von dem Start- bzw. Hauptfenster (Abb. 33) gelangt der Anwender durch Anklicken der dargestellten Symbole zu verschiedenen Unterfenstern. Diese beinhalten u. a. eine Betriebskarte, eine Betriebsübersicht, ein Bestandeslagerbuch und das Optimierungs-Tool.

4.4 Modellauswahl

4.4.1 Wuchsmodell

Da das vorgestellte System verschiedene forstliche Szenarien simulieren und auswerten soll, kommen Biommodelle zur Modellierung des Waldwachstums nicht in Frage. Biommodelle sind zur Abbildung verschiedener Eingriffssensitivitäten modellierter Maßnahmen aufgrund der groben Auflösung ungeeignet (HINRICHS 2006a). Reine Prozessmodelle, welche auch forstlich relevante Größen wie Durchmesser und Höhen ableiten, sind zur Zeit nicht verfügbar, da viele Vorgänge wie die Interaktionen zwischen mehreren Individuen (Bäumen) oder das Höhenwachstum noch nicht prozesshaft nachgebildet werden können (PRETZSCH 2001b). Die verfügbaren (hybriden) Prozessmodelle haben verschiedene Auflösungen. Diese reicht von Modellen, welche die Stoffproduktion von gesamten Beständen abbilden bis hin zu hochauflösenden Einzelbaum- oder Organmodellen. Lediglich Modelle der letzten Gruppe sind geeignet, verschiedene Eingriffe und eine entsprechende Wachstumsreaktion abzubilden. Ein Vorteil dieser Prozessmodelle besteht darin, dass Änderungen der standörtlichen Gegebenheiten sehr gut abgebildet werden können und somit auch längerfristige Simulationen z. B. unter Einbeziehung klimatischer Szenarien möglich sind. Nachteilig bei den hochauflösenden Modellen ist jedoch der Bedarf nach einer sehr detaillierten Datenbasis (Bestandes-, Standort- und Klimainformationen) und nach einer sehr hohen Rechenleistung (PRETZSCH 2001b). So ist auch der Einsatz hochauflösender Prozessmodelle in dem hier vorgestellte System aufgrund der hohen Rechenzeit und der in der praktischen Anwendung

meist nicht verfügbaren Datengrundlage ungeeignet. Zur Simulation der Entwicklung einzelner Bestände wird daher in dieser Arbeit auf die Open-Source-Variante TreeGrOSS (**T**ree **G**rowth **O**pen **S**ource **S**oftware) des statistischen Simulators BWINPro zurückgegriffen (NAGEL 2005, NAGEL 2009). Ein wesentlicher Grund zur Wahl dieses Modells besteht in der Konzeption als Einzelbaumwuchsmodell. Durch die Berechnung des Einzelbaumzuwachses in Abhängigkeit vom ausgeübten Konkurrenzdruck, kann das Modell sehr gut auf unterschiedliche modellierte Eingriffe reagieren und eine Rückkopplung auf den Einzelbaumzuwachs abbilden (vgl. Kap. 2.2.2). So kann im Rahmen der Simulation und Optimierung der Einfluss verschiedener Behandlungsalternativen auf den verbleibenden Bestand berücksichtigt werden. Ein weiterer Vorteil ist die Datenstruktur zum Aufbau und zur Simulation der Modellbestände. Es können Forsteinrichtungsdaten oder (Einzelbaum-) Stichprobendaten verarbeitet werden. Fehlende Einzelbaumwerte werden über eine Datenergänzungsroutine ermittelt. Zur Modellierung der Bestandesentwicklung werden keine weiteren Antriebsdaten, wie etwa Klima- oder Standortdaten benötigt. Neben den einzelnen Wuchsmodellkomponenten (Zuwachs, Einwuchs, Mortalität) und Ergänzungsroutinen beinhaltet die TreeGrOSS-Bibliothek ein Sortierungsmodul. Das auf Schaftformfunktionen (SCHMIDT 2001) basierende System ermöglicht es, die modellierten Einzelbäume in dimensionsabhängige Sortimente zu zerlegen und deren Volumen zu bestimmen. So können Eingriffe noch besser monetär bewertet und im Bestand verbleibende Massen abgeschätzt werden. Letzteres kann z. B. genutzt werden, um die Abschätzung der Totholzdynamik zu verbessern (vgl. Kap. 4.4.3).

Der Kern des Modells TreeGrOSS ist ein wahlweise distanzabhängiges oder distanzunabhängiges Zuwachsmodell, welches den Höhen- und Grundflächenzuwachs sowie die Kronenansatz- und Kronenbreitenveränderung für jeden Baum des Modellbestandes berechnet. Der Durchmesserzuwachs bzw. der Grundflächenzuwachs wird auf Basis der Kronenmantelfläche und der Kronenkonkurrenz sowie deren Veränderung in Folge von Eingriffen und dem Alter geschätzt. Die Kronenkonkurrenz wird über den *C66* abgebildet. Der *C66* eines Modellbaumes ist die Summe der Kronenschirmflächen (ks) aller Bäume in der Schnitthöhe des Bezugbaumes. Die Schnitthöhe berechnet sich aus dem Kronenansatz plus 33 Prozent der Kronenlänge. Liegt der Kronenansatz eines Baumes über der Schnitthöhe, wird die maximale Kronenschirmfläche berücksichtigt. Ist der Baum kleiner als die Schnitthöhe, wird er nicht berücksichtigt.

$$C66_i = \frac{\sum_{i=1}^{n} ks66_i}{A}$$

wobei:
A = Flächengröße [m²]
n = Anzahl Bäume

Gl. 6

Zusätzlich zum *C66* wurde dessen Veränderung durch einen Durchforstungseingriff (*C66c*) in das Modell eingeführt. Mit dem *C66c* kann die Reaktion eines Baums auf eine plötzliche Freistellung beschrieben werden. Findet keine Durchforstung/Freistellung statt, ist der *C66c* 0. Mittels einer baumartspezifischen Funktion zur maximalen Dichte wird der *C66* hergeleitet, ab dem für den Bezugsbaum der Konkurrenzdruck zu hoch wird und dieser abstirbt. Darüber hinaus bietet das Modell die Möglichkeit die Lebensdauer artspezifisch zu begrenzen.

Das Modell wurde für fünfjährige Zuwachsintervalle parametrisiert, lässt aber auch einjährige Prognoseintervallen zu. Bei einem Prognoseintervallen werden folgende Punkte durchlaufen:

- Überprüfung der altersbedingten Mortalität
- Überprüfung der konkurrenzbedingten Mortalität
- Schätzung von Höhen- und Grundflächenzuwachs
- Erhöhung des Alters um x Jahre
- Bestimmung des BHD und der Höhe nach x Jahren
- Statische Anpassung von Kronenansatz und -breite an die neuen Dimensionen

Bei allen Schätzungen kann optional die in der Parametrisierungsgrundlage beobachtete Streuung berücksichtigt werden, welche normalverteilt um den Schätzwert schwankt. So wird bei gleichen Einzelbaumattributen (BHD, Höhe, Alter, Kronenmantelfläche) eine Variation des Zuwachses erreicht.

Zur Schätzung des Einwuchs wird die Bestandesfläche in ein Raster aus 500 m² großen Zellen unterteilt. Zunächst wird mittels eines logistischen Modells für jede Rasterfläche geschätzt, wie hoch die die Wahrscheinlichkeit, dass Einwuchs vorkommt (*pE*) ist. Die Wahrscheinlichkeit wird in Abhängigkeit von der führenden Hauptbaumart und der c66-Klasse (*c66kL*) - dem Lichtangebot - geschätzt. Für die Berechnung des Parameters *c66kL* wird die Kronenschirmfläche aller Bäume, die die Rasterfläche bedecken berücksichtigt. Die Klassenbreite der *c66kL*-Werte beträgt 0,2 und der höchste Klassenwert 2,5.

4.4.2 Behandlungsmodell

Das von Duda vorgestellte System zur automatisierten Umsetzung von verschiedenen Behandlungsszenarien wurde basierend auf dem TreeGrOSS-Modul entwickelt (DUDA 2006). Das hat den Vorteil, dass es sich problemlos, ohne weitere Schnittstellen definieren zu müssen, in Kombination mit diesem Wuchsmodell einsetzen lässt. Das Behandlungsmodell basiert auf einem modularen Konzept. Es werden mehrere Behandlungsmodule definiert, die zu einer Behandlungskette zusammengeschaltet werden. So lassen sich sehr flexibel die unterschiedlichsten Bestandesbehandlungen abbilden und verschiedene Behandlungspfade oder Standardszenarien definieren. Das System bietet den Vorteil, dass in Kombination mit einer Metadatenbank und den einzelnen Bestandesdaten die Parametrisierung der einzelnen Module dynamisch erfolgt und nicht für jeden einzelnen Bestand manuell vorgegeben werden muss. So können in der Metadatenbank spezifizierte Varianten sehr leicht auf eine große Anzahl Bestände angewendet werden. Damit eignet sich das Behandlungsmodell für den Einsatz in einem mittelfristigen Optimierungssystem zur Generierung optimaler Behandlungspfade oder zu einem simulationsbasierten Variantenstudium.

Die einzelnen Behandlungselemente sind vier Bereichen zuzuordnen: Schutz, Endnutzung, Durchforstung und Verjüngung. In Tab. 8 sind die einzelnen Elemente aus diesem Bereich mit ihren art- und bestandsabhängigen Rahmenwerten aufgelistet. In dem System von Duda sind in der Schutzkategorie zwei Elemente definiert. Der Minderheitenschutz erhält einen gewissen Anteil seltener Baumarten in einem Bestand. Die als selten markierten Bäume werden bis zum Erreichen der baumartenspezifischen Zielstärke im Bestand erhalten. Das Element *Habitatbaumauswahl* markiert permanent eine definierte Anzahl von Modellbäumen und unterbindet die Entnahme dieser Bäume. Da auch Totholzmengen in dem vorgestellten System berücksichtigt werden sollen, wird auf Basis eines Totholzmodells (vgl. Kap. 4.4.3) ein drittes Schutz-Element ergänzt. Das Element *Totholz* kann aktiv die Anreicherung liegenden Totholzes steuern. Hierzu werden Abschnitte oder komplette genutzte Stämme in den Totholzpool verschoben und im Verlauf der weiteren Simulation modellhaft zersetzt. Die meisten Maßnahmenelemente sind den Kategorien Endnutzung und Durchforstung zuzuordnen. Diese Elemente steuern die Entnahme von Bedrängern und erntereifen Bäumen sowie die Auswahl von Zukunftsbäumen. Die wichtigsten Steuerparameter sind dabei die Zielbaumartenzusammensetzung und minimale bzw. maximale Entnahmemengen.

Tab. 8: Verwendete Maßnahmenelemente und ihre Zuordnung zu den Kategorien Schutz, Endnutzung, Durchforstung und Verjüngung. Nach DUDA (2006).

Kategorie	Maßnahmenelement	Relevante Rahmenwerte auf Artenebene	Relevante Rahmenwerte auf Bestandesebene
Schutz	Minderheitenschutz		Eingriffsturnus, Schutz von Minderheiten
	Habitatbaumauswahl		Anzahl Habitatbäume
Endnutzung	Prüfe Endnutzungsmasse		Eingriffsturnus, minimales Endnutzungsvolumen
	Zielstärkennutzung	Zielstärke	Eingriffsturnus, maximales Endnutzungsvolumen
	Zielstärkennutzung mit Samenbäumen	Zielstärke, Ziel-Artenanteil	Eingriffsturnus, maximales Endnutzungsvolumen
	Zielstärkennutzung mit bedingten Samenbäumen	Zielstärke, Ziel-Artenanteil	Eingriffsturnus, maximales Endnutzungsvolumen
	Entnahme konkurrierender Z-Bäume	Zielstärke	Eingriffsturnus, maximales Endnutzungsvolumen
	Entnahme einer erntereifen Schicht	Ziel-dg	Eingriffsturnus, maximales Endnutzungsvolumen, Endnutzungszeitraum
Durchforstung	Prüfe Durchforstungsmasse		Eingriffsturnus, minimales Durchforstungsvolumen
	Prüfe Eingriffsmasse		Eingriffsturnus, minimales Eingriffsvolumen
	permanente Z-Baum-Auswahl nach Zieltyp	Mindesthöhe für den ersten Eingriff	Eingriffsturnus, angestrebter Zieltyp
	permanente Z-Baum-Auswahl nach aktuellem Mischungsverhältnis	Mindesthöhe für den ersten Eingriff	Eingriffsturnus
	temporäre Z-Baum-Auswahl nach Zieltyp	Mindesthöhe für den ersten Eingriff	Eingriffsturnus, angestrebter Zieltyp
	temporäre Z-Baum-Auswahl nach aktuellem Mischungsverhältnis	Mindesthöhe für den ersten Eingriff	Eingriffsturnus
	Freistellung der permanenten Z-Bäume	Freistellungsgrad	Eingriffsturnus, maximales Durchforstungsvolumen, maximales Eingriffsvolumen
	Freistellung der permanenten Z-Bäume von im Zieltyp nicht vorgesehenen Arten	Freistellungsgrad, Ziel-Artenanteil	Eingriffsturnus, maximales Durchforstungsvolumen, maximales Eingriffsvolumen, angestrebter Zieltyp
	Freistellung der temporären Z-Bäume	Freistellungsgrad	Eingriffsturnus, maximales Durchforstungsvolumen, maximales Eingriffsvolumen
	Freistellung der temporären Z-Bäume von im Zieltyp nicht vorgesehenen Arten	Freistellungsgrad, Ziel-Artenanteil	Eingriffsturnus, maximales Durchforstungsvolumen, maximales Eingriffsvolumen, angestrebter Zieltyp
Verjüngung	Pflanzung	Ziel-Artenanteil	angestrebter Zieltyp

Die Endnutzung kann einzelstammweise erfolgen oder es wird eine komplette erntereife Schicht entnommen. Der Beginn von Durchforstungen richtet sich nach der Höhe der zu

durchforstenden Schichten. Es wird eine Mindesthöhe definiert, ab welcher die Durchforstungsroutinen angewendet werden. Zur Steuerung der Verjüngung von Beständen steht das Element *Pflanzung* zur Verfügung. Dieses Element führt gezielt virtuelle Pflanzungen durch, wenn zu erwarten ist, dass eine definierte Artenmischung (WET = Waldentwicklungstyp) nicht auf Basis der vorhandenen natürlichen Verjüngung (modelliert/erhoben) erreicht werden kann. Der Pflanzungszeitpunkt richtet sich nach der Hiebsreife des jeweiligen Bestands. Diese wird anhand der Durchmesserstruktur der führenden Schicht beurteilt. Hat ein definierter Anteil der Bäume 80 Prozent der Zielstärke erreicht, wird die Endnutzung eingeleitet.

4.4.3 Totholzmodell

Da mit dem ausgewählten Wuchsmodell BWINPro/TreeGrOSS Bestände einzelbaumweise modelliert werden, ist es in Kombination mit der im TreeGrOSS-Paket mitgelieferten Sortierroutine und dem o. g. Nutzungsmodell möglich, die Auswirkungen verschiedener Sortierungsvorgaben und Nutzungs- bzw. Pflegestärken auf die Entwicklung des Totholzes (stehend und liegend) abzubilden. Das hierzu entwickelte Modul kann Totholz, welches im Rahmen der natürlicher Mortalität anfällt, abbilden. Darüber hinaus kann Totholz „aktiv“ gebildet werden. Dies wird erreicht, indem definierte Abschnitte eines entnommenen Stamms im Modellbestand belassen werden. Die Zersetzung des liegenden Totholzes wird baumartengruppenspezifisch nach dem von Meyer (MEYER et al. 2009) vorgestellten Verfahren modelliert, welches den Volumenabbau ähnlich schätzt, wie das bisher in BWINPro verwendete Modell von Müller-Using (MÜLLER-USING 2005), jedoch auf einer breiteren Datengrundlage parametrisiert wurde und zusätzlich die Bruchwahrscheinlichkeiten von stehenden Bäumen oder Hochstümpfen abschätzen kann. Die Zersetzung bzw. Volumenreduktion wird dabei mittels einer linearen Abbaufunktion geschätzt (Gl. 7). Aufgrund des Modelltyps (linear) kann V_t für kleine t geringfügig größer als V_0 werden. In diesen Fällen wird V_t gleich V_0 gesetzt.

$$V_t = \begin{cases} V_0 & V_t > V_0 \\ V_0 - k \cdot t & sonst \end{cases}$$

Gl. 7

wobei:
V_t = Volumen zum Zeitpunkt t
V_0 = Ausgangsvolumen
k = Abbaurate (baumartengruppenspezifisch)
t = Zeit in Jahren

Durch eine entsprechende Definition der Sortierungsvorgaben kann gezielt Totholz mit definierten Dimension (Durchmesser, Länge) angereichert werden. Dies kann zum einen ökologisch bedeutsam sein und zum anderen einen finanziellen[8] Hintergrund haben.

Neben der Zersetzung liegenden Totholzes, kann das Modul auch stehendes Totholz abbilden. Dieses entsteht durch konkurrenzbedingte Mortalität und vor allem durch Extremereignisse (Stürme, Kalamitäten). Letztere sind jedoch in dem gewählten Wuchsmodell noch nicht implementiert. In diesem System fällt stehendes Totholz lediglich durch die konkurrenz- bzw. altersbedingte Mortalität an. Für das stehende Totholz wird nach jedem Wachstumszyklus eine Bruchwahrscheinlichkeit berechnet (Gl. 8).

$$P_{Bruch} = 1 - e^{\left(-\left(\frac{t-\theta}{\sigma}\right)^{c}\right)}$$

Gl. 8

wobei:
P_{Bruch} = Bruchwahrscheinlichkeit
θ = Schwellenwert Parameter
σ = Maßstabs-Parameter (charakteristische Lebensdauer)
c = Form-Parameter
t = Zeit in Jahren

Dieser Wert gibt an, wie hoch die Wahrscheinlichkeit ist, dass der jeweilige stehende Stamm oder Stumpf umbricht. Stehendes Totholz wird nicht wie das liegende Totholz modellhaft zersetzt. Erst nach dem Umbrechen greifen die oben beschriebenen Funktionen zur Schätzung der Volumenabnahme. Um den Zeitpunkt des Umfallens zu bestimmen, kommt eine baumartengruppenspezifische Weibullfunktion zum Einsatz. Mit dieser Funktion wird für alle toten stehenden Bäume nach jedem Simulationsintervall berechnet, wie hoch die Wahrscheinlichkeit ist, dass der entsprechende Baum umfällt. Diese Wahrscheinlichkeit wird mit einer Zufallszahl verglichen. Ist die Bruchwahrscheinlichkeit höher, wird der Baum dem Pool des liegenden Totholzes zugefügt und im Verlauf der weiteren Simulation die Volumenabnahme modelliert.

4.4.4 Modell zur Beschreibung von Sturmschäden

Biotische und abiotische Risiken wie beispielsweise Insektengradationen, Sturmschäden oder Waldbrände müssen bei waldbaulichen Managemententscheidungen mit einbezogen werden (HOLTHAUSEN et al. 2004, DUDA 2006). Die direkten und indirekten Schäden durch Windwurf oder Windbruch haben weit reichende betriebliche und wirtschaftliche

[8] In manchen Bundesländern wird der Erhalt von Totholz z. T. dimensionsabhängig finanziell gefördert (SACHSENFORST 2008).

Konsequenzen. Die Stürme der 1990er Jahre in Mitteleuropa (*Vivian*, *Wiebke* und *Lothar*) verursachten rund 340 Millionen Kubikmeter Schadholz, welches aufgearbeitet und vermarktet werden musste (SCHELHAAS 2008). Der letzte schwere Orkan mit deutlichen Sturmschäden in Mitteleuropa war *Kyrill* am 18. Januar 2007 mit einem Schadholzaufkommen von ca. 100 Millionen Kubikmeter (KRONAUER 2007). Um das Windwurfrisiko zu quantifizieren und dadurch eine risikobezogene Entscheidungsgrundlage in das vorgestellte DSS zu integrieren, wird das Windwurf-Modell von Schmidt (SCHMIDT et al. 2006a, SCHMIDT et al. 2010) aufgegriffen. Der Ansatz basiert auf einem generalisierten, additiven, gemischten Modell (gamm).

Gl. 9

$$g(\pi_{ijk}) = art_{ijk}^{T}\beta_{1art} + \log\left(\frac{d_{ijk}^{\,art_{ijk}\alpha_{1art}}}{h_{ijk}^{\,-art_{ijk}\gamma_{1art}}}\right) + \beta_2 ttd1_{ij} + \beta_3 ttd2_{ij} + \beta_4 ttd3_{ij} + \beta_4 ttd4_{ij} + f_1(rechts_{ij}; hoch_{ij})$$

wobei:
Art =Baumartengruppe
d = BHD /dg
h = Baumhöhe /hg
ttd1 – ttd4 = TOPEX to distance, jew. Summe aus zwei Radien in definierte Richtungen zur Sturmrichtung
rechts = Rechtswert (Gauss-Krüger)
hoch = Hochwert (Gauss-Krüger)
α_{1art}, β_{1art}, γ_{1art} = Vektoren von Regressionskoeffizienten
β_2, β_3, β_4, β_5 = Regressionskoeffizienten
f_1 = zweidimensionale Glättungsfunktion

Als Prädiktoren fungieren die Exponiertheit des Standorts (mod. TOPEX-Index), die Baumart und die Baumhöhe, sowie der BHD. Bei der Schätzung der Eintrittswahrscheinlichkeit von Sturmschäden für die Baumart Fichte kann noch zusätzlich der Vernässungszustand des Standorts berücksichtigt werden. Die räumliche Lage innerhalb des zur Parametrisierung verwendeten Windfeldes muss vorgegeben werden und beschreibt die Stärke des Sturms. Ebenfalls muss eine zu erwartende Hauptrichtung des Sturms angenommen werden. Auf Basis dieser Information kann in Kombination mit einem DGM (digitales Geländemodell) die Exponiertheit berechnet werden. Hierzu wird in acht definierten Richtungen zur Hauptrichtung des Sturms der Winkel zum sichtbaren Horizont in bis zu 1000 Meter Entfernung bestimmt. Die Berechnung des TOPEX erfolgt automatisch in der GIS-

Komponente des vorgestellten Systems (Kap. 4.4.5). Die GIS-Komponente berechnet ebenfalls die Verschneidung der rasterbasierten TOPEX-Werte und der Bestandespolygone. Somit kann für das untersuchte Gebiet in der Auflösung des verwendeten DGMs eine rasterbasierte Windwurfrisikokarte berechnet werden. Diese kann wiederum mit einem einzelnen Bestand verschnitten werden, so dass einzelbestandesweise für die vertretenen Baumartengruppen und Bestandesschichten das Windwurfrisiko quantifiziert werden kann.

4.4.5 GIS-Komponente

Forstliche Entscheidungsunterstützungssysteme beinhalten oft eine sog. GIS[9]-Komponente oder -Schnittstelle, da die Verbindung von Informationen mit einem Raumbezug gerade im Bereich der Planung waldbaulicher und forsttechnischer Maßnahmen sowie der Abfuhrlogistik besonders hilfreich ist. Neben der einfachen Visualisierung und Kartenerstellung können auch Planung und Controlling effizient unter Einbeziehung von GIS-Software gestaltet werden (STÖCKER 2008). Der Wirtschaftswald ist in Deutschland größtenteils in Planungs- bzw. Bewirtschaftungseinheiten mit einer mehr oder weniger homogenen Bestandesstruktur gegliedert (GADOW 2003). Die Bewirtschaftung des Waldes wird dementsprechend maßgeblich durch die räumliche Struktur dieser nutzungsorientierten Planungseinheiten (Abteilung, Unterabteilung, Unterfläche u. Hilfsfläche) bestimmt. Waldbauliche Maßnahmen werden in Bezug auf einen Bestand oder zusammenfassend für ähnliche Bestände definiert. Weiterhin können wichtige Informationen zur Erschließung oder Topographie unter Verwendung von raumbezogenen Funktionen in die Planung mit einbezogen werden. So sind neben der Maßnahmenplanung beispielsweise auch im Bereich des Poltermanagements, der Harvester-Einsatzplanung, der Abfuhrplanung oder bei der Planung und Kontrolle von Naturschutzmaßnahmen erhebliche Potentiale für den Einsatz geographischer Informationssysteme vorhanden.

Die beiden implementierten Anwendungen (Einzelplatz und web-basiert) greifen auf ein extra erstelltes GIS-Modul zurück. Für die Einzelplatzversion wäre auch die Nutzung einer frei verfügbaren Softwarebibliothek denkbar. Beispielsweise stellt die Java-Bibliothek *GeoTools* (GEOTOOLS 2009) mehrere Klassen und Methoden zum Einlesen, Verarbeiten und Darstellen von Geodaten zur Verfügung.

[9] GIS = Geographisches Informationssystem. Sachdaten werden mit einem Raumbezug versehen und können in einem GIS dargestellt, analysiert und weiterverarbeitet (Geoprozessing) werden.

Der WebBetriebsPlaner benötigt u. a. für den serverseitigen Einsatz jedoch folgende Eigenschaften und Funktionen, die zurzeit nicht alle in einer frei verwendbaren Standardbibliothek zur Verfügung stehen:

- Lesen von Geodaten im ESRI-Shapefile-Format (Polygon, Point, Line, Mixed)
- Lesen von GeoTiff/Jpeg und ESRI-Rasterdaten
- Umwandeln von Vektordaten in Rasterdaten
- Multi Layer: Mehrere Layer (Karten mit unterschiedlichen Themen) gleichzeitig darstellen
- Offscreen Rendering: Generieren der Karten im Arbeitsspeicher ohne Monitorkontext
- Alpha-Kanal: Die Flächen/Polygone werden mit einer definierten Transparenz gerendert.
- Image Streaming: Die Bilder/Karten müssen in einen serialisierbaren Datenstrom geschrieben werden können, um sie an einen Client zu übertragen.
- Daten-Buffer: Vorhalten aller Geometriedaten im Arbeitsspeicher → enorme Reduktion der Antwortzeit bei häufigen Clientanfragen
- Datenbankschnittstelle: Lesen und Speichern von Geometrien in einer Datenbank sowie Lesen von Sachdaten aus einer Datenbank und Kombination mit den Geometrien.

Neben einer ansprechenden und schnellen Darstellung von verschiedenen forstlich relevanten Entscheidungsvariablen soll die GIS-Komponente auch Funktionen beinhalten, welche Lage- und Nachbarschaftsparameter ermitteln und verschiedene Kennwerte (Hangneigung, Exposition, Exponiertheit) aus einem digitalen Geländemodell berechnen. Diese Parameter werden vor allem im Rahmen der Optimierung benötigt. Die für die GIS-Komponente erforderlichen Klassen werden in einer Java-Bibliothek zusammengefasst. Die Bibliothek beinhaltet die vier Packages *RasterData*, *VectorData*, *Shader* und *Tools* (Abb. 34). In dem Raster-Package sind die Klassen zusammengefasst, die zur Verarbeitung von Rasterdaten benötigt werden. Das Interface GeoImage wird von den Klassen GeoTIF und GeoJPG implementiert. Diese Klassen beinhalten die notwendigen Methoden und Datenfelder, um bildformatbasierte Rasterdaten einlesen, speichern und verarbeiten zu können. Neben der Bilddatei muss auch eine Referenzdatei vorhanden sein, welche Transformationsparameter (Position, Pixelgröße, Bilddrehung) beinhaltet, um einen korrekten Raumbezug herstellen zu können. Da die beiden Klassen das Interface GeoImage implementieren, ist es möglich, weiterverarbeitenden Methoden (Darstellung, Auswertung) lediglich ein Objekt vom Typ GeoImage zu übergeben. So wird die Weiterverarbeitung von den konkreten Implementierungen losgelöst. Dies ermöglicht es, die aktuell unterstützten Formate Tiff und Jpeg beliebig zu erweitern, ohne den Quellcode der auf das Interface zurückgreifenden

Methoden ändern zu müssen. Die Klasse GridASCII beinhaltet ähnliche Methoden, um Rasterdaten in dem von ESRI spezifizierten Format zu verarbeiten. Dieses Format speichert die Rasterdaten und die Metainformationen (Rastergröße, Koordinaten zur Positionierung des Rasters, Nodata-Value) in einer ASCII-Datei. Das Package *VectorData* fasst alle Klassen zusammen, die benötigt werden, um Daten im vektorbasierten Shapefile-Format verarbeiten zu können. Die zentrale Klasse *Shapefile* stellt Methoden bereit, um die Daten der drei zum Shapefile-Format gehörenden Dateien x.shp, x.shx, und x.dbf einzulesen und auf entsprechende (Java-) Datenobjekte zu übertragen.

GIS (L)		
	RasterData (P)	
		GeoImage (I) *GeoTIF (← GeoImage)* *GeoJPG (← GeoImage)* *GridASCII (C)*
	VectorData (P)	
		Shapefile (C) *ShapeObject* (C) *BoundingBox* (C) *Record* (C) *RecordField* (C) ...
	Shader (P)	
		Shader (I) *BTShader* (← Shader) *BTAgeShader* (← Shader) *ShapeValueShader* (← Shader) *SmoothValueShader* (← Shader) *BooleanShader* (← Shader) ...
	Tools (P)	
		RasterTools (C) *ShapeTools (C)* *MapRenderer (C)* ...

L= Library, P= Package, C= Class, I= Interface, ← = implementiert nachstehendes Interface oder erweitert die Klasse

Abb. 34: Der Aufbau der Klassenbibliothek der GIS-Komponente (nur die wichtigsten Klassen)

Die wichtigsten Datenobjekte sind jeweils als eigene Klasse implementiert. In den Objekten *Record* und *RecordField* werden die zu einem Shape (Punkt, Linie, Polygon) gehörenden Sachdaten gespeichert. Die Vektoren bzw. Koordinaten, welche zu einem Shape gehören, werden in dem *ShapeObject* in einem Array vom Type **double** organisiert. Wird ein Shapefile

eingelesen, werden alle Daten (Geometrien und Sachdaten) in den entsprechenden Objekten gekapselt und sind somit im Arbeitsspeicher vorhanden. Werden gleichzeitig mehrere Dateien verarbeitet, ist ein ausreichend großer Arbeitsspeicher erforderlich (mind. 1 GB). Der Vorteil, der sich daraus ergibt, ist eine sehr hohe Verarbeitungsgeschwindigkeit bei Darstellungs- und Analyseoperationen.

Eine einfache aber dennoch wichtige Klasse ist die *BoundingBox*. Dieses Objekt speichert die notwendigen Informationen, um ein Rechteck zu beschreiben, welches gerade so groß ist, dass alle Koordinaten des zugehörigen Shape-Objekts hineinfallen. Die BoundingBox ermöglicht es, raumbezogene Abfragen deutlich schneller zu gestalten. Soll z. B. ermittelt werden, ob ein Punkt in einem Polygon liegt, wird zunächst geprüft, ob der Punkt überhaupt in der BoundingBox liegt. Ist dies der Fall, kann die wesentlich rechenintensivere Prüfung, ob der Punkt auch in dem entsprechenden Polygon liegt, vorgenommen werden.

Für die Darstellung der Sachdaten durch verschiedene Farben, Schraffuren oder Symbole, kommen sog. Shader zum Einsatz. In dem Package *Shader* sind das Interface Shader und alle darauf basierenden Implementierungen gespeichert. Auch hier ist die Interface-Verwendung sinnvoll, da so der render-Methode jeder beliebiger Shader übergeben werden kann, wenn dieser sich von dem Interface *Shader* ableitet.

In dem Package *Tools* sind zwei Klassen enthalten, welche einmal für Vektordaten (ShapeTools) und einmal für Rasterdaten (RasterTools) alle benötigten Auswertungs- und Analysefunktionen beinhalten. So stellt die Klasse RasterTools Methoden zur Verfügung, welche die Hangneigung, die Exposition, die Exponiertheit und die in Kap. 4.5.2 beschriebene Betriebstypendurchmischung berechnen. Weiterhin beinhaltet die Klasse Funktionen zum Verschneiden von Vektor- und Rasterdaten. Die *ShapeTools* beinhalten vor allem Funktionen, um die Lage von Shape-Objekten zueinander zu beschreiben. Diese werden beispielsweise zur Bestimmung der Aggregation einzelner Bestände benötigt (Kap. 4.5.5). Eine weitere wichtige Funktion für die Darstellung von Polygonen mit Löchern oder Inselpolygonen ist die Identifikation des Umlaufsinns der einzelnen Punkte eines Polygons. Im Format von ESRI werden die Punkte von Inselpolygonen oder Löchern entgegen dem Uhrzeigersinn gespeichert, die Punkte „normaler“ Polygone im Uhrzeigersinn (ESRI 1998). Der Umlaufsinn wird über die Summe der Kreuzprodukte benachbarter Vektoren (Koordinaten oder

Eckpunkte des Polygons) bestimmt. Ist die Summe negativ, verläuft der Umlaufsinn entgegen dem Uhrzeigersinn.

Der *MapRenderer* ist eine der umfangreichsten Klassen der gesamten GIS-Bibliothek. Diese Klasse beinhaltet alle Methoden, um die verschiedenen Shape-Objekte zu skalieren und darzustellen. Da mehrere Layer (Auflieger, Raster- oder Vektordaten) mit einstellbarer Transparenz dargestellt werden sollen, musste eine spezielle Methode zum Zeichnen von Polygonen, welche Inselpolygone oder Löcher beinhalten, erstellt werden. In Abb. 35 wird anhand eines Code-Fragments dargestellt, wie der Rendervorgang implementiert wurde. Zunächst überschreibt der *MapRenderer* die Methode *PaintComponent* der erweiterten Klasse *JComponent*. In dieser Methode wird geprüft, ob eine Veränderung der Kartendarstellung vorliegt (Zoomen, Verschieben, anderer Shader ausgewählt). Andernfalls kann die Karte aus dem Buffer gerendert werden, da z. B. eine Nutzeraktion stattfindet, die es erfordert, lediglich über der unveränderten Karte zu zeichnen. Eine solche Aktion wäre beispielsweise das Markieren eines Zoom-Ausschnitts. Das Zeichnen aus dem Buffer läuft wesentlich schneller ab, da dieser die komplett gerenderte Karte als Image speichert und so keine Transformations- und Zeichenoperationen durchgeführt werden müssen. Wurden die Transformation (Zoom, Kartenausschnittsposition) oder die darzustellenden Geodaten geändert, muss die Karte neu berechnet und in den Buffer gespeichert werden. Ist dies der Fall, werden alle Transformationsparameter berechnet (*calculateScale*). Anschließend werden zunächst alle spezifizierten Rasterdaten und danach alle Vektordaten (*scaleAndPaint*) transformiert und in den Buffer in der angegebenen Reihenfolge gezeichnet. Zuletzt werden graphische Objekte wie ein Maßstab, ein Nordpfeil oder eine Legende gerendert. Nun kann der Buffer in den Graphikkontext des MapRenderers geschrieben und somit auf einem Bildschirm oder einem Drucker ausgegeben oder in einen Datenstrom zur Übertragung in einem Netzwerk geschrieben werden. Von der Methode *scaleAndPaint* werden zunächst alle Polygone mit Löchern gezeichnet. Danach werden in einer weiteren Iteration die Polygone ohne Löcher des jeweiligen Layers gerendert, da hierzu zwei verschiedene Methoden aufgerufen werden müssen. Die Methode *scaleAndPaintPoly* rendert Polygone ohne Löcher. Es soll eine transparente Darstellungen möglich sein, so dass durch einen oben liegenden Auflieger, alle darunter liegenden Layer zu einem bestimmten Anteil erkennbar sind. Hierzu muss die Pixelverschneidung im Bereich der Löcher so berechnet werden, dass der Farb- und Transparenzwert des Mutterpolygons (in welchem das Loch liegt) in diesem Bereich nicht berücksichtigt wird.

```
    @Override -> der Renderer erweitert die Klasse JComponent
    public void paintComponent(Graphics g){
        if(this.shp[0]!=null){
            if(isDrawingBounds || isDrawingMeassure){
                [...]
                zeichne Messpunkte oder Zoombox
                zeichne Karte aus Buffer bei Nutzerinteraktion
            }
            else{
                zeichne zunächst die Karte in einen Buffer:
                calculateScale();
                Graphics gb=buffer.getGraphics();
                [...]
                zeichne Rasterdaten
                [...]
                for(int i=0; i<render_ord.length; i++)
                    scaleAndPaint(gb, render_ord[i]);
                [...]
                zeiche weitere Kartenelemente
                Nordpfeil, Legende, Maßstab
                [...]
                }
                zeiche den Buffer in den Zeichenkontext des Renderers
                g.drawImage(buffer,0,0,null);
                [...]
```

```
private void scaleAndPaint(Graphics g, int index){
    if(shp[index]!=null){
        [...]
        zeichne zuerst alle Objekte außer den Polygonen ohne Löcher:
        while(itrShapeObjects.hasNext()){
            obj = (ShapeObject)itrShapeObjects.next();
            type=obj.getType();
            switch(type){
                [...]
                zeichne Objekte vom Typ Punkt oder Linie
                case ShapeObject.POLYGON:
                    if(obj.hasCounterclockwise) -> wenn Polygon mit Loch
                        scaleAndPaintPolygonWithHoles(g, obj, sel, index);
                [...]
            }
            [...]
            zeichne Polygone ohne Inselpoygone
            [...]
            zeichne selektierte Objekte mit entsprechender Randfarbe
            [...]
```

```
private void scaleAndPaintPolygonWithHoles(Graphics g, ShapeObject o,
boolean sel, int shpi){
        Graphics2D g2d= (Graphics2D)g;
        int[] x,y;
        int np=o.getPointCount(); -> Anzahl Vektoren im Polygon (o)
        [...]
        GeneralPath kann je nach Umlaufsinn der Punkte
        (im Uhrzeigersinn oder dagegen) erkennen, ob ein Polygon
        ein Loch darstellt:
        GeneralPath a= new GeneralPath(GeneralPath.WIND_NON_ZERO);
        for(int i=0; i<o.getPartCount(); i++){
            [...]
            schreibe alle Punkte des Teilpolygons in die Arrays x und y
            a.append(new Polygon(x,y,ninp), sel);
        }
        setze Linienstärke
        Setze Shader und Farben;
        g2d.draw(a); -> zeichne alle (Teil-)Polygone eines Shape-Objekts
        [...]
```

Abb. 35: Verkürzter Programmcode zum korrekten Rendern von Polygonen mit Inselpolygonen oder Löchern.

Dabei kann auf die Java-Klasse *GeneralPath* zurückgegriffen werden. Diesem Objekt können bei entsprechender Parametrisierung Polygone mit einem Umlaufsinn im und entgegengesetzt des Uhrzeigersinns übergeben und Löcher korrekt gerendert werden. In Abb. 36 ist zu erkennen, dass Löcher und evtl. darin liegende Inselpolygone korrekt gezeichnet werden.

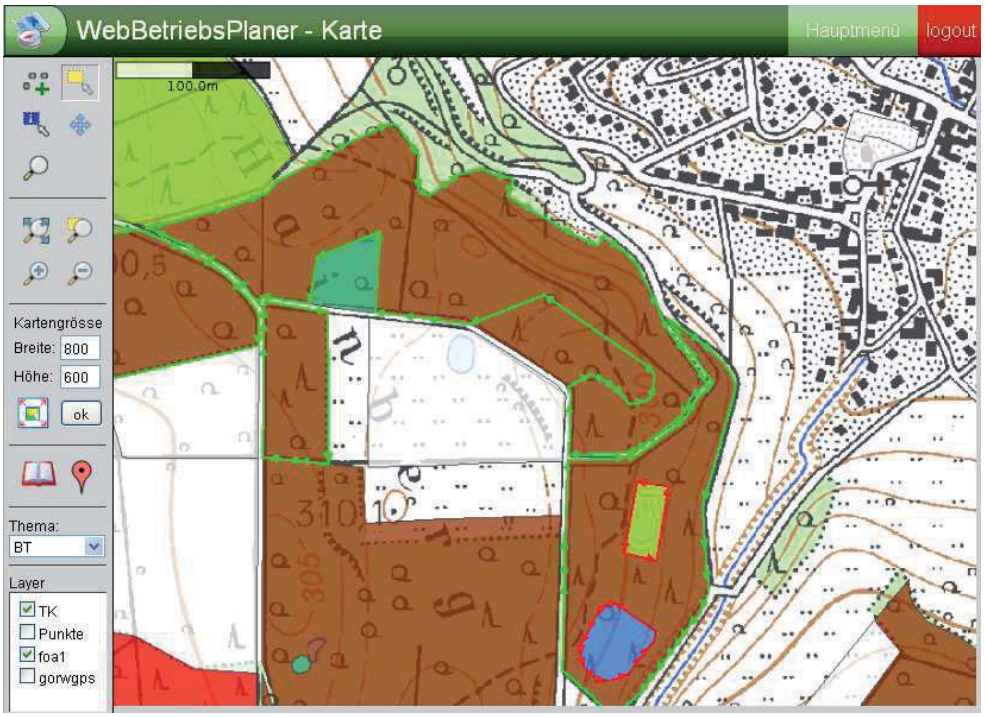


Abb. 36: GIS-Seite des WebBetriebsPlaners. Die Flächenpolygone sind nach dem Betriebstyp eingefärbt. Die transparente Darstellung erlaubt es z. B. eine topographische Karte zu hinterlegen.

Das momentan selektierte Polygon besteht aus mehreren Teilflächen. Selektierte normale Polygone werden mit einer grünen Grenze und den Umlaufsinn anzeigenden Pfeilen dargestellt. Löcher werden rot umrandet. In der Abbildung sind zwei Löcher vorhanden in denen jedoch zwei Inselpolygone liegen. Diese sind entsprechend ihres Betriebstyps korrekt eingefärbt und man kann ebenfalls die darunter liegende Karte erkennen. Bei einer falschen Darstellungsweise würde sich die braune Farbe des Mutterpolygons mit der blauen bzw. hellgrünen Einfärbung der Inselpolygone vermischen und der Alpha-Wert würde verdoppelt werden. D. h. die Transparenz würde abnehmen und die darunter liegenden Layer schlechter erkennbar sein.

4.4.6 Datenmanagement

Um die vorgestellten (Teil-)Modelle nutzen zu können, bedarf es Daten zu den einzelnen Beständen einer Auswertungseinheit (z. B. Revier, Genossenschaft, Forstamt/Betrieb). Es

müssen ausreichende Informationen vorliegen, so dass jeder reale Bestand durch einen Einzelbaum-Modellbestand abgebildet werden kann. Dies ist erforderlich, da das einzelbaumbasierte Wuchsmodell TreeGrOSS bzw. BWINPro verwendet wird (Kap. 4.4.1). Sollen räumliche Informationen in die Auswertung mit einbezogen werden, können Geodaten zu den Bestandesgeometrien, dem Wegenetz und zum Gewässersystem mit einbezogen werden. Ausreichend sind jedoch Daten zu den einzelnen Beständen. Die Geodaten zu den Bestandesgeometrien werden zum einen für eine flächenscharfe Berechnung von Zielindikatoren benötigt, zum anderen für die Optimierungskomponenten (Kap. 4.6). Auf die Daten zum Wegenetz wird ebenfalls optional im Rahmen der Optimierung zurückgegriffen. Zusätzlich zu den Sach- und Geodaten benötigt das System für die einzelnen Modellkomponenten (Eingriffsmodell, Datenergänzung etc.) diverse Parameter. Diese sind in der sog. Modelldatenbank abgespeichert und können z. T. vom Nutzer individuell angepasst werden.

Grundsätzlich können zwei Datenstrukturen zum Aufbau der virtuellen Bestände eingelesen werden (Einzelbaumdaten vgl. Tab. 9 oder Bestandesdaten vgl. Tab. 10).

Tab. 9: Minimal benötigte Datenstruktur zum Erzeugen virtueller Bestände auf Basis von Einzelbaumerhebungen (Vollaufnahme, Stichprobe).

Variable	Datentyp	Beschreibung
STPN	Integer	eindeutige ID für jede Aufnahmeeinheit
ha	Double	Größe der Aufnahmeeinheit (ha)
Jahr	Integer	Aufnahmejahr
Besttyp	Integer	Bestandestyp (optional)
BNr	Text	eindeutige Bezeichnung des einzelnen Probebaums
Art	Integer	Art des Probebaums im Niedersachsencode (z. B.: 211=Buche)
Faktor	Double	Anzahl Bäume je Hektar, die der Probebaum repräsentiert
Alter	Integer	Alter des Probebaums
Azi	Double	Azimut des Probebaums in Gon oder Grad
Entf	Double	Entfernung zum Probekreismittelpunkt in Meter
BHD1	Double	Brusthöhendurchmesser in cm
BHD2	Double	zweiter BHD (Kreuzkluppung) sonst 0
Hoehe	Double	Baumhöhe in Meter (optional)
KrAns	Double	Kronenansatz (optional)
Anm	String	Beliebiger Text (max. 255 Zeichen)

Das Rohdaten-Interface verarbeitet Einzelbaum- und Bestandesdaten. Einzelbaumdaten beinhalten Informationen auf Baumebene (z. B. Alter, Art, BHD usw.). Einzelbaumdaten werden im Rahmen von Stichprobeninventuren oder Vollaufnahmen erhoben. Die erforderlichen Eingangsdaten bei Verwendung von Einzelbauminformationen sind in Tab. 9 aufgelistet. Bestandesdaten beinhalten für Schichten aggregierte Daten (z. B. Forsteinrichtungsdaten). Die Zugehörigkeit eines Baumes zu einer Schicht richtet sich nach Baumart und Alter oder Höhe. Da bei Forsteinrichtungsdaten Dimensionsangaben (dg, hg) meist aus Ertragstafeln stammen und vor allem der dg so meist unterschätzt wird (WOLLBORN u. BÖCKMANN 1998), kann optional eine automatische Durchmesserkorrektur vorgenommen werden. Dabei wird auf das von Wollborn und Böckmann im Kontext dieser Problematik vorgestellte Kalibrierungsmodell zurückgegriffen.

Die in Tab. 10 dargestellte Datenstruktur muss bei Verwendung von Forsteinrichtungsdaten vorhanden sein, um TreeGrOSS-konforme, virtuelle Bestände generieren zu können.

Tab. 10: Minimal benötigte Datenstruktur zum Erzeugen virtueller Bestände auf Basis von Forsteinrichtungsdaten.

Variable	Datentyp	Beschreibung
ID	Integer	eindeutige Bestandes-ID
Art	Integer	Baumart im Niedersachsencode
Alter	Integer	Alter der jeweiligen Baumart in der entsprechenden Schicht
dg	Double	Durchmesser des Grundflächenmittelstamms der jeweiligen Baumart in der entsprechenden Schicht
hg	Double	Höhe des Grundflächenmittelstamms der jeweiligen Baumart in der entsprechenden Schicht
rLK	Integer	relative Leistungsklasse
B°	Double	Bestockungsgrad
Schicht	Integer	Schicht
Schichtant.	Integer	prozentualer Anteil der jeweiligen Baumart in der jeweiligen Schicht
ha	Double	ideelle Fläche der jeweiligen Baumart in der Schicht [ha]
Best_ha	Double	reale Bestandesgröße [ha]
G	Double	Grundfläche der Baumart der jew. Schicht [m²]
d_max	Double	maximaler Durchmesser der jew. Art der Schicht
Jahr	Integer	Aufnahmejahr

Auf Basis dieser beiden Datenstrukturen werden einzelbaumbasierte Modellbestände generiert. Fehlende Daten (Einzelbaumdurchmesser, Stammfußkoordinaten) werden von

verschiedenen Routinen ergänzt. Die in Tab. 11 aufgelisteten Parameter beschreiben eindeutig einen Modellbestand und werden entsprechend der Unterscheidung nach Bestandes- oder Einzelbaumparameter in zwei Datenbanktabellen gespeichert.

Tab. 11: Datenstruktur zum Speichern der Modellbestände.

Variable	Datentyp	Beschreibung
Bestandesparameter		
STPN	Integer	eindeutige ID für jede Aufnahmeeinheit, Schlüsselfeld
Jahr	Integer	Aufnahmejahr
Endnutzungsdringlichkeit		s.o.
Pflegedringlichkeit		s.o.
y-Diversität		s.o.
h-index		s.o.
Besttyp	Integer	Bestandestyp (optional)
Einzelbaumparameter		
STPN		s. o.
BNr	Char	eindeutige Bezeichnung des einzelnen Modellbaums, Schlüsselfeld
Art	Integer	Art des Modellbaums im Niedersachsencode (z. B.: 211=Buche)
Faktor	Double	Anzahl Bäume je Hektar, die der Modellbaum repräsentiert
Alter	Integer	Alter des Modellbaums
X	Double	X-Koordinate im Bestand
Y	Double	Y-Koordinate im Bestand
BHD	Double	Brusthöhendurchmesser in cm
Vorrat	Double	Derbholzvorrat des Modellbaums
Hoehe	Double	Baumhöhe in Meter (optional)
KrAns	Double	Kronenansatz (optional)
Ausscheidejahr	Integer	Ausscheidejahr, sonst 0
Ausscheidegrund	Integer	1 = Mortalität, 2 = Pflege, 3 = Nutzung
Habitatbaum	Boolean	Als Habitatbaum ausgewählt.
Anmerkung	String	Beliebiger Text (max. 255 Zeichen)

Um die Anbindung des Entscheidungsunterstützungssystems an eine vorhandene Datenbank zu erleichtern, wurde eine flexible Datenbankschnittstelle implementiert. Diese bietet standardmäßig die Verbindung zu den Datenbanksystemen MySQL, PostgreSQL, Oracle und MS Access. Durch den objektorientierten und modularen Aufbau, lässt sich leicht die Unterstützung von weiteren Datenbanken realisieren. Das Datenbankmanagement ist mit

allgemeinen Datenbankabfragen zum Erzeugen/Löschen von Tabellen, Indizieren von Spalten sowie zum Abfragen/Manipulieren von Datensätzen ausgestattet. Die Abfragen sind in nativen SQL-Code gehalten und so theoretisch auf jede, den SQL-Standard unterstützenden Datenbank anwendbar. Realisiert wird die Datenbankschnittstelle über die Java Database Connectivity (JDBC). Die Java Database Connectivity der Java-Plattform ist ein einheitliches Interface zu Datenbanken verschiedener Hersteller und speziell auf relationale Datenbanken ausgerichtet (SUN 2008). JDBC ist somit eine universelle Datenbankschnittstelle vergleichbar mit ODBC (Open Database Connectivity) unter Windows-Systemen. Die grundlegenden Funktionen von JDBC sind Datenbankverbindungen aufzubauen und zu verwalten sowie über eine bestehende Verbindung SQL-Anfragen an die Datenbank zu senden und die Rückgabewerte (Ergebnisse und Systemmeldungen) in Java-Datentypen umzuwandeln und somit Programmen zur Verfügung zu stellen. Entsprechend sind für jede einzubindende Datenbank eigene Treiber erforderlich, welche die JDBC-Spezifikation berücksichtigen und die erforderlichen Methoden und Felder implementieren. Nach der JDBC-Spezifikation werden verschiedene Typen von Treibern unterschieden:

Typ-1-Treiber

Ein JDBC-Typ-1-Treiber kommuniziert ausschließlich über einen JDBC-ODBC-Bridge-Treiber und nicht direkt mit der Datenbank. Damit benötigt ein Typ-1-Treiber mindestens einen installierten ODBC-Treiber. Der JDBC-ODBC-Bridge-Treiber wandelt JDBC- in ODBC-Anfragen um. Die Anfrage an die Datenbank übernimmt nachfolgend der ODBC-Treiber. Ein Typ-1-Treiber sollte nur dann verwendet werden, wenn es keinen spezifischen JDBC-Treiber sondern nur einen ODBC-Treiber gibt.

Typ-2-Treiber

Ein Typ-2-Treiber kommuniziert über eine systemspezifische Programmbibliothek auf dem Rechner, welcher die Datenbankabfrage absetzt (Client) mit dem Datenbankserver. Das bedeutet, dass entsprechend dem Betriebssystem zusätzlich zu dem JDBC-Treiber eine Programmbibliothek benötigt wird.

Typ-3-Treiber

Mittels des Typ-3-Treibers werden die JDBC-API-Befehle in generische DBMS-Befehle übersetzt und (über ein Netzwerkprotokoll) an einen Middleware-Treiber auf einem Anwendungsserver übertragen. Erst dieser Anwendungsserver transformiert die Befehle für

die spezifischen Datenbankserver und leitet sie an diese weiter. Ein Typ-3-Treiber benötigt damit keine plattformspezifischen Bibliotheken und keine Informationen über den verwendeten Datenbankserver.

Typ-4-Treiber

Beim Typ-4-Treiber werden die JDBC-API-Befehle direkt in DBMS-Befehle des jeweiligen Datenbankservers übersetzt und (über ein Netzwerkprotokoll) an diesen übertragen. Ein Middleware-Treiber wird dabei nicht verwendet. Damit kann ein Typ-4-Treiber schneller als ein Typ-3-Treiber sein, ist aber weniger flexibel.

Für das hier vorgestellte System werden vorzugsweise Typ-4-Treiber verwendet, da diese eine schnelle Kommunikation zwischen Datenbankserver und dem Java-Programm ermöglichen und für den Anwender den Vorteil bieten, dass keine weiteren Programmbibliotheken oder eine Middleware installiert werden muss.

Das implementierte Datenbanksystem identifiziert Datensätze eindeutig über eine Bestandes-ID und die Einzelbaum-Datensätze zusätzlich über eine Baum-ID. Beim Erstellen von Datentabellen werden diese Spalten automatisch mit einem Index versehen, um schnelle Zugriffszeiten auch bei großen Datenmengen zu garantieren. Ein Datenbankindex, ist eine von den Daten getrennte Struktur, welche die Suche und das Sortieren nach bestimmten Feldern (Spalten) beschleunigt. Ein Index besteht aus einer Menge von Pointern (Zeiger, Verweise), die eine Ordnungsrelation auf eine oder mehrere Spalten in einer Tabelle definieren. Wird bei einer Abfrage auf eine indizierte Relation ein indiziertes Feld als Suchkriterium verwendet, sucht das Datenbankmanagementsystem die der Abfrage entsprechenden Datensätze anhand der Zeiger. In der Regel kommen hier Baum-Strukturen (z. B. B^+-Bäume) zum Einsatz. Bei Abfragen auf Relationen ohne Index müssen die Spalten sequentiell durchsucht werden, was selbst auf schnellen Systemen viel Zeit beanspruchen kann.

4.4.7 Modellmanagement - Parallelisierte Simulation

Das Modellmanagement übernimmt im Kontext eines DSS die Aufgabe der Steuerung und Kopplung der einzelnen Teilmodelle und die Kommunikation mit dem Datenmanagement. In dem vorgestellten System steuert das Modellmanagement die einzelnen Teilmodelle so, dass jeweils ein vollständiges Prognoseintervall zu einem ausgewählten Startzustand berechnet wird. Ein Prognoseintervall umfasst die Modellierung des Wachstums der Einzelbäume, der natürlichen Mortalität bzw. Totholzentwicklung, des Einwuchs und die Simulation

waldbaulicher Maßnahmen (Pflege, Nutzung, Schutzmaßnahmen). In einem Simulationsintervall werden alle Bestände des Betriebs für einen definierten Zeitraum fortgeschrieben. Die Simulationsdauer eines Intervalls kann ein bis dreißig Jahre betragen. Nach jedem Simulationsintervall wird der prognostizierte Zustand in der angebundenen Datenbank abgespeichert. Dieser kann anschließend ausgewertet werden oder als Ausgangszustand für die Simulation eines weiteren Intervalls genutzt werden. Zur Durchführung der Simulation werden drei mögliche Verfahren implementiert:

- Einzelplatz (single-/multi thread)
- Serverseitig (single-/multi thread)
- Verteilt auf mehrere Server (single-/multi thread)

Bei der Einzelplatzlösung läuft das gesamte System bestehend aus GUI und dem Simulationsserver physisch auf nur einem Rechner und auch nur in einer Java Virtual Machine. Die Steuerung des Servers übernimmt das Programm, ohne dass der Nutzer mit der Installation und Parametrisierung des Simulationsservers konfrontiert wird. Bei der serverseitigen Simulation wird der Simulationsserver auf einem gesonderten Rechner oder einem Cluster, bestehend aus mehreren Maschinen, installiert. Der Client sendet dann lediglich die Spezifikation der zu simulierenden Variante an den Server und erhält Informationen über den Simulationsfortschritt vom Server zurück. Die Informationen zwischen Client und Server werden über serialisierte Objekte mittels des entsprechenden Netzwerkprotokolls ausgetauscht.

In allen Varianten kann die Simulation auf mehrere Threads aufgeteilt werden. Ein Thread bezeichnet einen Ausführungsstrang in der Abarbeitung eines Programms. Ein Thread ist Teil eines Prozesses, welchem eigener Speicherraum und weitere Betriebssystemmittel zugeordnet werden. Deswegen ist der Verwaltungsaufwand für Threads üblicherweise geringer als der für Prozesse. Mittels der Threadtechnologie ist es möglich, Rechenvorgänge parallel ablaufen zu lassen (OAKS u. WONG 2004). Gerade auf Systemen mit mehr als einer CPU oder mit einer CPU mit mehreren Kernen kann in Kombination mit einer geeigneten Programmierung, die mehrere Threads startet, um eine Aufgabe zu lösen, die Abarbeitungsgeschwindigkeit deutlich erhöht werden. Voraussetzung hierzu ist jedoch, dass das zu berechnende Problem parallelisierbar ist. Es muss in mehrere Teilberechnungen zerlegbar sein, die unabhängig voneinander ablaufen können. Bei der Zuwachsprognose und

der virtuellen Bestandesbehandlung auf Basis eines einzelbaumorientierten Wuchsmodells ist diese Vorraussetzung erfüllt. Die einzelnen Planungseinheiten (Modellbestände) können unabhängig voneinander fortgeschrieben werden.

Das vorgestellte System wurde so implementiert, dass mehrere Bestände parallel in den Arbeitsspeicher des Einzelplatzrechners oder einer Servermaschine geladen und simuliert werden können. Dabei kann der Server auch Aufgaben mehrerer Clients annehmen und parallel verarbeiten (Abb. 37).

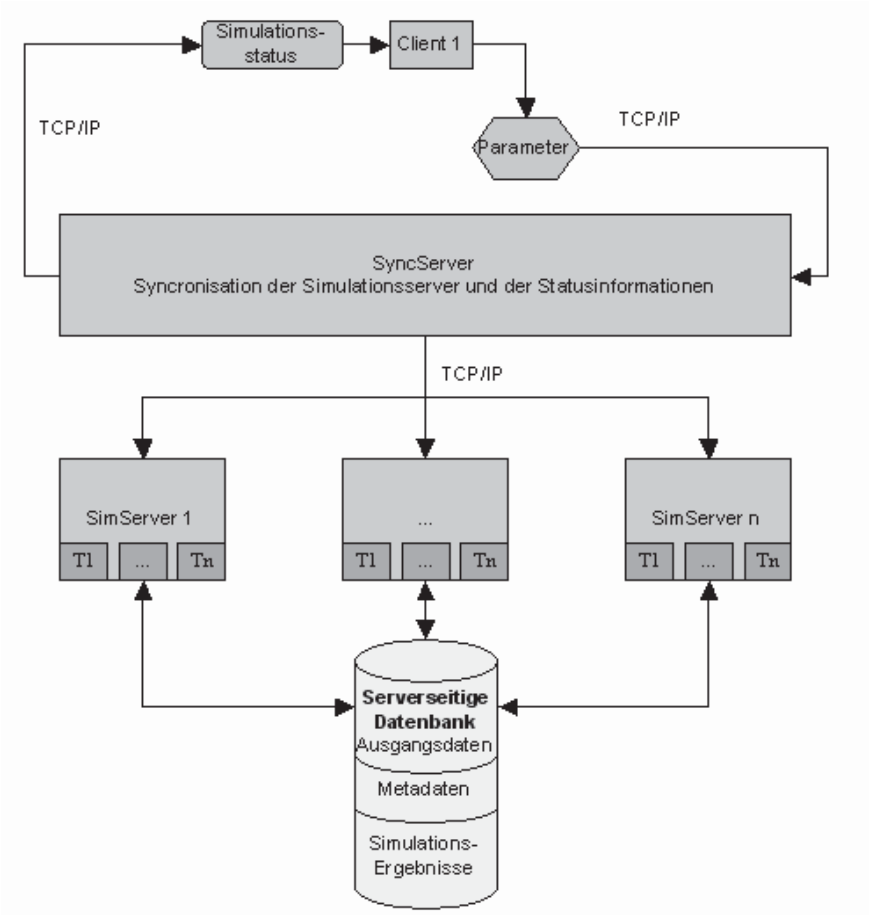


Abb. 37: Schematische Darstellung der aufgeteilten Simulation. Der Synchronisationsserver teilt die Gesamtanzahl der zu simulierenden Bestände auf mehrere Simulationsserver auf und verwaltet den Simulationsablauf.

Die Ergebnisse für eine festgelegte Anzahl von Beständen werden in einem arbeitsspeicherseitigen Batchcontainer gesammelt und nach Erreichen einer definierten Anzahl an simulierten Beständen in die Datenbank geschrieben. Der Container wurde mit einem doppelten Speicher versehen, so dass nach Erreichen der Speicherkapazität die simulierten Bestände in den zweiten Speicher geschrieben werden können. Der gefüllte Speicher wird dann simultan in die Datenbank geschrieben und geleert. Ist der zweite

Speicher vollständig gefüllt, wird wieder der erste Speicher gefüllt. Aus dieser Double-Buffer-Strategie resultieren zwei Vorteile. Zum einen wird bei einer ausgewogenen Dimensionierung der Speicherkapazität des Buffers in Abhängigkeit der Rechengeschwindigkeit die Wartezeit auf Schreibvorgänge in die Datenbank minimiert. Zum anderen wird die Geschwindigkeit der Einfügeabfrage ebenfalls verbessert, da ein größerer Datenblock durch den Batchvorgang mit einer einzelnen Datenbankabfrage übertragen wird und so das Verhältnis von eigentlichen Daten und Abfragemetadaten verbessert wird. Dies wirkt sich wiederum positiv auf die Übertragungsgeschwindigkeit der Daten in die Datenbank aus. Die Anzahl der einzelnen Simulations-Threads und die Größe des Zwischenspeichers sind innerhalb gegebener Schranken frei wählbar, werden aber im Rahmen der clientseitigen Simulation dem Anwender auf Basis des verfügbaren Arbeitsspeichers und der Anzahl der Prozessoren/Kerne vorgeschlagen. Die Server-Applikation ist über einen konsolenbasierten Client steuerbar oder wird ohne zusätzliche Nutzerinteraktion von dem Einzelplatzprogramm aufgerufen und gesteuert.

Neben der Parallelsimulation auf einer einzelnen Maschine unterstützt das System auch die Aufteilung auf zwei oder mehr Servereinheiten, die auf verschiedenen Rechnern laufen. Dabei kann die Aufteilung der Prognose aller Bestände mit der Rechenleistung der einzelnen Maschinen gewichtet werden. Sollen beispielsweise 600 Bestände fortgeschrieben werden und es stehen ein Rechner mit 4 Prozessoren (A) und ein anderer mit 2 Prozessoren (B) zur Verfügung, kann die Anzahl der jeweils zu verarbeitenden Bestände über das Prozessorverhältnis bestimmt werden. Rechner A bekommt 400 Bestände zugewiesen, Rechner B 200. Abb. 38 zeigt die Kommunikation bei einer serverseitigen Simulation mit einem Server und n Clients. Die Clients und der Server kommunizieren über das bestehende Netzwerkprotokoll (z. B. TCP/IP[10]). Der Client sendet eine Simulationsanfrage in Form eines serialisierbaren Objekts an den Server, welcher an einem einstellbaren Port auf Anfragen „lauscht“. Dabei werden alle nötigen Parameter (Simulationsdauer, Ziel-/Ausgangsdatenbank, Modellspezifikationen usw.) an den Server übermittelt.

[10] Das Transmission Control Protocol (TCP) zerlegt zu versendende Objekte/Nachrichten in Datenpakete und setzt diese beim Empfänger wieder zusammen. Das Internet Protocol (IP) sorgt für ein korrektes Routen des Datenstroms durch das Netz.

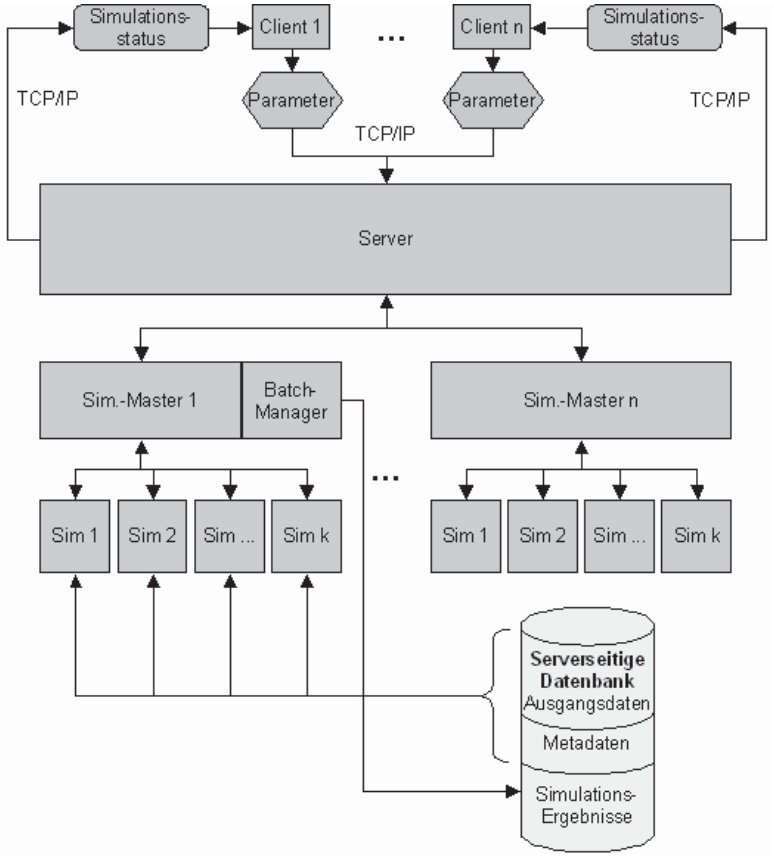


Abb. 38: System der Server-Client-Kommunikation und des parallelen Simulierens.

Akzeptiert der Server die Verbindung, wird dem Simulationsprozess eine eindeutige ID zugewiesen, so dass nach Beenden der Verbindung bei laufender Simulation bei einer erneuten Verbindung über diese ID die Kommunikation zur entsprechenden Simulation wieder aufgenommen werden kann. Hierzu wird ein Master-Thread gestartet, der die Kommunikation über den Server mit den Clients und die Koordination der einzelnen, parallel laufenden Simulations-Threads übernimmt. Es werden der Simulationsfortschritt und aufgetretene Fehler über eigens definierte Message-Objekte an den Client weitergeleitet, der diese aufbereitet und an die GUI sendet. Die Koordinationsaufgaben des Master-Threads belaufen sich auf das Initialisieren der spezifizierten Anzahl an Simulations-Threads sowie das Bearbeiten von Anfragen oder das Versenden von Meldungen zur laufenden Simulation. Das Speichern der einzelnen Simulationsergebnisse wird ebenfalls von dem Master-Thread übernommen. Je nach Hardwareausstattung führt diese Technologie zu einer Verbesserung der Simulationsgeschwindigkeit. Abb. 39 zeigt die Simulationsdauer in Abhängigkeit der verwendeten Threadanzahl auf drei Systemen mit den in Tab. 12 dargestellten Hardware- und Softwarekonfigurationen.

Tab. 12: Ausstattung der Rechner, welche zum Test des Zusammenhangs von Rechenzeit und verwendeter Anzahl von Simulations-Threads verwendet wurden.

System Ausstattung	UP_HT	MP2	MP4
Betriebssystem	Windows XP Professional	OpenSuSE 10.2 (x86_64)	Windows XP Professional
CPU(s):	1x Intel Pentium 4 mit 3,0 Ghz	2 x Intel Core2 Duo 6700 mit 2,66 Ghz	2 x Intel Xeon Quad E5420 mit 2,5 Ghz
Arbeitsspeicher	1 GB	4 GB	4 GB
Festplatte	80 GB System und Daten	80 GB Systemplatte, 1 TB für die Daten (2x 1 TB als Raid 1)	80 GB System, 500 GB Daten

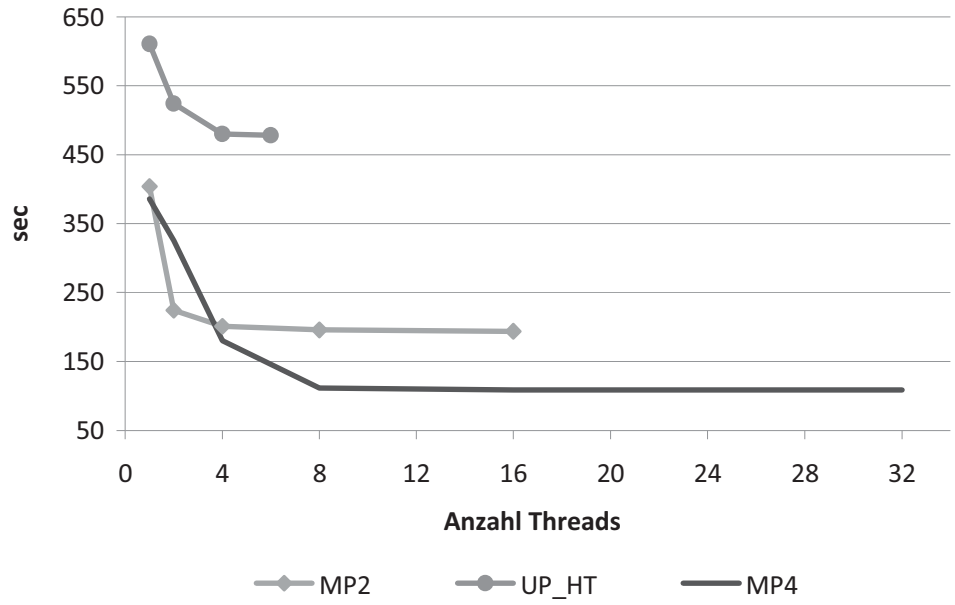


Abb. 39: Simulationsdauer für 2000 Bestände mit einer Größe von 0.25ha in Abhängigkeit der Threadanzahl auf drei PC-Systemen. MP2= Multiprozessor mit 2x Intel Core2 Duo, UP_HT Uniprozessor mit Intel Pentium (mit Hyperthreading) und MP4= Multiprozessor mit 2x Intel Xeon Quad E5420.

Insgesamt werden 2000 Modellbestände mit einer Größe von 0,25 ha fünf Jahre fortgeschrieben (Mortalität, Zuwachs, Einwuchs) und anschließend einem Nutzungsalgorithmus unterzogen. Es ist deutlich zu erkennen, dass bis zur Auslastung aller Kerne der beiden Prozessoren der Multiprozessorsysteme MP2 und MP4 eine Beschleunigung des Simulationsprozesses erreicht werden kann (Abb. 39). Bei dem Simulationslauf mit lediglich einem aktiven Simulations-Thread wird eine Rechenzeit von über 400 (MP2) bzw. 385 (MP4) Sekunden benötigt. Bei vier aktiven Simulations-Threads liegt die Rechenzeit nur noch bei ca. 200 bzw. 180 Sekunden. Beim Sprung von 2 auf 4 Simulations-Threads ist die Verbesserung der Simulationsgeschwindigkeit im Vergleich zum Sprung auf 1 bzw. 2

Simulations-Threads auf dem System MP2 wesentlich geringer. Dies liegt daran, dass die Steuer-Threads (Master-Thread, Serverkommunikation) und das Datenbanksystem ebenfalls Rechenzeit benötigen und unter Volllast des Systems die zur Verfügung stehende Rechenzeit für die eigentlichen Simulations-Threads einschränken. Das System MP4 besitzt insgesamt acht Prozessorkerne (2x4). Auf diesem System ist eine Steigerung der Simulationsgeschwindigkeit bis zur Verwendung von acht Threads erkennbar. Eine weitere Erhöhung der Simulationsthreadanzahl (über die Anzahl der verfügbaren Kerne hinaus) führt auf beiden Multiprozessorsystemen zu keiner weiteren Steigerung der Simulationsgeschwindigkeit. Auch auf dem Uniprozessor-System (UP_HT) kann durch Erhöhung der Threadanzahl die Simulationsgeschwindigkeit gesteigert werden. Jedoch ist der Effekt nicht so deutlich wie auf den Mehrkern- bzw. Multiprozessor-Systemen. Hier bewirkt vor allem die Hyper-Threading-Technologie, dass das System bei Durchführung von Rechenoperationen gleichzeitig Lese- bzw. Schreibvorgänge durchführen kann und die Prozessorkapazitäten insgesamt besser ausgenutzt werden können, da zwei virtuelle Prozessoren sich die Ressourcen des physischen Prozessors teilen.

4.5 Entscheidungsrelevante Indikatoren

Zur Entscheidungsunterstützung müssen der aktuelle Waldzustand (generiert aus Inputdaten) und mögliche Waldentwicklungsszenarien (generiert durch Simulation) bewertet werden können. Um hierbei möglichst alle Waldfunktionen zu berücksichtigen, müssen verschiedenen Indikatoren verfügbar sein, welche auf Betriebs- und Bestandesebene Information zu den entsprechenden Bereichen beinhalten. In Tab. 13 wird ein Überblick über die wichtigsten, in dem vorgestellten System verfügbaren Indikatoren gegeben.

Die meisten Indikatoren können direkt aus den Einzelbaumattributen oder aus den durch die Modellkomponenten verfügbaren Parametern abgeleitet werden. Der Großteil beschreibt die forstlichen Ressourcen (Vorrat, Grundfläche, dg, hg etc.). Es werden aber auch ökonomische und ökologische sowie raumbezogene Indikatoren berechnet. Auf speziell für das vorgestellte Entscheidungsunterstützungssystem entwickelte Indikatoren wird in den anschließenden Kapiteln näher eingegangen. Meist können die Indikatoren sowohl für den Bestand als auch für den gesamten Betrieb oder verschiedenen Straten (z. B. Revier) berechnet werden.

Tab. 13: Für die Entscheidungsunterstützung verfügbare Indikatoren.

Indikator	**Beschreibung**	**Ebene**
Vorrat [Vfm/ha oder Vfm]	Gesamter oberirdischer Derbholzvorrat, absolut oder je Hektar, über alle Baumarten oder nach Baumart getrennt	Bestand, Betrieb
Grundfläche [m²/ha]	Grundfläche über alle Arten oder getrennt nach Baumart	Bestand, Betrieb
Höhenbonität [m]	Höhe im Alter 100 der jew. Baumart und/oder Schicht	Bestand
DG [cm]	BHD des Grundflächenmittelstamms je Baumart und/oder Schicht	Bestand
HG [m]	Höhe des Grundflächenmittelstamms je Baumart und/oder Schicht	Bestand
Stammzahl	Stammzahl gesamt, je Art und/oder Schicht	Bestand
Zuwachs [Vfm ha^{-1} a^{-1}]	Jährlicher Derbholzuwachs der letzten Wachstumsperiode je Hektar, der Zuwachs wird aus dem Volumenzuwachs des verbleibenden Bestands und den Abgängen (Nutzung, Mortalität) sowie dem Einwuchs berechnet, gesamt oder je Art	Bestand, Betrieb
Natürliche Mortalität [Vfm ha^{-1} a^{-1}]	Mittlerer Derbholzvorrat der in der letzten Wuchsperiode abgestorbenen Bäume, gesamt oder je Art	Bestand, Betrieb
Bestandestyp (BT)	Der Bestandestyp wird aus der jew. Baumartenzusammensetzung abgeleitet (NIEDERSÄCHSISCHE LANDESFORSTEN 1987)	Bestand
Durchmesser-Verteilung	Vorrat über Durchmesserstufen (5 cm) gesamt oder je Art	Bestand, Betrieb
Altersklassen-Verteilung	Vorrat oder Fläche nach Altersklassen, gesamt oder je Art	Bestand, Betrieb
Baumartenanteile [%]	Prozentuale Baumartenanteile (Vorrat)	Bestand, Betrieb
Vornutzungsmasse [Efm/ha]	Vornutzungsmassen je Eingriff und Hektar, gesamt oder je Art	Bestand, Betrieb
Endnutzungsmasse [Efm/ha]	Endnutzungsmassen je Eingriff und Hektar, gesamt oder je Art	Bestand, Betrieb
Sortenstruktur	In Abhängigkeit vom Nutzer vorgegebener Aushaltungsszenarien kann der Ausscheidende Bestand sortiert werden	Bestand, Betrieb
Pflegedringlichkeit	Kap. 4.5.4.1	Bestand, Betrieb
Endnutzungs-dringlichkeit	Kap. 4.5.4.2	Bestand, Betrieb
Erlöse (ekf) [€]	Erlös aus Vor- oder Endnutzung abzüglich der Erntekosten (vgl. DUDA 2006)	Bestand, Betrieb
Abtriebswert (ekf) [€]	Erntekostenfreier Abtriebswert (vgl. DUDA 2006)	Bestand, Betrieb
Ökonomischer Erfolg	Kap. 4.5.3	Bestand, Betrieb
Konstante Nutzung	Kap.4.5.1	Bestand, Betrieb
Habitatbäume [n/ha] [Vfm/ha]	Anzahl bzw. Vorrat der als Habitatbaum markierten und geschützten Bäume	Bestand, Betrieb
Totholzvorrat stehend [m²/ha]	Volumen des stehenden Totholzes, gesamt oder je Art	Bestand, Betrieb
Totholzvorrat liegend [m²/ha]	Volumen des liegenden Totholzes, gesamt oder je Art, mit oder ohne Stubben	Bestand, Betrieb
A-Index	Beruht auf den Shannon-Index und beschreibt ein vertikales Art-Höhen-Profil. Für einschichtige Reinbestände liefert der Index niedrige Werte, für artenreiche mehrschichtige Mischbestände resultieren die höchsten Werte.	Bestand
Bestandestypen (BT)-Durchmischung	Kap. 4.5.2	Betrieb
Baumartendiversität	Auf Basis des Shannon-Index (SHANNON 1948) berechnete Baumartendiversität	Betrieb
Windwurfrisiko	Anteil der durch Sturm geworfenen Bäume	Bestand, Betrieb

4.5.1 Konstante Nutzung

Der Indikator *konstante Holzlieferung (ef)* kommt dann zum Tragen, wenn eine durchgeführte Simulation mehrere Eingriffsintervalle (n) umfasst. Über diesen Indikator wird beschrieben, wie gleichmäßig sich Nutzungsmassen (simuliert oder real) über den betrachteten (Simulations-) Zeitraum verteilen. Zunächst wird der Verteilungsindikator *d* über die Summe der Beträge der Differenzen aus dem Verhältnis der Erntemassen in den einzelnen Perioden (meist. 1 Jahr) zum Gesamteinschlag (p_i) und dem Anteil bei völliger Gleichverteilung (1/n) berechnet (Gl. 9). Die Normierung auf das Intervall [0 1] wird durch Division von d durch d_{max} erreicht. d_{max} ist der maximal mögliche Wert, den *d* annehmen kann. Er berechnet sich wie *d*, wobei angenommen wird, dass eine maximale Ungleichverteilung vorliegt, wenn die gesamte Nutzungsmasse nur in einer der n Perioden anfällt. Der Wert für *d* errechnet sich in diesem Fall wie folgt:

$d= 1-(1/n) + ((1/n)(n-1)) = 1+((n-2)/n)$

Konzentriert sich beispielsweise die Holznutzung bei insgesamt zwei möglichen Eingriffsintervallen nur auf das erste Intervall, nimmt *ef* einen Wert nahe 0 an. Je gleichmäßiger sich die Erntemassen auf die Intervalle verteilen, umso größer wird der *ef*. Bei völliger Gleichverteilung nimmt der Indikator den Wert 1 an.

$$ef = \begin{cases} 1-\dfrac{d}{d_{\max}} & n>1 \\ 1 & n=1 \end{cases} \quad \text{mit} \quad d=\sum_{i=1}^{n}\left|\frac{1}{n}-p_i\right| \qquad d_{\max}=1+\left(\frac{n-2}{n}\right)$$

Gl. 10

wobei:
n = Anzahl Simulationsperioden
p_i = Nutzungsmassenanteil der Periode i an der Summe aller Nutzungsmassen

Die Berechnung von *ef* ist an die Evenness (E_s) von Simpson (SIMPSON 1949) angelehnt. Diese berechnet sich aus der Summe der Quadrate der einzelnen Anteile aller beobachteten Arten (in diesem Fall der Nutzungsmassen in den Perioden 1 bis n) und dem bei n Arten bzw. Perioden maximal möglichen Wert (Gl. 11). Die Evenness (E_p) nach Pielou (PIELOU 1966) berechnet sich aus dem Verhältnis des Shannon-Index für die Nutzungsmassen zu dem bei p Perioden maximal möglichen Shannon-Index (*ln(p)*) (Gl. 11). Vergleicht man die berechneten Indizes E_p, E_s und *ef* für verschiedene Nutzungssituationen, zeigt sich, dass die Evenness (sowohl E_p als auch E_s) für die gleiche Verteilungssituation der Nutzungsmassen auf die einzelnen Intervalle insgesamt höhere Werte aufweist (mit Ausnahme der oberen und unteren Grenzen 0 und 1).

$$E_p = \frac{-\sum_{i=1}^{p} e_i}{\ln(p)} \quad \text{mit} \quad e_i = \begin{cases} cr_i \ln(cr_i) & cr_i > 0 \\ o & cr_i = 0 \end{cases}$$

wobei: *Gl. 11*

p = Anzahl Simulationsperioden

cr_i = relative Nutzungsmassen in der Periode i (Nutzungsmasse der Periode i / Gesamtnutzungsmasse)

$$E_s = \frac{1 - \sum_{i=1} cr_i^2}{1 - \frac{1}{n}}$$

Weiterhin ist zu beobachten, dass die Evenness schneller ansteigt als der Index *ef*. Es werden also Situationen geringerer Gleichverteilung verglichen mit Situationen größerer Gleichverteilung höher bewertet als es durch den Index *ef* geschieht. Dieser „bestraft" stärkere Ungleichverteilungen somit stärker. Dieser Sachverhalt ist in Abb. 40 dargestellt. Es sind die Werte der Indikatoren E_p, E_s und *ef* für verschiedene Nutzungsmassenverteilungssituationen in aufsteigender Reihenfolge aufgetragen.

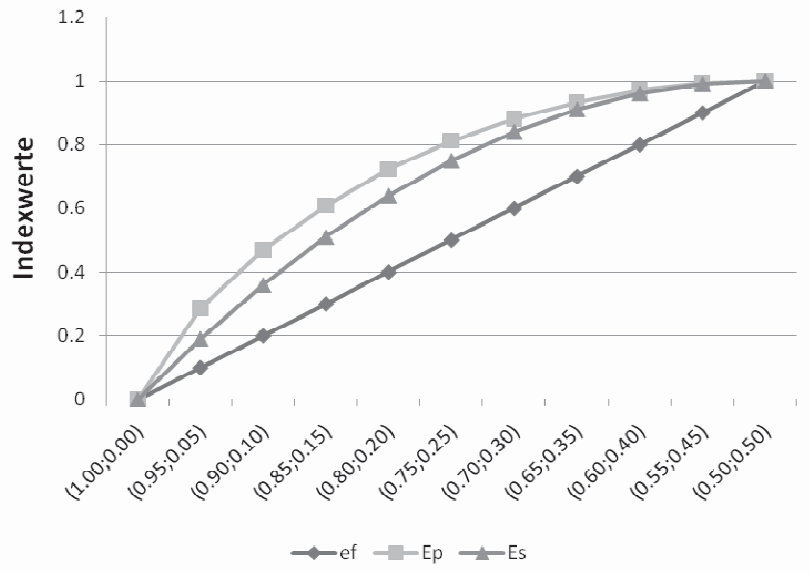


Abb. 40: Indexwerte E_p, E_s *und ef für Nutzungsmassenverteilungen (vereinfacht für zwei Nutzungsintervalle) mit zunehmender Gleichverteilung.*

Es werden zwei Nutzungsperioden unterstellt. Auf der X-Achse sind die Anteile am Gesamteinschlag in den beiden Perioden aufgetragen. Für die Situation (1; 0) liefern alle Indikatoren einen Wert von 0. Es liegt eine völlige Ungleichverteilung vor. Zu der Situation (0,5; 0,5) berechnen die drei Methoden erwartungsgemäß einen Wert von 1 (völlige Gleichverteilung).

4.5.2 BT-Durchmischung

Über die Baumartenanteile oder den Baumarten-Shannon-Index eines Betriebs können Aussagen über die Artenvielfalt getroffen werden. Der Index *Bestandestypendurchmischung* hingegen sagt etwas über die räumliche Verteilung und somit die landschaftliche Vielfalt oder den Abwechslungsreichtum der Landschaft aus. Der hier vorgestellte Ansatz basiert auf der Berechnung des Mischungs-Index nach Füldner (FÜLDNER 1995). Dieser Index wird zur Quantifizierung der räumlichen Baumartenmischung in einzelnen Beständen verwendet. Es wird für jeden Baum der Anteil seiner n nächsten Nachbarn mit derselben Art des Bezugsbaums berechnet. Aus den einzelnen Anteilen wird ein Index für den gesamten Bestand aggregiert. Ein ähnliches Vorgehen wird in dem vorgestellten System zur Berechnung der Bestandestypendurchmischung verwendet. Die vektorbasierte (Shapefile) Bestandestypenkarte (vgl. Abb. 41 oben links) wird von einer speziellen Klasse des GIS-Moduls (RasterTools, Kap. 4.4.5) in Rasterzellen (50x50 m) umgewandelt (vgl. Abb. 41 oben rechts). Anschließend wird für jede Rasterzelle geprüft, wie viele der direkt angrenzenden acht Rasterzellen einen anderen BT aufweisen (vgl. Abb. 41 unten links). Geostatistische Verfahren arbeiten oft mit einem sog. „moving window“, welches über die direkten Nachbarn hinaus noch weitere Rasterzellen abstandsgewichtet berücksichtigt (LEITÃO et al. 2006). Der Mittelwert über alle Rasterzellen des Untersuchungsgebiets bzw. des Betriebs kann als sog. Durchmischungskoeffizient bezeichnet werden und charakterisiert die landschaftliche Vielfalt anhand des Abwechslungsreichtums der Bestandestypen. Um die Interpretation und die Vergleichbarkeit dieses Koeffizienten zu verbessern, wird der Wert durch die maximal mögliche Durchmischung geteilt (Gl. 12). Diese wird ähnlich wie der Durchmischungskoeffizient berechnet. Der einzige Unterschied besteht darin, dass jedem Bestand nicht der tatsächliche Bestandestyp zugewiesen wird, sondern eine nur einmal vergebene Ziffer. Dadurch ist sichergestellt, dass alle Nachbarn eines Bestandes einen anderen, fiktiven Bestandestyp aufweisen. Die Berechnung des Durchmischungskoeffizienten auf Basis der fiktiven Bestandestypen ergibt den maximal möglichen Durchmischungswert. In Abb. 41 (unten links) ist zu erkennen, dass die hell eingefärbten Rasterzellen, welche eine hohe Durchmischung aufweisen, im Bereich der Bestandesgrenzen liegen. Dies ist damit zu begründen, dass alle Nachbarn eines Bestands einen anderen (fiktiven) BT aufweisen. Durch Normieren mit der maximal möglichen Durchmischung wird der einfache Durchmischungskoeffizient direkt vergleichbar und interpretierbar. Ein Wert von 1 bedeutet, dass die maximal mögliche Durchmischung vorliegt. Bei der Interpretation ist jedoch zu

beachten, dass ein hoher Durchmischungswert nicht zwangsläufig auch auf eine große Anzahl verschiedener Bestandestypen schließen lässt. Die Durchmischung in einem Betrieb mit nur drei verschiedenen Bestandestypen kann beispielsweise größer als in einem Betrieb mit sechs verschiedenen Bestandestypen sein. Ausschlaggebend ist die räumliche Verteilung der Bestandestypen.

Vektorbasierte Bestandestypenkarte

Rasterbasierte Bestandestypenkarte

Rasterbasierte Anteile gleicher Nachbarzellen (Bestandestyp)

Rasterbasierte Anteile gleicher Nachbarzellen (fiktiver BT)

Abb. 41: Darstellung der einzelnen Schritte zur Berechnung des normierten Durchmischungskoeffizienten. In den beiden unteren Abbildungen deuten dunkle Schattierungen auf einen geringen Anteil von Nachbarzellen mit anderem Bestandestyp hin. Je heller die Schattierung wird, desto mehr benachbarte Zellen weisen einen anderen BT auf.

$$dn = \frac{\frac{1}{n}\sum_{i=1}^{n} bt_i}{\frac{1}{n}\sum_{i=1}^{n} \max_i}$$

Gl. 12

wobei:
dn = normierte Durchmischungskoeffizient
n = Anzahl Rasterzellen
bt_i = Anteil der benachbarten Rasterzellen mit unterschiedlichem BT
max_i = Anteil der benachbarten Rasterzellen mit unterschiedlichem fiktiven BT (→ maximal möglicher Anteil)

4.5.3 Ökonomischer Erfolg

Zur Bewertung des ökonomischen Erfolgs (*oet*) eines Szenarios oder einer Variantenkombination im Rahmen eines Optimierungslaufes werden der Abtriebswert (*A*) und die Summe der verzinsten Erlöse aus Eingriffen der jeweiligen Simulationsperiode (*e*) gleich gewichtet verknüpft. Durch Kombination des Abtriebswertes und der Erlöse wird sowohl die Vermögenssphäre als auch die Finanzsphäre der Variantenkombination (des waldbaulichen Handelns) berücksichtigt (Gl. 13). Sowohl der Abtriebswert als auch die Erlöse werden unter Berücksichtigung der Erntekosten berechnet. Zur Berechnung der Erlöse kommen baumartenspezifische, vom BHD und dem Z-Baum-Status abhängige Bewertungsfunktionen zum Einsatz (DUDA 2006). Da die Modellbäume keine Qualität zugeordnet bekommen, wird verallgemeinernd davon ausgegangen, dass die ausgewählten Z-Bäume durch einen besseren Pflegezustand eine bessere Qualität und somit höhere Erlöse erreichen.

$$oet = A_{t_1} + \sum_{i=1}^{n} e_i (1+p)^{n-i} \qquad \textit{Gl. 13}$$

wobei:
oet = ökonomischer Erfolg
A_{t1} = erntekostenfreier Abtriebswert zum Zeitpunkt t_1
e_i = erntekostenfreie Erlöse zum Simulationsjahr i
p = Zinssatz/100
n = Simulationsdauer in Jahren

Ein weiterer geeigneter Indikator wäre der Holzproduktionswert (MÖHRING et al. 2006). Dieser Wert berücksichtigt in einer annuisierten Darstellung ebenfalls beide Sphären. Die Vermögenssphäre wird über die diskontierte Veränderung des Abtriebswertes (t_0 bis t_1) abgebildet. Da bei dem vorliegenden Optimierungsproblem für alle Variantenkombination der Ausgangsabtriebswert (t_0) identisch ist, wird auf die in Gl. 13 beschriebene Berechnung des ökonomischen Erfolgs zurückgegriffen.

4.5.4 Handlungsdringlichkeit

Dieser Index dient der Bestimmung der Dringlichkeit, in einem Bestand eine Pflegemaßnahme oder Endnutzung durchzuführen. Wichtig dabei ist die Relation der ermittelten Werte zueinander und nicht die absolute Aussagekraft. Dieser Index wird im Rahmen der kurzfristigen und mittelfristigen Optimierung eingesetzt, wobei unter anderem eine Unterauswahl an möglichst pflege- bzw. nutzungsdringlichen Beständen getroffen werden soll, respektive die ermittelte Nutzungsstrategie mittelfristig zu einer geringen Handlungsdringlichkeit führen soll. Die beiden Komponenten *Endnutzungsdringlichkeit* und *Pflegedringlichkeit* werden auch bei einer nutzergesteuerten Szenariosimulation berechnet

und gespeichert, so dass diese im Rahmen eines Variantenstudiums zur Verfügung stehen. Die Bestimmung und Quantifizierung der Eingriffsdringlichkeit für einzelne Bestände ist ein wichtiger Indikator im Rahmen der Entscheidungsunterstützung. Verbleiben Bestände zu lange in einem ungepflegten Zustand, ist mit Einbußen der zu erwartenden Qualität und Masse des Endnutzungsbestands und sinkender Bestandesstabilität zu rechnen. Ledermann u. Neumann (LEDERMANN u. NEUMANN 2009) bestimmen die Eingriffsdringlichkeit für Vornutzungsbestände über die Bestandesdichte und die Endnutzungsdringlichkeit mittels geschätzter Wertzuwächse. Je geringer der Wertzuwachs ausfällt, umso dringlicher ist die Durchführung der Endnutzung eines Bestandes.

4.5.4.1 Pflegedringlichkeit

Die Pflegedringlichkeit drückt die besondere Notwendigkeit zur Durchführung von Pflegemaßnahmen in Vornutzungsbeständen aus. Pflegemaßnahmen verfolgen vor allem das Ziel, den Wertzuwachs zu fördern, die Bestandesstabilität zu sichern und die Baumartenmischung zu regulieren (DENGLER 1992). Oft wird die Pflegedringlichkeit einer Behandlungseinheit am Bestockungsgrad oder gar am Zeitraum bis zur letzten durchgeführten Maßnahme festgemacht. Dies sind Indikatoren, die nur zum Teil oder keine objektive Einschätzung der tatsächlichen Pflegedringlichkeit zulassen.

Zur quantitativen Bestimmung der Pflegedringlichkeit eines Modellbestandes werden, um dem Ziel der Vornutzung gerecht zu werden, folgende Indikatoren berücksichtigt:

- Bedrängung der Z-Bäume
- Zielerreichungsprozent nach Duda (DUDA 2006)
- Bestandesdichte

Die Bedrängung der Z-Bäume wird über die auf der Arbeit von Döbbeler (Döbbeler 2004) basierende maximale Dichte hergeleitet. Für jeden Z-Baum eines modellhaft abgebildeten Vornutzungsbestands wird der Konkurrenzdruck (C_{66}) hergeleitet. Weiterhin wird für jeden Z-Baum die maximal erträgliche Konkurrenz berechnet (C_{66max}). Diese wird aus der maximalen Stammzahl eines Bestandes abgeleitet, dessen *hg* und *dg* dem BHD und der Höhe des jeweiligen Z-Baums entsprechen. Das Verhältnis von tatsächlichem und maximalem Konkurrenzdruck beschreibt die Dringlichkeit, den jeweiligen Z-Baum freizustellen. Je größer das Verhältnis wird, umso stärker ist der Konkurrenzdruck auf den Bezugsbaum. Das so berechnete Verhältnis kann minimal den Wert 0 annehmen, welcher gleichzeitig den

kleinstmöglichen Konkurrenzdruck beschreibt. Werte über 1 sind eher unwahrscheinlich, da ab dieser Konkurrenzsituation der Baum modellbedingt abstirbt. Das Mittel der Einzelbaumwerte ergibt einen Indikator zur Beschreibung der Bedrängungssituation der Z-Bäume auf Bestandesebene (vgl. Gl. 14).

$$pd_i = \begin{cases} 0.6\left(\frac{1}{n}\sum_{j=1}^{n}\frac{C_{66_j}ist}{C_{66_j}\max}\right)+0.25\left(\frac{1}{k}\sum_{l=1}^{k}\frac{C_{66l}ist}{C_{66l}\max}\right)+0.15(1-zp_i) & zp \text{ bekannt} \\ 0.7\left(\frac{1}{n}\sum_{j=1}^{n}\frac{C_{66_j}ist}{C_{66_j}\max}\right)+0.3\left(\frac{1}{k}\sum_{l=1}^{k}\frac{C_{66l}ist}{C_{66l}\max}\right) & zp \text{ unbekannt} \end{cases}$$

Gl. 14

wobei:
pd_i = Pflege-Dringlichkeit von Bestand i
n = Anzahl Z-Bäume im Bestand i
k = Anzahl Füllbäume im Bestand i
zp_i = Zielerreichungs-Prozent (DUDA 2006)

Nach der gleichen Vorgehensweise wird der Bedrängungszustand für den Füllbestand hergeleitet, um eine Aussage über die gesamte Bestandesdichte zu erhalten. Wurde für den Bestand eine Zielbestockung zugewiesen, geht neben den beiden konkurrenzbasierten Teil-Indizes zusätzlich das Zielerreichungsprozent in die Berechnung der Pflegedringlichkeit ein. Dieser Wert beschreibt den Grad der bereits erreichten Zielbaumartenzusammensetzung (DUDA 2006). Die zwei bzw. drei Teil-Indikatoren werden verschieden stark gewichtet und additiv verknüpft. Liegt das Zielerreichungsprozent nicht vor, wird diese Komponente aus der Gleichung zur Bestimmung der Pflegedringlichkeit eliminiert und die Gewichte für die Teilkomponenten *Bedrängungszustand Zukunftsbäume* und *Bedrängungszustand Füllbestand* auf 0,7 und 0,3 festgelegt. Liegen alle drei Indikatoren vor, verteilen sich die Gewichte wie folgt: 0,6 *Bedrängungszustand Zukunftsbäume*, 0,25 *Bedrängungszustand Füllbestand* und 0,15 *Zielerreichungsprozent*.

4.5.4.2 Nutzungsdringlichkeit

Bei der Quantifizierung der Endnutzungsdringlichkeit (*ed*) werden zwei Terme gleich gewichtet, additiv verknüpft (Gl. 14). Der erste Term beschreibt die Vorratsanteile zielstarker Bäume am Gesamtvorrat. Die Vorratsanteile werden für drei Durchmesserklassen berechnet. Die erste Klasse *z1* beinhaltet alle Bäume mit einem BHD ab Zielstärke bis einem BHD kleiner Zielstärke zuzüglich 5cm. Die zweite Klasse (*z2*) beinhaltet Bäume mit einem Durchmesser im Intervall Zielstärke+5cm ≤ BHD < Zielstärke+10cm. In die letzte Klasse *z3* fallen alle Bäume mit einem Durchmesser gleich oder größer der baumartenspezifischen Zielstärke+10cm. Die Vorratsanteile der einzelnen Klassen werden unterschiedlich gewichtet

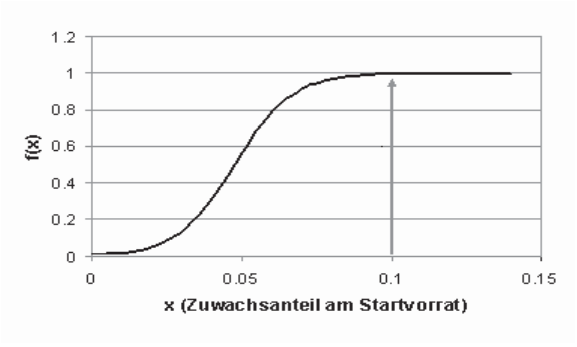
aufsummiert. Dadurch wird erreicht, dass der Anteil der Bäume, welche den Zieldurchmesser am stärksten überschreiten, die Endnutzungsdringlichkeit stärker erhöht. Der zweite Term beschreibt den relativen jährlichen Zuwachs aller zielstarken Bäume für eine 10-jährige Wachstumsperiode.

$$ed_i = 0.5\left(0.2\frac{v_{z1}}{v_{ges}} + 0.3\frac{v_{z2}}{v_{ges}} + 0.5\frac{v_{z3}}{v_{ges}}\right) + 0.5\left(1 - f\left(\frac{\Delta v_z}{v_z}\right)\right) \qquad \textit{Gl. 15}$$

wobei:
ed_i = Endnutzungsdringlichkeit des i-ten Bestandes
v_{ges} = Vorrat aller Bäume
v_{z1} = Vorrat Bäume mit Zielstärke ≤ BHD < Zielstärke+5cm
v_{z2} = Vorrat Bäume mit Zielstärke+5cm ≤BHD<Zielstärke+10cm
v_{z3} = Vorrat Bäume mit BHD≥Zielstärke+10cm
Δvz = Zuwachs zielstarker Bäume
v_z = Vorrat zielstarker Bäume
f = Skalierungsfunktion für den relativen Zuwachs

Zur Berechnung muss der jeweilige Bestand *i* folglich zehn Jahre fortgeschrieben werden. Der relative Zuwachs wird über eine Funktion f transformiert. Dadurch wird erreicht, dass ein relativer Zuwachs größer 10 Prozent einem Wert von 1 entspricht. Ein Zuwachs von 0 entspricht einem Wert von ebenfalls ungefähr 0. Für f wird eine sigmoide Funktion gewählt, welche in den Grenzbereichen (nahe 0 und nahe 1) einen flacheren Verlauf aufweist, so dass Zuwachswerte im Grenzbereich zu relativ ähnlichen Transformationswerten führen (Abb. 42).

(a)



(b)

$$f(x) = \frac{1}{1 + e^{-(52.63x - 2.5)/0.5}}$$

Abb. 42: (a) Plot der verwendeten sigmoiden Funktion zum Transformieren der relativen jährlichen Zuwächse, (b) Sigmoide Transformationsfunktion.

Dadurch werden schwachen Zuwächsen geringere Werte zugewiesen, so dass eine schwache Zuwachssituation durch die Transformation „strenger“ bewertet wird. Der transformierte

Zuwachswert wird von 1 subtrahiert, so dass der zweite Term für hohe Zuwachswerte klein und für geringe Zuwachswerte groß bzw. maximal 1 wird, da in diesem Fall der Index zur Beschreibung der Endnutzungsdringlichkeit ebenfalls einen hohen Wert annehmen soll. Bestände mit zielstarken Bäumen, welche einen vergleichsweise höheren zukünftig zu erwartenden Zuwachs aufweisen, werden so als weniger eingriffsdringlich eingestuft als Bestände, auf denen die zielstarken Bäume geringere oder keine Zuwächse leisten. In letzteren Beständen sollte eher die Endnutzung eingeleitet werden, da hier für die erntereifen Bäume das Risiko einer Entwertung steigt.

4.5.5 Aggregation

Es gibt eine Vielzahl an Möglichkeiten, die räumliche Nähe von Beständen zu definieren (BAILEY u. GATRELL 1995, CHEN u. GADOW 2002). Beispielsweise könnte die Länge gemeinsamer Grenzen zur Beschreibung der Aggregation verwendet werden (KURTTILA et al. 2002). Eine weitere Möglichkeit beruht auf dem euklidischen Abstand bzw. der Distanz zwischen den Beständen. Oder es werden einem Bestand Bestände als direkte Nachbarn zugewiesen, wenn sie innerhalb eines definierten Buffers um den Bezugsbestand liegen (HURME et al. 2007).

In der vorliegenden Arbeit wird die räumliche Nähe bzw. die Klumpung über zwei untergeordnete Indikatoren definiert. Zum einen wird die minimale Wegstrecke zwischen allen ausgewählten Beständen berechnet, zum anderen wird ein Wert hergeleitet, welcher die Klumpung der ausgewählten Bestände beschreibt. Die benötigten räumlichen Rechenoperationen (minimale Wegstrecken ermitteln, direkte Nachbarn bestimmen usw.) werden nicht in einem externen GIS durchgeführt, sondern direkt mit der implementierten GIS-Komponente (Kap. 4.4.5) realisiert. Dies erhöht die Performance und die Praktikabilität, da ein zusätzlicher Datenaustausch und die zusätzliche Installation eines GIS entfallen.

Die kürzeste Wegstrecke zwischen allen Beständen wird je nach Verfügbarkeit von Geoinformationen über zwei verschiedene Vorgehensweisen bestimmt. Liegen keine Geoinformationen zur Erschließung und zum Wegenetz vor, wird der euklidische Abstand zwischen zwei Beständen verwendet (Gl. 16).

$$d_{ij} = \sqrt{\left(v_{i1} - v_{j1}\right)^2 + \left(v_{i2} - v_{j2}\right)^2} \qquad \textit{Gl. 16}$$

wobei:
d_{ij} = Distanz zwischen Bestand i und Bestand j
$\vec{v}_i$, $\vec{v}_j$ = Ortsvektoren der Centroide von Bestand i und j

Liegen Geoinformationen zum Wegenetz vor, wird die kürzeste Route zwischen den einzelnen Beständen berechnet und anschießend die optimale Eingriffsreihenfolge bestimmt (vgl. Abb. 43 a). Zur Berechnung der kürzesten Route zwischen zwei Beständen wird die modifizierte Variante des Dijkstra-Algorithmus (vgl. Kap. 3.8.3) verwendet. Dieser Algorithmus ist ein geeignetes Verfahren, um in einem Wegenetz die optimale (günstigste, kürzeste) Route zwischen einem Start- und einem Zielknoten zu ermitteln. Durch Modifikation eines erweiterten, attributierten Graphen können weitere Routeneigenschaften, wie Gefälle oder Art der einzelnen Wegstrecken berücksichtigt werden. Das System muss das gegebene Erschließungssystem in einen Graphen im Sinne der Graphen-Theorie übersetzen (vgl. Kap 3.8). Um den zu durchsuchenden Graphen so klein wie möglich zu halten und somit die Suchgeschwindigkeit so schnell wie möglich zu gestalten, wird um alle vom Hauptsuchalgorithmus aktuell ausgewählten Bestände ein rechteckiger Buffer gelegt. Dieser wird mit dem Wegenetz verschnitten und alle in dem Buffer liegenden oder seine Grenze schneidenden Wege zum Aufbau des Graphen ausgewählt. Knotenpunkte, an denen nur zwei Wege (z. B. unterschiedlichen Typs) zusammenlaufen, werden automatisch entfernt. Dies ist möglich, da in diesem Fall nur die Wegstrecke und keine weiteren Attribute berücksichtigt werden. Durch dieses Vorgehen wird der Suchraum (der Graph) zusätzlich verkleinert.

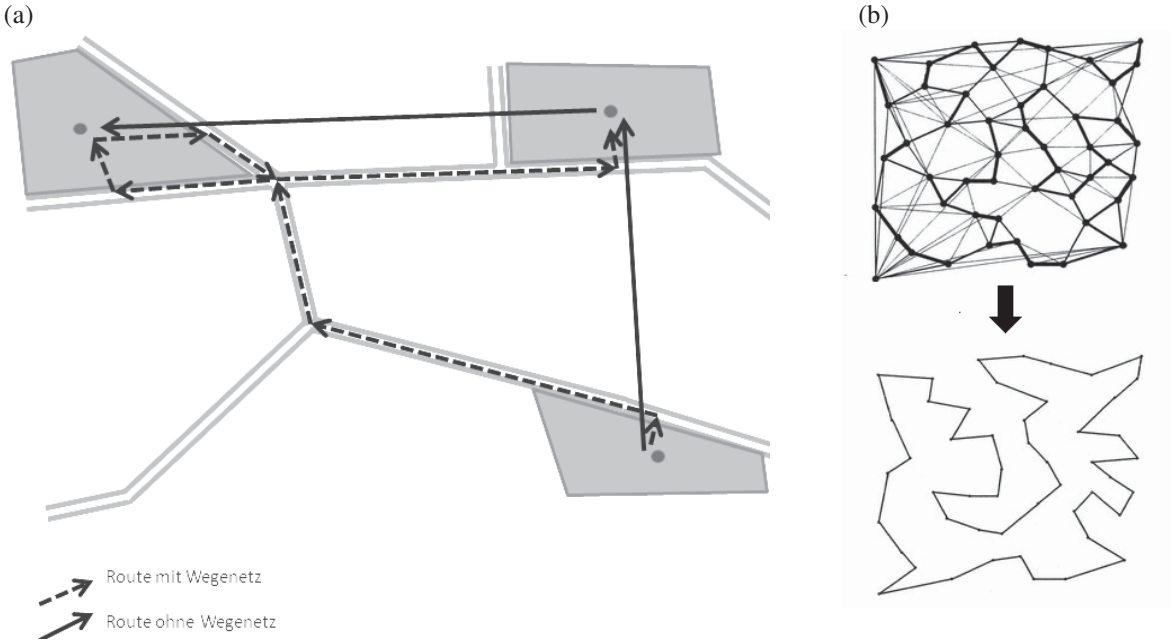


Abb. 43: a: Euklidische Distanzen (durchgezogen) und kürzeste Route im vorliegenden Wegenetz (gestrichelt). b: Traveling-Salesman-Problem: Ein gegebenes System aus Knoten (Beständen) und Verbindungen, die alle einmal in einer Reihenfolge angesteuert werden sollen, die die benötigte Gesamtstrecke minimiert. sowie ein Lösungsvorschlag des Minimierungsproblems.

Die implementierte Klasse *Graph* erlaubt es den Startknoten zu verändern, ohne den gesamten Graphen neu aufbauen zu müssen. Dies hat den Vorteil, dass die benötigte Rechenzeit für die Bestimmung der kürzesten Route zwischen zwei Beständen minimiert wird. Das System ist somit in der Lage, die kürzeste Route zwischen zwei Beständen zu berechnen. Die Suche nach der kürzesten Route, die alle ausgewählten Bestände einschließt, stellt ein ähnliches Problem dar, wie es bei dem klassischen Traveling-Salesman-Problem vorliegt (vgl. Abb. 43 b). Es gilt die Eingriffsreihenfolge so festzulegen, dass die Summe aller Umsetz-Wegstrecken möglichst klein wird. Dieses Problem wird mit einem metaheuristischen Optimierungsverfahren gelöst. Neben der Auswahl von Beständen zur Nutzung wird somit auch die optimale Reihenfolge, in der die einzelnen Bestände abzuarbeiten sind, ermittelt. Die Suchraumgröße zu diesem Optimierungsproblem berechnet sich nach Gl. 17.

$$D=(n-1)!/2 \qquad \textit{Gl. 17}$$

wobei:
D = Anzahl möglicher Reihenfolge-Kombinationen,
N = Anzahl ausgewählter (anzusteuernder) Bestände

In der vorliegenden Arbeit wird die Aggregation (*A*) der Bestände über einen der Durchmischung (GADOW 2003) angelehnten Indikator beschrieben. Zu allen aktuell ausgewählten Beständen (*i*) wird der Anteil direkter, ebenfalls ausgewählter Nachbarbestände an der maximal möglichen Anzahl ausgewählter Nachbarn (p_i) berechnet (Gl. 18).

$$A=\begin{cases}\dfrac{1}{n}\displaystyle\sum_{i=1}^{n}\frac{1}{p_i}\sum_{j=1}^{m_i} g_{ij} & n>1 \\ 1 & n=1\end{cases} \qquad \textit{Gl. 18}$$

mit $(i \neq j)$

und

$$g_{ij}=\begin{cases}1 & j\ genutzt \\ 0 & j\ nicht\ genutzt\end{cases}$$

und

$$p_i=\begin{cases}m_i & m_i \leq n-1 \\ n-1 & m_i > n-1 \\ 1 & m_i = 0\end{cases}$$

wobei:
n = Anzahl ausgewählte Bestände
i = Bezugsbestand
j = Nachbarbestand von Bestand i
m_i = Anzahl Nachbarbestände von Bestand i

Übersteigt die Gesamtanzahl ausgewählter Bestände (*n*) vermindert um 1 die Anzahl der direkten Nachbarn des Bestandes, wird p_i gleich der Anzahl der Nachbarn (m_i) des Bestands gesetzt. Ansonsten ist p_i gleich der Anzahl aller ausgewählten Bestände minus 1. Das Mittel über alle ausgewählten Bestände beschreibt den Grad der Aggregation. Im vorliegenden System sind zwei Bestände benachbart, wenn die zugehörigen Polygone innerhalb einer

Toleranz von 5 Metern einen gemeinsamen Grenzabschnitt besitzen. Um die Ermittlung direkter Nachbarn nicht bei jedem Optimierungslauf erneut durchführen zu müssen, werden diese nur einmal bestimmt und die Bezugsbestands-ID und die IDs der zugehörigen Nachbarn in einer Datenbanktabelle gespeichert. Der so berechnete Aggregations-Index kann Werte zwischen 0 und 1 annehmen. Je höher der Index ist, umso stärker sind die zur Nutzung vorgeschlagenen Bestände räumlich aggregiert. Wird vom Suchalgorithmus nur ein Bestand ausgewählt, nimmt *A* immer den Wert 1 an. Abb. 44 zeigt ein Beispiel für die Berechnung des Aggregations-Indikators für vier verschiedene Auswahlsituationen. In allen dargestellten Situationen werden insgesamt vier Bestände ausgewählt. In der ersten Auswahlsituation (oben links) hat keiner der vier Bestände einen direkten Nachbarbestand, der ebenfalls ausgewählt ist. Der Aggregations-Indikator ist somit 0. Die höchste Aggregation in der unten links dargestellten Situation erzielt. Alle ausgewählten Bestände haben laut Definition drei direkte Nachbarn, welche ebenfalls ausgewählt wurden. *A* ist in diesem Fall *1/4(3/3+3/3+3/3+3/3)=1*. In der unten rechts dargestellten Situation tritt der Fall ein, dass für einen Bestand (dunkler eingefärbt) m_i kleiner ist als *n-1*. Dieser Bestand könnte maximal zwei ebenfalls ausgewählte Nachbarn haben, da er insgesamt nur zwei direkte Nachbarn hat. In diesem Fall ist $p_i = 2$ und die Aggregation berechnet sich wie folgt: *1/4(**1/2**+1/3+1/3+1/3)=0,375*

Durch diese Vorgehensweise wird verhindert, dass die Aggregation unterschätzt wird, wenn mehr Bestände ausgewählt wurden, als ein einzelner Bestand an Nachbarn aufweisen kann.

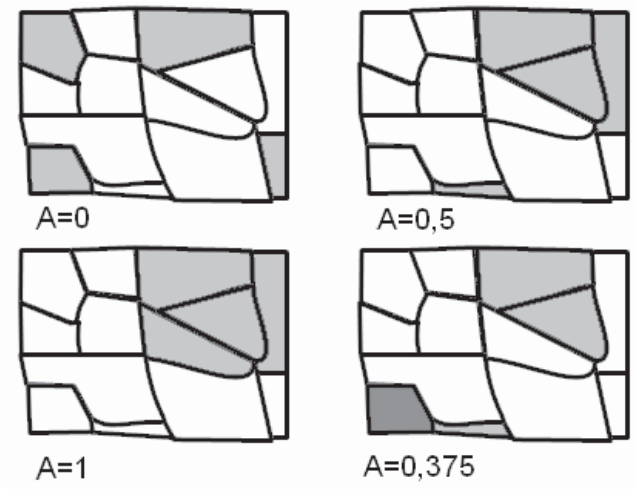


Abb. 44: Aggregations-Indikator A für vier verschiedene Auswahlsituationen mit jeweils vier ausgewählten Beständen.

4.6 Optimierungsmodul

Das Optimierungsmodul stellt die vierte Hauptkomponente des in dieser Arbeit entwickelten forstlichen DSS dar. In Kap. 3.7 wurden mehrere Verfahren hinsichtlich ihrer Lösungsgüte und Lösungsgeschwindigkeit verglichen. Dabei hat sich gezeigt, dass die Varianten PSA, PGA und SAMG hinsichtlich dieser beiden Parameter die besten Ergebnisse im Vergleich mit den übrigen Verfahren liefern. Da das Verfahren PSA durch die Verwendung mehrerer Nachbarschaftsrelationen sehr robust arbeitet und im Vergleich auf Basis zwei verschiedener Optimierungsprobleme die besten qualitativen Lösungen erzielt hat, wird dieses Verfahren vorzugsweise bei der Implementierung des Optimierungsmoduls verwendet. In den nachfolgenden Kapiteln wird die Konzeption des Optimierungsverfahrens zu den drei eingangs der Arbeit definierten Problemstellungen (*Optimale Bestandesauswahl zur Bereitstellung definierter Sortimente*, *Optimierung der Nutzung und Pflege*, *Optimierung der Bestandesauswahl zur Schaffung von Naturschutzflächen*) dargestellt. Dabei werden vor allem die Konstruktion der Zielfunktion, das Vorgehen zum Generieren der Lösungsalternativen und der Ablauf des gesamten Lösungsprozesses behandelt.

Wie gut oder schlecht durch eine im Rahmen der Optimierung generierten Lösung die Ziele des jeweiligen Betriebs erreicht werden können, wird über die sog. Ziel- oder Nutzenfunktion quantifiziert. Da bei allen drei zu lösenden Optimierungsproblemen mehrere Indikatoren in die Zielfunktion einbezogen werden sollen, resultiert daraus das Problem, dass nicht alle Zielindikatoren direkt miteinander zu vergleichende Werte liefern. Berücksichtigt eine Zielfunktion z. B. den Holzerlös und die absolute Totholzmenge zu gleichen Anteilen, ergibt sich das Problem, diese beiden Werte in einer Funktion sinnvoll miteinander zu verknüpfen. Eine Möglichkeit besteht darin, alle Indikatoren in ein einheitliches Wertesystem zu übertragen. Hierzu müsste in diesem Fall ermittelt werden, wie viel Euro ein m³ Totholz wert ist. Eine andere Möglichkeit besteht darin, alle Indikatoren so zu normieren, dass sie nur Werte zwischen 0 und 1 annehmen können. Da einige der von dem vorgestellten System verwendeten Zielindikatoren bereits normiert berechnet werden, bietet sich letzteres Vorgehen an. Sind der minimal und der maximal mögliche Wert eines Indikators *x* bekannt, bietet sich die in Gl. 19 dargestellte lineare Transformation an.

$$x_t = (x - x_{\min}) / (x_{\max} - x_{\min})$$ *Gl. 19*

wobei:
x_t = Transformation von x
x_{min} = minimal möglicher Wert von X
x_{max} = maximal möglicher Wert von x

Neben der linearen Transformation sind noch zwei weitere Transformationstypen möglich: Exponentiell und Sigmoid (KEENEY u. SICHERMAN 1976). In Abb. 45 sind die drei grundlegenden Transformationstypen dargestellt. Über eine geeignete Transformationsfunktion wird zum einen die Skalierung der Indikatorwerte auf das Intervall von 0 bis 1 vollzogen. Dies erleichtert die Konstruktion der Zielfunktion hinsichtlich der Verknüpfung und Gewichtung der einzelnen Indikatoren bzw. Teilnutzenfunktionen. Zum anderen wird eine Bewertung des Indikators im Intervall zwischen der kleinsten und größten Ausprägung bestimmt. Da die Bewertung von der Transformationsfunktion abhängt, ist bei der Wahl einer Transformationsfunktion die jeweilige Verzerrung des Ausgangswertes zu berücksichtigen. Bei der linearen Funktion führt eine Zunahme des Indikatorwertes zu einer konstanten Vergrößerung des transformierten Wertes. Der exponentielle Funktionstyp führt bei Vergrößerung des Indikators zu einer immer höheren Bewertung des Indikators.

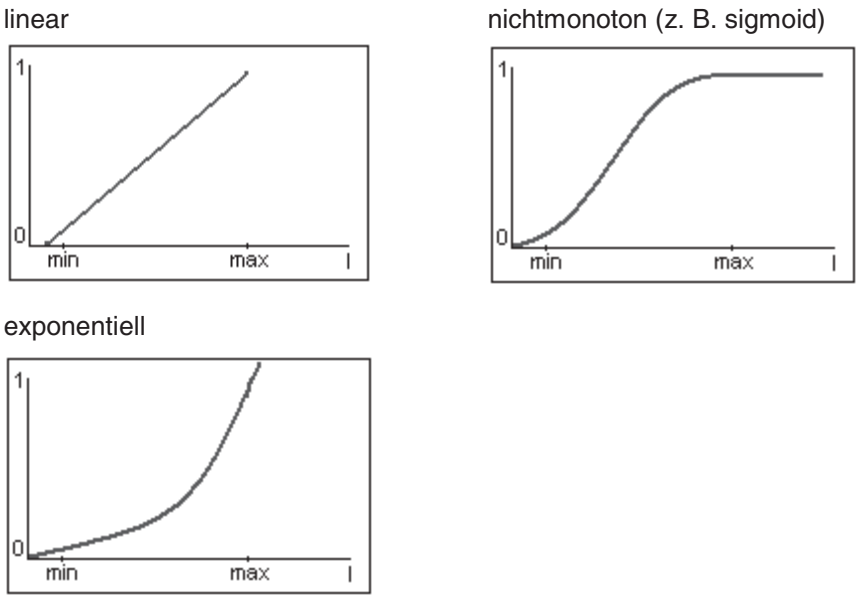


Abb. 45: Grundtypen der Transformationsfunktion.

Bei dem sigmoiden Typ ist in den Grenzbereichen nahe des Maximums bzw. des Minimums des Indikatorwertes eine geringere Steigung der Bewertungsfunktion gegeben. Dieser Typ wird dann eingesetzt, wenn z. B. ab einem bestimmten Wert des Indikators diesem bei einer Steigerung kein gesteigerter Nutzen beigemessen wird oder die minimalen und maximalen Grenzen des zu transformierenden Indikators nicht bekannt sind.

Neben der Vergleichbarkeit einzelner Indikatoren führt eine mehrkriterielle Zielfunktion zu einem Gewichtungsproblem. Sind die einzelnen Indikatoren transformiert, müssen sie noch miteinander verknüpft werden. Die einfachste Möglichkeit wäre, den Mittelwert aus allen

Indikatoren zu bilden. Dieses Vorgehen unterstellt jedoch, dass alle Indikatoren gleich gewichtet werden. Je nach Zielvorstellungen des Anwenders ist dies nicht der Fall. In dem vorgestellten System hat der Anwender zwei Möglichkeiten, die Gewichte der jeweiligen Indikatoren zu bestimmen. Sind die Gewichte bekannt oder vorgegeben, können sie direkt eingegeben werden. Die zweite Möglichkeit beruht darauf, einen Paarvergleich durchzuführen (SAATY 1980, LEXER 2000, SAATY 2000). Bei diesem Vorgehen werden alle Indikatoren paarweise gegenübergestellt und der Anwender muss auf einer Skala festlegen, wie viel wichtiger ihm der eine Indikator (*i*) als der andere (*j*) ist (= *V)*. Dabei gilt, dass V_{ij} der Reziprokwert von V_{ji} ist. Es sind dementsprechend bei *n* zu vergleichenden Indikatoren *n(n-1)/2* Paare zu beurteilen. Die Ergebnisse des Paarvergleichs werden in einer Paarvergleichsmatrix organisiert, mit welcher die einzelnen Gewichte über den Eigenvektor zum größten Eigenwert ermittelt werden können (vgl. Kap. 2.2.4). Die gewichteten und transformierten Indikatoren werden additiv verknüpft. Dabei kann die Zielfunktion aus beliebig vielen Ebenen bestehen. Dies bedeutet., die Indikatoren der höchsten Ebene können ihrerseits durch mehrere Indikatoren der nächst tieferen Ebene definiert werden. Die einzelnen Indikatoren werden basierend auf ihrer hierarchischen Ordnung zu einer mehrdimensionalen Nutzenfunktion additiv verknüpft. Additive Nutzenfunktionen unterstellen die nur selten als erfüllt anzunehmende Substituierbarkeit und Präferenzunabhängigkeit der Kriterien (KEENEY u. RAIFFA 1976). Ihr Einsatz ist jedoch als pragmatischer Lösungsansatz zu rechtfertigen, da meist keine praktikablen Alternativen vorliegen (LEXER 2000).

Da die Optimierungskomponente objektorientiert implementiert wurde (vgl. Kap. 3.6), müssen für die drei vorgestellten Optimierungsprobleme lediglich Implementierungen der Interfaces *CostFunction* (Zielfunktion) und *VariationFunction* (Funktion zum Bestimmen neuer Lösungen) problemspezifisch realisiert werden. Eine Veränderung an der Implementierung des Suchalgorithmus ist nicht erforderlich. Bei der Implementierung der Klasse *VariationFunction* sind zwei Punkte zu berücksichtigen. Der Methode *vary* wird ein Parametersatz übergeben, welcher von dieser so variiert wird, dass eine neue Parameterkombination aus der direkten Nachbarschaft der übergebenen Kombination resultiert. Des Weiteren muss der neue Parametersatz gültig sein. Er darf also keine der ggf. geforderten Restriktionen verletzen. Bei allen vorgestellten Optimierungsproblemen entspricht die Anzahl der Parameter der Anzahl der Bestände, für die eine optimale

Behandlung ermittelt werden soll oder aus denen eine optimale Untermenge bestimmt werden soll.

Die Methode *getCosts*, die bei einer Implementierung des Interfaces *CostFunction* „erzwungen“ wird, erwartet einen gültigen Parametersatz, um die Kosten oder den Nutzen aus dieser Parameterkombination zu ermitteln. Bei der Konstruktion der Zielfunktion ist darauf zu achten, dass die Zielfunktionswerte durch geeignete Transformationen in einem Intervall zwischen 0 und 1 liegen. In diesem Fall ist bei der Parametrisierung des Suchalgorithmus keine problemspezifische Anpassung erforderlich, da dieser standardmäßig mit für diesen Wertebereich angepassten Parametern (Starttemperatur, Abkühlungsstrategie, Abbruchkriterien) initialisiert wird.

4.6.1 Optimale Bestandesauswahl zur Holzbereitstellung

Im Rahmen der optimalen Bestandesauswahl soll eine Lösung des operativen Problems der momentanen Verfügbarkeit von Rohholz unter Berücksichtigung räumlicher und bestandesspezifischer Restriktionen ermittelt werden. Hierzu wird eine optimale Teilmenge aus allen Beständen ausgewählt. Optimal bedeutet, dass unter Einhaltung der geforderten Rohholzmenge solche Bestände zur Nutzung vorgeschlagen werden, die einen hohen Grad an Pflege- oder Nutzungsdringlichkeit aufweisen, eine vergleichsweise hohe Menge Rohholz liefern und räumlich dicht beieinander liegen. Die räumliche Nähe und Verteilung der ausgewählten Bestände ist dabei ein wichtiger Faktor. Je weiter die einzelnen Bestände auseinander liegen und je geringer die Aggregation der Bestände ist, umso größer werden der Polter- sowie der Abfuhraufwand und der Zeitbedarf zum Umsetzen von Maschinen (logistischer Aufwand). Inwieweit diese beiden Faktoren berücksichtigt werden, wird vom Anwender definiert. Mit einer sog. Slidebar (Schiebregler) kann er die Gewichtung des logistischen Aufwands und der Pflegedringlichkeit direkt festlegen. Da lediglich zwei Faktoren berücksichtigt werden, muss in diesem Fall keine systematische Gewichtung des jeweiligen Teilnutzens vorgenommen werden.

Die Zielfunktion setzt sich auf der obersten Ebene aus den zwei Indikatoren *Logistischer Aufwand* und *Handlungsdringlichkeit* zusammen (Abb. 46). Da die Gesamtnutzenfunktion maximiert werden soll, muss der Indikator *Logistischer Aufwand* entsprechend transformiert werden, so dass für einen geringen Aufwand ein großer Indikator-Wert resultiert. Auf der zweiten Ebene der Zielfunktion setzt sich der *logistische Aufwand* wiederum aus zwei Indikatoren zusammen: Distanz bzw. Wegstrecke (*D*) und Aggregation (*A*). Die Aggregation

erhält ein Gewicht von 0,4 und die Wegstrecke entsprechend 0,6. Die Eingriffsdringlichkeit wird über den größeren der beiden Indikatoren Pflege (*P*)- oder Nutzungsdringlichkeit (*N*) bestimmt.

		G			G		T
Gesamtnutzen	Σ	n.d	min← logistischer (Umsetz- u. Abfuhr-) Aufwand	Σ	0.6	D (Wegstrecken/Distanz)	sigmoid
					0.4	A (Aggregation)	linear
		n.d	max← mittlere Eingriffsdringlichkeit	Σ	1 oder 0	P (Pflegedringlichkeit)	linear
					1 oder 0	N (Nutzungsdringlichkeit)	linear

Abb. 46: Ermittlung des Gesamtnutzens einer Auswahl von Beständen im Kontext der kurzfristigen Optimierung unter Berücksichtigung der Eingriffsdringlichkeit und des Polter- bzw. Abfuhraufwands, n.d.= durch Nutzer definiert.

Zur Ermittlung der optimalen Bestandesauswahl werden insgesamt vier Schritte abgearbeitet (Abb. 47). Im ersten Schritt werden die nutzerspezifischen Parameter abgefragt. Der Anwender kann Optionen zur Beschränkung der Nutzung bis zu einer maximalen Hangneigung, oder zum Nutzungsverbot auf einzelnen Flächen festlegen. Weiterhin muss der Anwender die gesuchten Zielsortimente und die jeweils gewünschten Mengen spezifizieren.

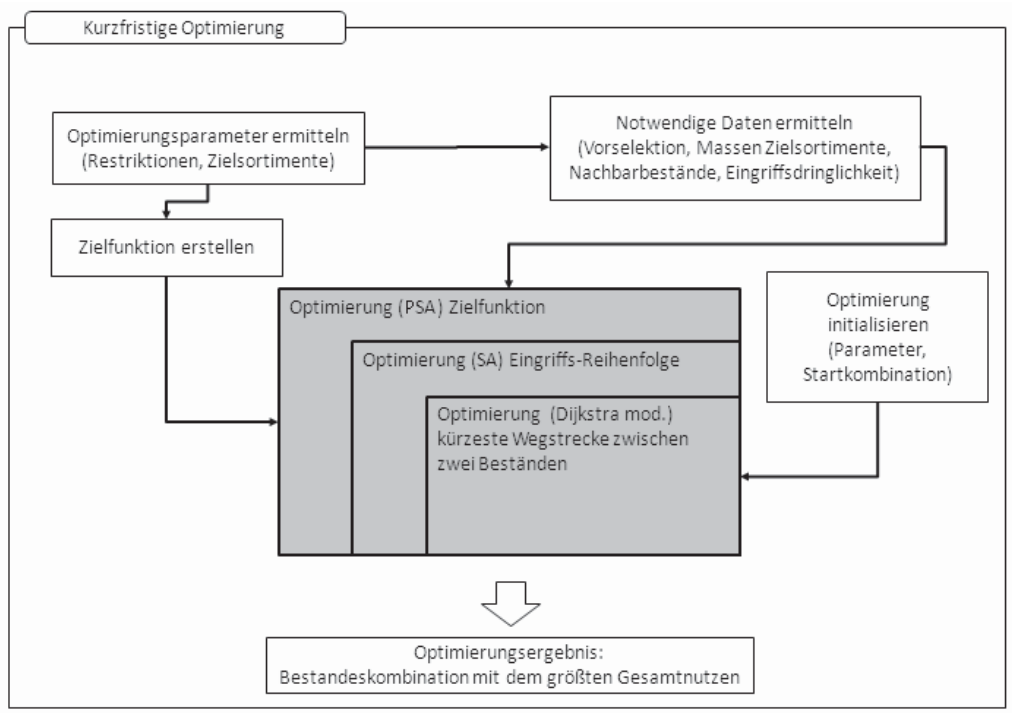


Abb. 47: Ablauf der kurzfristigen Optimierung (Bestandesauswahl).

Die Sortimente werden über die Parameter *Baumart*, *Mittendurchmesser* und *Länge* definiert. Bei den Dimensionsangaben werden Ober- und Untergrenze abgefragt. So ist es möglich, einen Wertebereich oder einen konkreten Wert (z. B. Fixlänge) vorzugeben. Auf Basis dieser Informationen wird der Suchraum vorab verkleinert. Es werden nur solche Bestände in die zu durchsuchende Menge aufgenommen, die keine der Restriktionen verletzen und in denen mindestens eine Baumart aus der Liste der spezifizierten Sortimente vorkommt. Durch diese Reduktion des Suchraums werden sowohl die Berechnung der einzelnen Indikatoren als auch der eigentliche Optimierungsvorgang beschleunigt. Im zweiten Schritt wird die Handlungsdringlichkeit der einzelnen Planungseinheiten bestimmt. Hierzu wird auf die in Kap. 4.5.4 beschriebene Methode zur Bestimmung der Handlungsdringlichkeit zurückgegriffen. Anschließend werden die theoretisch zur Verfügung stehenden Mengen, der vom Nutzer gewünschten Zielsortimente ermittelt. Diese werden über die Durchführung eines virtuellen Eingriffs (Durchforstung und Endnutzung, vgl. Kap. 4.4.1) je Bestand und einer anschließenden Sortimentierung der entnommenen Bäume bestimmt. Dabei werden nur solche Bestände berücksichtigt, die in der Vorauswahl (Schritt 1) selektiert wurden. Die Eingriffsmodalitäten werden einer XML-Datei entnommen, wobei ein moderates Eingriffsszenario in Anlehnung an einen naturnahen Waldbau vordefiniert ist. Der Anwender kann die Eingriffsparameter jedoch auch frei anpassen. Über die spezifizierten Eingriffsparameter wird das in Kap. 4.4.2 vorgestellte Eingriffsmodell gesteuert. Anschließend kann der eigentliche Optimierungsprozess gestartet werden (Schritt 4). Die Suchraumgröße zu diesem Problem hängt von der Anzahl der ausgewählten Bestände (*nsel*) ab ($D=2^{nsel}$). Der Optimierungsprozess wird durch das Generieren einer Startlösung eingeleitet. Lösungen werden für das vorliegende Optimierungsproblem binär kodiert. Folglich wird für jeden der potentiellen Bestände eine 0 oder 1 vergeben und dieser Wert in einem Array an einer fixen Speicherstelle abgelegt. Eine 1 wird dann vergeben, wenn ein Bestand ausgewählt wurde. Andernfalls wird in dem Array eine 0 gespeichert. Auf Basis dieser Kodierung wird eine Implementierung des Interfaces *VariationFunction* (Kap. 3.6) erstellt, die an einer oder mehreren zufällig bestimmten Speicherstellen des Arrays der übergebenen Ausgangs- bzw. Vorgängerlösung die Nullen gegen Einsen vertauscht und umgekehrt. Die neue Lösung wird dann an die Implementierung des Interfaces *CostFunction* übergeben, welches die neue Lösung auf Verletzung von Restriktionen prüft und den Zielfunktionswert berechnet. Die Startlösung wird generiert, indem Bestände aus der Menge aller potentiellen Bestände zufällig ausgewählt werden, bis die gewünschten Mengen der

spezifizierten Sortimente erreicht werden. Mit dieser Lösung wird das parallele Simulated Annealing initialisiert und gestartet, um die optimale Kombination aus der Gesamtheit aller Bestände zu suchen. Im Rahmen der Optimumsuche müssen ausgehend von der Vorgängerlösung neue Lösungsvorschläge (*VariationFunction*) generiert und mit der Zielfunktion bewertet werden. Hierzu werden zunächst alle benötigten Indikatoren einzeln berechnet. Die für alle potentiellen Bestände einmalig berechneten Pflege- bzw. Handlungsdringlichkeiten werden für die ausgewählten Bestände mit der zugehörigen Fläche gewichtet und aufsummiert. Durch Division mit der Summe der Flächen aller ausgewählten Bestände ergeben sich die Indikatoren *N* und *P*. Die Berechnung der Aggregation erfolgt auf Basis der Nachbarschaftsbeziehungen zu den anderen ausgewählten Beständen (Kap. 4.5.5). Die Berechnung der benötigten (minimalen) Wegstrecken, um alle ausgewählten Bestände anzusteuern, ist ein eigenes Optimierungsproblem und wird, wie in Kap. 4.5.5 beschrieben gelöst. Da zur Bestimmung der minimalen Wegstrecke bei einem gegebenen Wegenetz zwei Optimierungsverfahren zum Einsatz kommen (Simulated Annealing und Dijkstra), werden folglich zur Bestimmung der optimalen Bestandesauswahl bis zu drei Suchverfahren ineinander verschachtelt (Abb. 47). Im letzten Schritt werden nach Ablauf der Optimierung die binär kodierte Lösung und die dadurch erreichten Werte der zur Bewertung verwendeten Indikatoren in eine vom Anwender lesbare Form übersetzt, in einem Dialog dargestellt und auf Wunsch in einer Datenbanktabelle gespeichert.

4.6.2 Optimierung von Pflege und Nutzung für einen Einrichtungsturnus

Im Rahmen der Optimierung der Eingriffszeitpunkte und -stärken für die Vor- und Endnutzung sollen für jeden Bestand des Betriebs die Eingriffsmodalitäten für einen zehnjährigen Zeitraum ermittelt werden, die ein mehrkriterielles Zielsystem (Abb. 48) für den gesamten Betrieb optimieren. Damit wird das Ziel verfolgt, die Planung der bestandesweisen Behandlung im Rahmen der klassischen Forsteinrichtung zu unterstützen. Das Zielsystem setzt sich aus den Komponenten konstante Nutzung (*ef*), ökonomischer Erfolg (*oet*), logistischer Aufwand (*la*) und Verbesserung der Eingriffsdringlichkeit (*egd*) zusammen. Die Indikatoren *konstante Holznutzung* und *räumlich-zeitliche Aggregation* der genutzten Bestände zielen vor allem darauf ab, bei limitierten jährlichen Arbeitskapazitäten diese optimal einsetzen zu können. Es werden alle Indikatoren in die Zielfunktion aufgenommen, wobei die Gewichtung, wie eingangs des Kap. 4.6 beschrieben, vorgenommen wird. Soll ein Indikator völlig unberücksichtigt bleiben, muss ihm ein Gewicht von 0 zugewiesen werden.

Die Teilnutzen *la* und *egd* werden von 1 subtrahiert, da die Nutzenfunktion maximiert werden soll und der Nutzen von *la* und *egd* mit kleineren Werten größer wird. Zur Lösung des vorliegenden Optimierungsproblems wird auf die simulationsbasierte Optimierung in Kombination mit dem Mehrpfad-Prinzip (GADOW 2006) zurückgegriffen.

		G		T	G		T
Gesamtnutzen	Σ	n.d.	1-la (logistischer Aufwand)		0.3	D (Wegstrecken/Distanz)	sigmoid
					0.7	A (Aggregation)	linear
		n.d.	ef (konstante Nutzung)	linear			
		n.d.	1-egd (Eingriffsdringlichkeit)	linear			
		n.d.	oet (Ökonomischer Erfolg)	linear			

Abb. 48: Aufbau der Zielfunktion zur Optimierung der Eingriffsmodalitäten. G =Gewichtung, n.d. = wird durch Nutzerinteraktion definiert, T = Transformationstyp.

Es werden für jeden Bestand verschiedene Behandlungspfade simuliert und die benötigten Zielindikatoren für jeden Pfad berechnet. Das Generieren der Pfade wird durch zwei Instanzen der Klasse *PathGenerator* durchgeführt. Ähnlich wie das eigentliche Simulationssystem arbeiten die beiden Instanzen parallel, wodurch die Simulation der Pfade und die Berechnung der Zielindikatoren deutlich beschleunigt werden. Anders als das System zur Variantensimulation lädt der *PathGenerator* einen Bestand aus der Datenbank in den Arbeitsspeicher und erzeugt vor der Simulation eines Pfades im Arbeitsspeicher eine Kopie des vollständig eingelesenen Modellbestands. Da bei der Pfadgenerierung mehrere Varianten/Pfade ausgehend von dem selben Ausgangszustand simuliert werden müssen, spart dieses Vorgehen ebenfalls deutlich Rechenzeit ein.

Die Suchraumgröße *D* zu dem vorliegenden Optimierungsproblem ergibt sich aus der Anzahl der Bestände, der Anzahl verschiedener Eingriffsstärken und der Anzahl der möglichen Eingriffszeitpunkte (Gl. 20).

$$D = (e \cdot t)^n \qquad \text{Gl. 20}$$

wobei:
t= Anzahl möglicher Eingriffszeitpunktkombinationen
e= Anzahl möglicher Eingriffsstärken

Bei drei verschiedenen Eingriffsstärken beinhaltet der Suchraum für lediglich vier Bestände bereits ca. 37 Mio. Kombinationsmöglichkeiten. Die verschiedenen Kombinationen von Eingriffszeitpunkt und Eingriffsstärke können als Pfad im Sinne des Mehrpfadprinzips verstanden werden. Durch die Verwendung des Mehrpfadprinzips wird der Suchraum verkleinert und somit die Performance des Suchprozesses verbessert. Wird der Entwicklungspfad aus mehreren hintereinander geschalteten Behandlungsintervallen mit mehreren möglichen Behandlungsvarianten aufgebaut, vergrößert sich der Suchraum exponentiell mit der Anzahl der Eingriffsintervalle, da in jedem Intervall einem Bestand jede der definierten Varianten zugewiesen werden kann (Abb. 49 rechts). Bei den hier vorgestellten Optimierungsverfahren wird aufgrund der schnelleren Suche und der Beschränkung auf einen Forsteinrichtungsturnus auf die pfadbasierte Variante zurückgegriffen, welche von einer gleich bleibenden Bestandesbehandlung ausgeht, also keine dynamische Variation der Varianten eines Bestands vorsieht. Im ersten Schritt zur Optimierung der mittelfristigen Bestandesbehandlung werden die verschiedenen Varianten über die Maßnahmenelemente des verwendeten Behandlungsmodells (Kap. 4.4.2) definiert und in Kombination mit dem Wuchsmodell TreeGrOSS in den Modellbeständen virtuell umgesetzt.

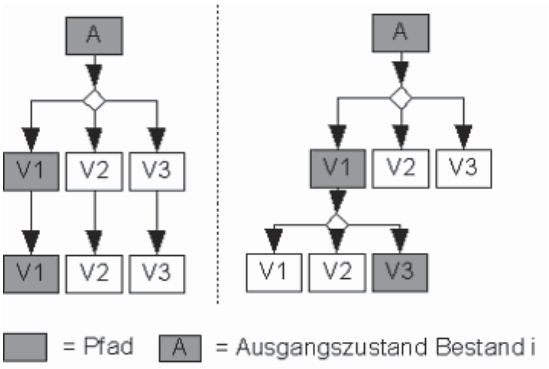


Abb. 49: Pfaddefinition. Links: Je Pfad wird nur eine Variante zugelassne. Rechts: Pro Pfad werden in jedem Simulationsintervall alle Varianten zugewiesen.

Alternativ zum Mehrpfadprinzip ist es möglich, die einzelbestandesweise Maßnahmenkombination und -definition in den Schranken gegebener Restriktionen und den Möglichkeiten der verwendeten Modelle völlig dem Suchalgorithmus zu überlassen. Die Suchraumgröße bei Behandlungsdefinition durch den Suchalgorithmus hängt im Wesentlichen von der Anzahl der einzelnen Behandlungselemente und der

Steuerparameteranzahl sowie der Anzahl der möglichen Werte der Steuerparameter ab. Die Suchraumgröße berechnet sich in diesem Fall nach Gl. 21.

$$D = \prod_{i=1}^{b} \prod_{j=1}^{m_i} \prod_{k=1}^{p_j} n_k$$

wobei:
b =Anzahl Bestände,
m_i = Anzahl möglicher Maßnahmen für Bestand i,
p_j = Anzahl Parameter von Maßnahme j,
n_k = Anzahl Ausprägungen von Parameter j

Gl. 21

Hinsichtlich der Eingriffszeitpunkte können kein, ein oder zwei Eingriffe in dem zehnjährigen Zeitraum bzw. innerhalb eines Entwicklungspfades modellhaft in einem Bestand durchgeführt werden. Werden zwei Eingriffe modelliert, so muss zwischen dem ersten und zweiten Eingriff mindestens eine Zeitspanne von vier Jahren liegen. Wird nur ein Eingriff durchgeführt, so kann dieser in jedem der simulierten 10 Jahre stattfinden. Es ergibt sich die in Tab. 14 dargestellte Matrix möglicher Eingriffszeitpunkte.

Tab. 14: Mögliche Eingriffszeitpunkte in einem Simulationsintervall von 10 Jahren und die entsprechende Kodierung, iES= Index der Eingriffsstärke (1 bis Anzahl verschiedener Eingriffsstärken), (1) = ein Eingriff, (2a/2b) =zwei Eingriffe, (x)= kein Eingriff erlaubt.

Jahr Kodierung	1	2	3	4	5	6	7	8	9	10
1+((iES-1)*26)	1									
2+((iES-1)*26)		1								
3+((iES-1)*26)			1							
4+((iES-1)*26)				1						
5+((iES-1)*26)					1					
6+((iES-1)*26)						1				
7+((iES-1)*26)							1			
8+((iES-1)*26)								1		
9+((iES-1)*26)									1	
10+((iES-1)*26)										1
11+((iES-1)*26)	2a	x	x	x	x	2b				
12+((iES-1)*26)	2a	x	x	x	x		2b			
13+((iES-1)*26)	2a	x	x	x	x			2b		
14+((iES-1)*26)	2a	x	x	x	x				2b	
15+((iES-1)*26)	2a	x	x	x	x					2b
16+((iES-1)*26)		2a	x	x	x	x	2b			
17+((iES-1)*26)		2a	x	x	x	x		2b		
18+((iES-1)*26)		2a	x	x	x	x			2b	
19+((iES-1)*26)		2a	x	x	x	x				2b
20+((iES-1)*26)			2a	x	x	x	x	2b		
21+((iES-1)*26)			2a	x	x	x	x		2b	
22+((iES-1)*26)			2a	x	x	x	x			2b
23+((iES-1)*26)				2a	x	x	x	x	2b	
24+((iES-1)*26)				2a	x	x	x	x		2b
25+((iES-1)*26)					2a	x	x	x	x	2b
26+((iES-1)*26)	x (keine Eingriffe)									

Insgesamt resultieren 26 verschiedene Eingriffszeitpunktkombinationen. Werden im Rahmen der Pfadgenerierung drei abgestufte Eingriffsstärken vorgesehen, müssen für jeden Bestand maximal 78 Simulationen durchgeführt werden. Die Simulation der möglichen Behandlungspfade wird vor dem Optimierungslauf durchgeführt. Die relevanten Indikatoren zu jedem Pfad und Bestand werden in einer geeigneten Datenstruktur im Arbeitsspeicher abgelegt. Dies ermöglicht den schnellstmöglichen Zugriff bei der Bewertung einer Pfadkombination (Lösung) durch die Zielfunktion während des Optimierungsprozesses. Zur Kodierung der einzelnen Pfade werden diese fortlaufend nummeriert. Bei zwei möglichen Eingriffsstärken verschlüsseln die Ziffern von 1 bis 26 alle möglichen Pfade mit den schwächeren Eingriffen. Die Ziffern von 27 bis 52 werden allen möglichen Eingriffszeitpunktkombinationen für die höhere Eingriffsstärke in gleicher Reihenfolge zugewiesen. Eine Pfadkombination beinhaltet dementsprechend für jeden Bestand des Betriebs an einer festen Speicherstelle (Index) die jeweilige Pfadnummer. Die Speicherstelle entspricht dabei der Speicherstelle der gespeicherten Indikatoren eines Bestandes. Die zu einer Pfadkodierung gehörenden Indikatoren können so über den Index direkt abgerufen werden. Die Variationsfunktion muss dementsprechend angepasst werden. An einer oder mehreren zufällig ausgewählten Stellen der übergebenen Pfadkombination muss eine andere, gültige Pfadcodierung eingetragen werden. Die kodierten Pfadkombinationen müssen dann entsprechend des Zielsystems bewertet werden. Hierzu wird eine Implementierung des Interfaces *CostFunction* erstellt, welche die Pfadkombinationen decodiert und die zu jedem Pfad eines Bestandes die zuvor gespeicherten Informationen abruft und aggregiert (Abb. 50). Die Indikatoren *egd* und *oet* werden aus den pfadspezifischen gespeicherten Bestandeswerten flächengewichtet gemittelt. Die Aggregation (*A*), die Wegstecke (*D*) und das Maß der Gleichverteilung der Nutzungsmassen (*ef*) berechnet sich wie in Kap. 4.5 beschreiben. *A* und *D* werden für jeden Eingriffszeitpunkt berechnet und gemittelt. Bei der Mittelwertberechnung werden die jährlichen Werte von *A* und *D* nicht mit der Fläche der genutzten Bestände gewichtet, da gerade auch bei geringeren jährlichen Nutzungen der logistische Aufwand minimiert werden soll. Eine hohe Aggregation eines Jahres mit großer Nutzungsfläche soll nicht schlechtere Aggregationen zu Eingriffszeitpunkten mit geringerer genutzter Fläche kompensieren. Dies wäre bei einer flächengewichteten Mittelwertberechnung der Fall.

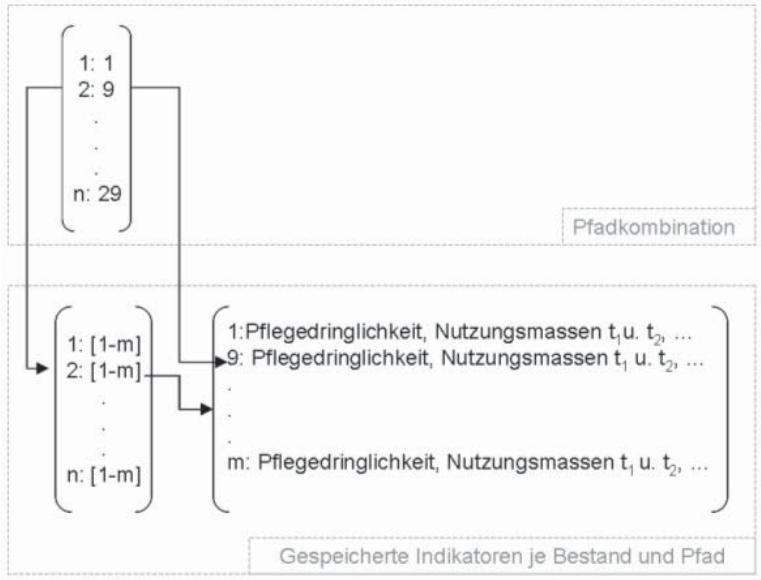


Abb. 50: Verknüpfung der kodierten Pfadkombination mit den für alle Pfade eines Bestands gespeicherten Zielindikatoren, n= Anzahl Bestände, m= Anzahl Pfade pro Bestand.

Der *CostFunction* werden bei der Initialisierung die nutzerdefinierten Restriktionsparameter übergeben (Abb. 51). Der Nutzer kann den Rahmen sowohl der jährlichen als auch der gesamten Nutzungsmassen vorgeben (Minimum/Maximum). Verletzt die aktuell zu bewertende Pfadkombination eine oder mehrere der Restriktionen, wird die Lösung nicht für ungültig erklärt. Es wird ein negativer, der Höhe der Restriktionsverletzung entsprechender Wert zurückgegeben. Beispielsweise führt eine Überschreitung der maximalen jährlichen Nutzungsmasse von 10 Prozent zu einem Funktionswert von -0,1. Da der Funktionswert maximiert werden soll, wird die Lösung somit als schlecht eingestuft. Da sie jedoch nicht ungültig ist und evtl. ein Potential für eine bessere Lösung darstellt, ist es für den Lösungsprozess von Vorteil, derartige Lösungen zwar als schlecht einzustufen, diese jedoch mit in den weiteren Lösungsprozess mit einzubeziehen. Der Ablauf des eigentlichen Optimierungsprozesses ist dem zur Lösung des Problems zu Bestandesauswahl sehr ähnlich. Nach der vorgezogenen Pfadermittlung sowie der Berechnung aller notwendigen Indikatoren werden die problemspezifischen Implementierungen der *CostFunction* und der *VariationFunction* dem parallelisierten Simulated Annealing (PSA) übergeben. Das Verfahren ermittelt die Pfadkombination, welche das vorgegebene Zielsystem optimiert. Nach der Optimumsuche wird eine Tabelle erstellt, welche für jeden Bestand den oder die optimalen Eingriffszeitpunkte und die entsprechenden Entnahmemengen beinhaltet.

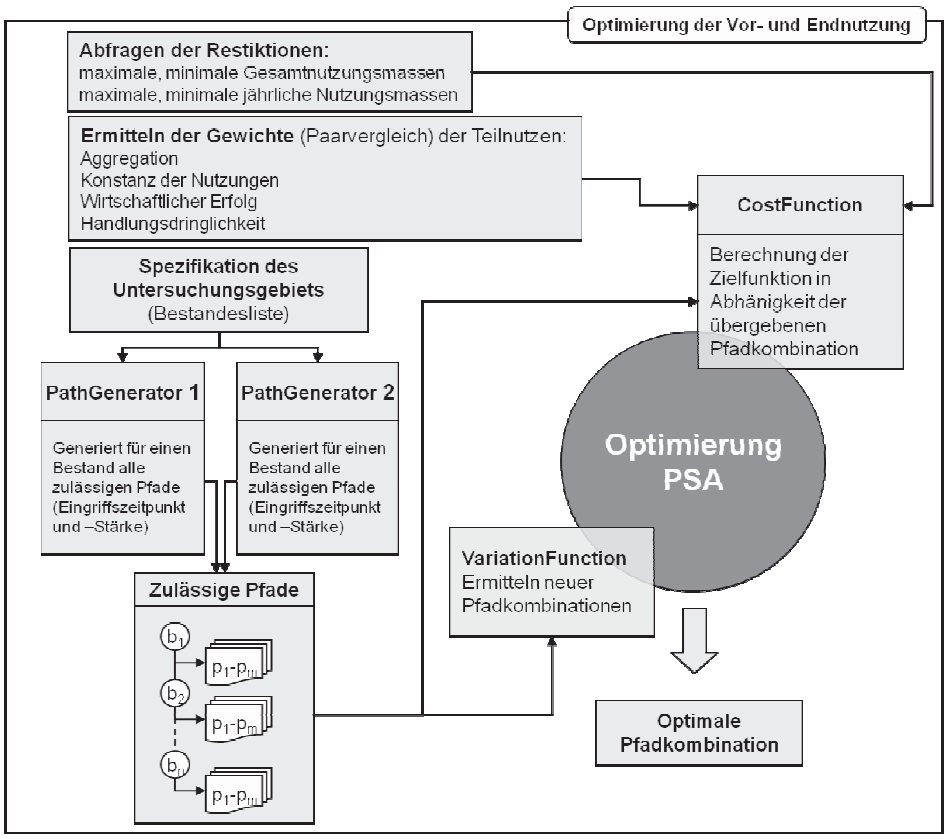


Abb. 51: Optimierung der Vor- und Endnutzungsmodalitäten für einen zehnjährigen Planungszeitraum.

4.6.3 Optimale Auswahl von Naturschutzflächen

Die in diesem Kapitel vorgestellte Optimierungskomponente soll dem Anwender helfen, konkrete Bestände auszuwählen, welche aus der regulären Nutzung ausgeschlossen werden sollen, um beispielsweise vertraglich vereinbarte naturschutzfachliche Auflagen zu erfüllen. In diesem Zusammenhang entsteht oft ein Zielkonflikt dadurch, dass die ausgewählten Bestände konkrete Eigenschaften aufweisen müssen und zum anderen die finanziellen Einbußen durch die Stilllegung der Flächen möglichst minimal ausfallen sollen. Z. B. stellen vor allem alte Bestände ein weitaus größeres Habitatpotential dar als jüngere Bestände (MONING u. MÜLLER 2009). Jedoch ist auch die monetäre Wertschöpfung bei der Nutzung älterer Bestände bis zu einem baumartenabhängigen Maximalalter, ab welchem mit hoher Wahrscheinlichkeit eine Entwertung einsetzt, deutlich höher. Die Eigenschaften, die die ausgewählten Bestände einzeln oder insgesamt aufweisen sollen, müssen externen Vorgaben oder den betrieblichen Zielen, welche durch die Stilllegung erreicht werden sollen, entsprechen. Die vorgestellte Optimierungskomponente kann dabei folgende Parameter berücksichtigen, welche in der Fachliteratur als wichtige Indikatoren oder Schwellenwerte für

potentielle Habitate beschrieben werden (CHRISTENSEN et al. 2005, MÜLLER et al. 2007, MONING u. MÜLLER 2009): *Hangneigung* und *Exposition*, *Alter des Hauptbestands*, *Baumartenzusammensetzung*, *vorhandenes Totholz* und *Totholzneubildung* sowie die *Mindestgröße der Hotspots*. Ein Hotspot wird aus einer Teilmenge oder allen ausgewählten Beständen gebildet, welche räumlich zusammenhängen bzw. benachbart sind (vgl. Kap. 4.5.5). Auf Basis dieser optionalen Restriktionen wird durch das Optimierungsverfahren eine Bestandesauswahl ermittelt, welche die definierten Restriktionen erfüllt und den geringsten finanziellen Verlust durch einen Nutzungsverzicht generiert.

Die Totholzneubildung und der Nutzenentgang werden simulativ ermittelt. Vor der eigentlichen Optimumsuche werden zwei waldbauliche Szenarien gerechnet. Der Simulationszeitraum umfasst jeweils 30 Jahre. Das eine Szenario sieht keinerlei Eingriffe vor. Es dient der bestandesweisen Abschätzung der Totholzdynamik. Zur Abschätzung der Totholzdynamik kommt das in Kap. 4.4.3 vorgestellte Totholzmodul zum Einsatz. Das zweite Szenario sieht eine normale Bewirtschaftung der Flächen vor und ermöglicht die Ermittlung der in den 30 Jahren anfallenden Vor- und Endnutzungsmassen sowie deren monetäre Bewertung. Das Bewirtschaftungsszenario kann vom Anwender im Rahmen der Steuerparameter frei definiert und der tatsächlichen betrieblichen Bewirtschaftungsstrategie angepasst werden. Die Bewertung des Nutzenentgangs erfolgt über die Ermittlung des erntekostenfreien Holzerlöses der modellierten Nutzungsmassen und einer entsprechenden Verzinsung (2 Prozent, vgl. DUDA 2006). Die erntekostenfreien Holzerlöse werden nach den von Duda vorgestellten Bewertungsfunktionen einzelstammweise ermittelt und bestandesweise aufsummiert (DUDA 2006). Das Alter des Hauptbestands und die Baumartenzusammensetzung beziehen sich auf die Ausgangsdatenlage und werden nicht im Rahmen einer Szenariosimulation ermittelt. Die Geländeinformationen werden auf Basis eines rasterbasierten digitalen Geländemodells durch die GIS-Komponente (Kap. 4.4.5) berechnet und mit den Bestandespolygonen verschnitten. Die beschreibenden Parameter werden bestandesweise in einer geeigneten Datenstruktur abgelegt. Zur Speicherung der Parameter wird eine neues Objekt definiert, welches alle relevanten Daten zu einem Bestand speichert. Diese Objekte werden für alle Bestände des betrachteten Gebiets zunächst auf Verletzung der Restriktionen geprüft, die an den Einzelbestand gebunden sind. Erfüllt ein Bestand alle Anforderungen, wird das entsprechende Objekt in einem Vektor an einer festen Speicherstelle gespeichert. Auf Basis dieses Vektors wird der initiale Parametersatz definiert, welcher von der Nutzenfunktion bewertet und von der Variationsfunktion modifiziert wird.

Der Parameter für einen entsprechenden Bestand steht an der selben Speicherstelle, wie das zum selben Bestand zugehörige Datenobjekt. Auf diese Weise wird ein sehr schneller Zugriff auf die bestandesspezifischen Daten ermöglicht. Die Parameter können die Werte 1 oder 0 annehmen. Eine 1 wird vergeben, wenn der Bestand ausgewählt werden soll. Die Bewertungsfunktion, aggregiert flächenscharf die Summe der bestandesweisen Parameter für die ausgewählten Bestände und berechnet den entsprechenden Zielfunktionswert. Restriktionen, welche für die gesamte Bestandesauswahl aufgestellt werden, können erst im Rahmen der Bewertung der Lösung während des Optimierungsprozesses geprüft werden. In diesem Fall werden Lösungen, die eine oder mehrere Restriktionen verletzen nicht als ungültig verworfen, sondern mit einem dem Grad der Restriktionsverletzung entsprechenden schlechten Zielfunktionswert versehen. So werden auch Lösungen in den Suchprozess mit einbezogen, welche zwar ungültig sind, jedoch eine gute Basis für eine sehr gute oder optimale Lösung darstellen können. Dies fördert insgesamt die Robustheit der Optimumsuche.

Der Umfang des Suchraums (*D*) ergibt sich aus der Anzahl der zulässigen Bestände und der binären Parameterdefinition und berechnet sich entsprechend nach Gl. 22.

$$D = 2^n$$

wobei:
n = Anzahl Bestände ohne Restriktionsverletzung

Gl. 22

Zur Lösung des Optimierungsproblems wird ebenfalls das Verfahren PSA (paralleles Simulated Annealing) eingesetzt. Die Implementierung der Zielfunktion (*TargetFunction*) berechnet das in Abb. 52 dargestellte Zielsystem. Werden Restriktionen oder Zielgrößen auf Basis der nutzerspezifischen Vorgaben aus dem Zielsystem eliminiert, werden die Gewichte der entsprechenden Zielgrößen bzw. Restriktionen auf 0 gesetzt. Dies bezieht sich jedoch nur auf die Restriktionen für die Gesamtauswahl. Die bestandesspezifischen Restriktionen wie BT (Bestandestyp) und Alter der Hauptschicht werden bereits bei der Vorauswahl potentieller Bestände berücksichtigt. Dies führt zu einer Verringerung des Suchraums und einer schnelleren Optimabestimmung. Die für die gesamte Bestandesauswahl geltenden Restriktionen können vom Anwender aktiviert und entsprechend definiert werden. Die Implementierung der *VariationFunction* gestaltet sich in diesem Fall sehr einfach. Es muss lediglich an einer oder mehreren zufälligen Stellen des Parameter-Arrays der vorhandene Wert invertiert werden. Es wird also eine 1 gegen eine 0 getauscht und umgekehrt. Da dem

parallelen Simulated Annealing zwei verschiedene Variationsfunktionen zur Verbesserung der Intensivierung und Diversifikation des Suchprozesses übergeben werden können, wird einmal eine Variationsfunktion implementiert die marginale Änderungen an der aktuellen Lösung vornimmt, also eine intensive Suche bewirkt. Die zweite Variationsfunktion ändert zufällig für bis zu fünf Prozent der potentiellen Bestände den Status (ausgewählt = 1, nicht ausgewählt = 0) und führt zu einer Diversifikation der Suche.

			G		T	G		T
Gesamtnutzen	∑		0.5	Hotspot-Eigenschaften		n.d.	Wirtschaftliche Einbußen (min)	linear
						n.d.	Totholz (max)	linear
						n.d.	Laubholzanteil (max)	linear
			0.5.	Aggregation (max)	linear			
	Bestand			Alter der Hauptschicht, BT (wird bei Vorauswahl berücksichtigt)				
Restriktionen	Bestandesauswahl	∑		Hotspot-Größe [ha]				
				Mittlerer Laubholzanteil [%]				
				Mittlerer Totholzvorrat [Vfm ha^{-1}]				
				Mindestflächenanteil mit def. Hangneigung/Exposition [%]				
				Gesamtfläche (nutzerdefinierte Flächengröße aller ausgewählten Bestände) [ha]				

Abb. 52: Aufbau des Zielsystems im Kontext der optimalen Flächenauswahl. G =Gewichtung, n.d. = wird durch Nutzerinteraktion definiert, T = Transformationstyp.

Nach der Definition des Zielsystems und den vorbereitenden Berechnungen der bestandesweisen relevanten Parameter, wird die Optimumsuche gestartet. Hierzu wird das Verfahren PSA (Paralleles Simulated Annealing) verwendet. Nach Abschluss der Optimumsuche wird das Ergebnis tabellarisch und kartographisch dargestellt. Die Tabelle beinhaltet die IDs der ausgewählten Bestände und die aggregierten Parameter des Zielsystems. Die erstellte Karte hebt die ausgewählten Bestände in der aktuellen Darstellung farblich hervor. Insgesamt ist die Struktur dieser Optimierungskomponente der in Kap. 4.6.2 dargestellten Komponente sehr ähnlich. Der einzige Unterschied im Verfahrensablauf besteht darin, dass keine optimale Pfadkombination sondern eine optimale Bestandesauswahl vorgenommen wird.

5 Anwendungsbeispiel

Im Folgenden werden die Funktionalität des vorgestellten DSS anhand eines konkreten Beispiels demonstriert und die Ergebnisse diskutiert. Neben der Qualität der Ergebnisse wird auch die benötigte Rechenzeit betrachtet, welche maßgeblich die Nutzerfreundlichkeit bzw. die Praxistauglichkeit der Software beeinflusst. Das Anwendungsbeispiel richtet sich nach dem dreistufigen Aufbau der Entscheidungsunterstützung. Es werden zunächst der Status-Quo der zugrunde liegenden Forsteinrichtungsdaten aufbereitet und verschiedene Indikatoren zur Entscheidungsunterstützung abgeleitet. Im Rahmen der zweiten Entscheidungsunterstützungs-Stufe (Variantenvergleich) werden vier waldbauliche Varianten simuliert und anhand ausgewählter Indikatoren verglichen. In der dritten Stufe werden auf Basis des vorgestellten Optimierungssystems mehrere Handlungsalternativen generiert. Alle Berechnungen werden auf einem Multiprozessor-PC (CPUs: 2 x Intel Core2 Duo 6700 mit 2,66 Ghz, 4GB RAM) durchgeführt.

5.1 Untersuchungsgebiet

Die Gesamtfläche des Untersuchungsgebietes umfasst ca. 15.359 ha Waldfläche. Diese verteilt sich auf 4.001 forstliche Gliederungseinheiten (Bestände), die zu den niedersächsischen Forstämtern Clausthal (10.584 ha) und Riefensbeek (ca. 137 ha) sowie zum Nationalpark Harz (4.638 ha) gehören. Die durchschnittliche Bestandesgröße beträgt 3,8 ha.

Das Untersuchungsgebiet gehört zu den forstlichen Wuchsgebieten Harz und Berglandschwelle, wobei mehr als die Hälfte der Waldflächen dem Wuchsbezirk Montaner Mittel- und Oberharz zuzuordnen ist, der sich über die Höhenstufe von 475 – 700 m üNN erstreckt. Die Wuchsbezirke Westlicher und Südwestlicher Harzrand (300 – 475 m üNN) sowie Harzhochlage/Hochharz (ober- und hochmontane Höhenstufe > 650/700 m üNN) sind mit etwa den gleichen Anteilen von ca. einem Fünftel der Fläche des Untersuchungsgebietes vertreten. Die Datengrundlage besteht aus der Forsteinrichtung der Niedersächsischen Landesforsten aus dem Jahre 2003 und den entsprechenden Geodaten.

5.2 Status quo und entscheidungsrelevante Indikatoren

Auf der ersten Stufe zur Unterstützung waldbaulicher Entscheidungen bietet das System die Möglichkeit, den aktuellen Zustand einer Waldfläche anhand der Berechnung verschiedener Indikatoren zu beschreiben und dadurch eine erste Entscheidungsgrundlage zu schaffen. Um die einzelnen entscheidungsrelevanten Indikatoren berechnen zu können, müssen aus den für

das gewählte Beispiel vorliegenden Forsteinrichtungsdaten zunächst virtuelle Bestände (vgl. Abb. 54) aufgebaut werden.

Tab. 53: Rahmenwerte des Ausgangszustands

Status quo	Mittel	Max	Min
Vorrat [Vfm/ha]	283	677	0
G [m²/ha]	28	48	0
Stammzahl [n/ha]	443	3885	0
Artenanzahl [n/Bestand]	1,6	9	0
A-Wert	0,16	2,2	0
Ekf Abtriebswert [€/ha]	6.725	29.552	-178
BT-Durchmischung	0,352	1	0
Pflegedringlichkeit	0,41	1	0
Endnutzungs-dringlichkeit	0,1	0,89	0

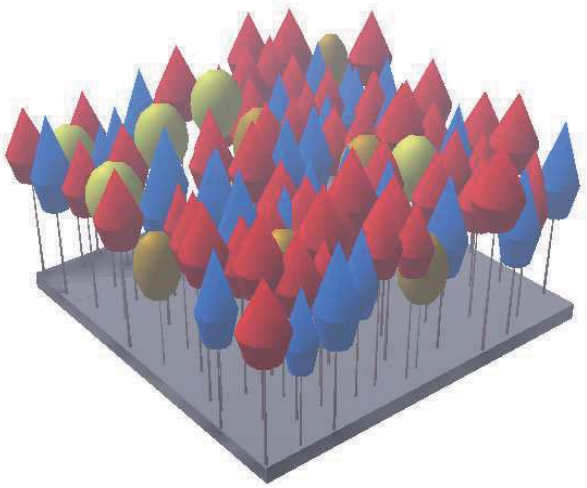


Abb. 54: Ein generierter Modellbestand, 0.2 ha mit Lärche (rot), Fichte (blau), Eiche (gelb) und Buche (braun).

Die vom System generierten Einzelbaumdaten werden in der Zieldatenbank abgespeichert, so dass jederzeit auf diesen Ausgangszustand zurückgegriffen werden kann und darauf aufbauend verschiedene Simulationen durchgeführt werden können. Die Simulationsergebnisse werden in der selben Datenstruktur, aber in extra Tabellen, abgelegt. Somit ist es möglich, verschiedene Entwicklungspfade aufzubauen. Für die insgesamt 4001 generierten Bestände ergeben sich auf Ebene des Untersuchungsgebiets die in Tab. 53 dargestellten Indikatoren. Die Indikatoren können auch für Untereinheiten bis hin zum einzelnen Bestand berechnet werden und im Rahmen der Planung von Maßnahmen auf Ebene der entsprechenden Untereinheit oder des Einzelbestands verwendet werden. Der mittlere Vorrat liegt bei 283 Vfm je Hektar. Die über alle Modellbestände gemittelte Grundfläche beläuft sich auf 28 m². Die Stammzahl je Bestand beträgt im Mittel ca. 443 Bäume je Hektar. Durchschnittlich kommen 1,6 Arten je Bestand vor. Das Maximum liegt bei 9 Arten je Bestand. Der A-Wert liegt in einem Wertebereich zwischen 0 und 2,2. Der Mittelwert beträgt 0,16. Da für den Status quo keine Eingriffsinformationen vorliegen, kann zur ökonomischen Bewertung lediglich der erntekostenfreie Abtriebswert herangezogen werden. Dieser liegt im Mittel bei 6725 € ha^{-1}. Bezogen auf das Derbholzvolumen dominieren die Baumartengruppe Fichte mit einem Anteil von ca. 87 Prozent bzw. 3,9 Millionen Vorratsfestmetern (Vfm). Mit einem Anteil von 10,5 Prozent (ca. 473.000 Vfm) folgt die Baumartengruppe Buche (vgl.

Abb. 55 links). Bezieht man die Baumartenflächen (ideelle Teilflächen auf Basis der modellierten Kronenschirmflächen) auf die gesamte Waldfläche des Untersuchungsgebietes, zeigt sich ein sehr ähnliches Bild. Die Baumartengruppe Fichte dominiert wiederum mit über 80 Prozent vor der Buche, die 10 Prozent der Fläche einnimmt. Die unter „Sonstige“ zusammengefassten Baumartengruppen (Eiche, anderes Laubholz mit hoher Umtriebszeit (ALh), anderes Laubholz mit niedriger Umtriebszeit (ALn), Douglasie, Kiefer, Lärche) nehmen 8 Prozent der Fläche ein (Abb. 55 rechts).

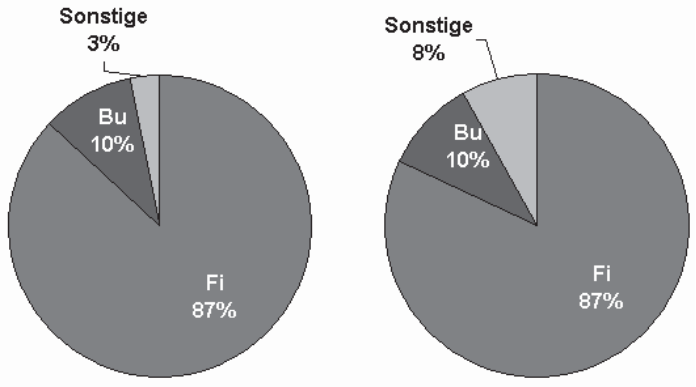


Abb. 55: Vorratsverteilung (links) und Flächenverteilung (rechts) nach Baumartengruppen.

5.2.1 BT-Durchmischung

In Abb. 56 (links) wird die räumliche Verteilung der Baumarten im Untersuchungsgebiet auf der Basis von Bestandestypengruppen dargestellt. Wie die Baumartenverteilung schon andeutet, wird der größte Teil der Bestände durch die Baumart Fichte (blau) geprägt. Lediglich im Nordosten des Untersuchungsgebiets sind gehäuft von Laubholz (vornehmlich Buche, braun) geprägte Bestände anzutreffen. Rechts ist die Betriebstypendurchmischung dargestellt. Es fällt auf, dass im Nordosten des Waldgebietes eine deutlich stärkere Betriebstypendurchmischung vorliegt. Der größte, fichtendominierte Flächenanteil zeigt eine eher geringe Durchmischung. Der entsprechende Index liegt bei 0,35. Dies bedeutet, dass ca. 35 Prozent der maximal möglichen Durchmischung erreicht werden. Über die Betriebstypen-Durchmischung können u. a. Rückschlüsse auf den Erholungswert des Gebiets getroffen werden. Ist die Durchmischung hoch, kann gefolgert werden, dass der Abwechslungsreichtum die Attraktivität des Gebiets als Erholungsziel steigert. Zu bedenken ist jedoch, dass auch ein abwechslungsreiches Relief oder die Durchschneidung mit Gewässern den Erholungswert einer Landschaft erhöhen können. Größere Gewässer werden bei der Berechnung der BT-

Durchmischung bereits als „Bestand mit anderem BT“ berücksichtigt und erhöhen den Indexwert. Die Berücksichtigung zusätzlicher Geodaten zu beispielsweise vorhandenen Fließgewässern, der Zugänglichkeit (Erschließung) oder der Zerschneidung (z. B. durch stark befahrene Straßen) könnte die Aussagekraft des Indexes bezüglich des Erholungswertes verbessern (NOHL 2001).

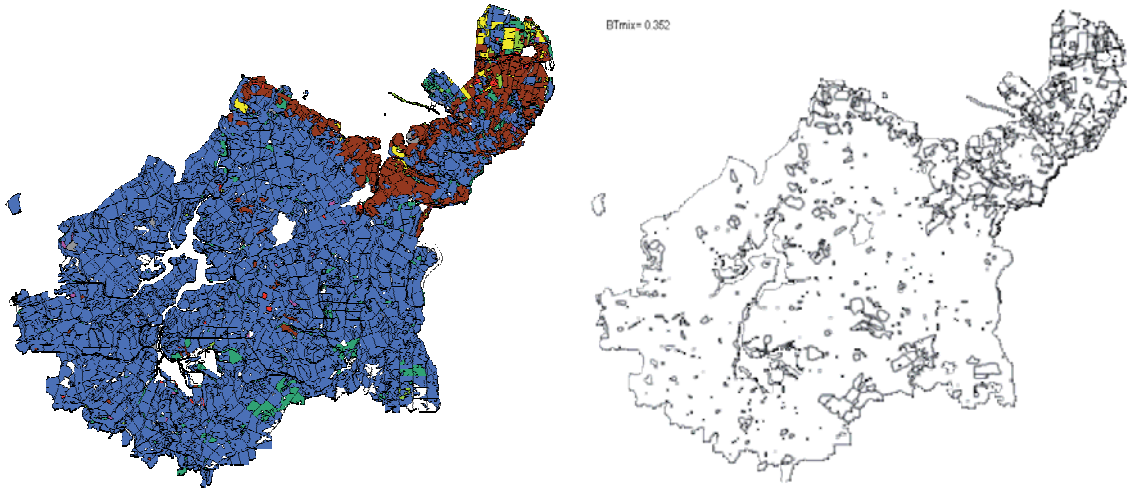


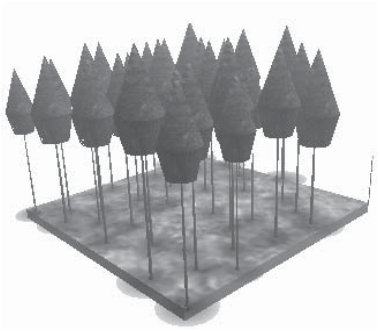
Abb. 56:Links: Gliederung des Untersuchungsgebiets in Planungseinheiten nach gruppierten Bestandestypen (blau = Fichte, braun= Buche, gelb = Eiche). Rechts: Rasterkarte zur Betriebstypendurchmischung. Je dunkler eine Rasterzelle eingefärbt ist, umso größer ist der Anteil benachbarter Zellen mit unterschiedlichem BT.

Zur Beurteilung der räumlichen (Baumarten-) Diversität ist der beschriebene Indikator gut geeignet. Neben der Baumartenzusammensetzung (über den BT) kann der Indikator für jedes beliebige Bestandesattribut, wie etwa die dominierende Altersklasse, die Verjüngungssituation (Verjüngung vorhanden ja/nein) oder die Bestandesdichte berechnet werden.

5.2.2 Handlungsdringlichkeit

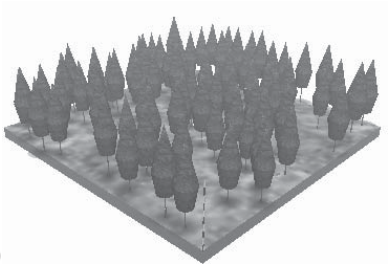
Ein im Rahmen dieser Arbeit vorgestellter Indikator zu Bewertung des Status quo oder eines simulierten Entwicklungspfads, ist die sog. Handlungsdringlichkeit. Dieser Indikator setzt sich aus zwei Teilindikatoren zur Beschreibung der Dringlichkeit von Pflegemaßnahmen und der Dringlichkeit einer Endnutzung zusammen. Diese beiden Indikatoren werden automatisch nach dem Generieren des Status quo sowie jedem Simulationslauf einzelbestandesweise berechnet und in eine entsprechende Datenbanktabelle gespeichert. Für den Status quo des vorliegenden Untersuchungsgebiets errechnen sich eine Pflegedringlichkeit von 0,41 und eine Endnutzungsdringlichkeit von 0,1. Der Vorteil dieser Indikatoren liegt in der Mischbestandsfähigkeit. Bei der Berechnung der Indikatoren werden u. a. die

Konkurrenzsituation bzw. der Zuwachs der einzelnen Bäume eines Modellbestands berücksichtigt. Dabei wird nach Zukunftsbäumen und Füllbestand differenziert. Die Zukunftsbäume erhalten höhere Gewichte bei der Berechnung des Indikators auf Ebene des Bestandes. In Abb. 57 sind beispielhaft drei Modellbestände und die jeweils dazugehörige Pflege- bzw. Eingriffsdringlichkeit dargestellt.



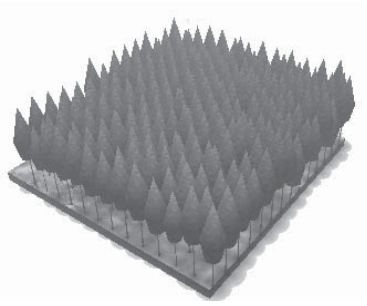
(a)

Vorrat: 412 Vfm
dg: 51,8 cm
G: 29,4 m²
Endnutzungsdringlichkeit: 0.63
Pflegedringlichkeit: 0.01



(b)

Vorrat: 68 Vfm
dg: 21,6 cm
G: 11,1 m²
Endnutzungsdringlichkeit 0,0
Pflegedringlichkeit 0,2



(c)

Vorrat: 308 Vfm
dg: 24,99 cm
G: 34,2 m²
Endnutzungsdringlichkeit 0,0
Pflegedringlichkeit 0,65

Abb. 57: Drei Modellbestände mit unterschiedlich starker Pflege- bzw. Eingriffsdringlichkeit.

Die Endnutzungsdringlichkeit reicht hier von 0,63 bis 0,0. Die Pflegedringlichkeit von 0,65 bis 0,01. Bestand (a) bekommt eine Eingriffsdringlichkeit von 0,63 zugewiesen. Der dg von 51,8 cm deutet darauf hin, dass ein hoher Vorratsanteil des Fichtenbestandes Durchmesserklassen über der Zielstärke von 45 cm zuzuordnen ist. Ca. 88 Prozent des Vorrats werden von Bäumen mit einem BHD größer Zielstärke gebildet. Die Bestände (b) und (c) weisen eine Endnutzungsdringlichkeit von 0,0 auf, da hier keine zielstarken Bäume vorhanden sind. Die Pflegedringlichkeit ist bei Bestand (b) ebenfalls gering (0,2). Dies ist schon bei Betrachtung des visualisierten Modellbestands zu vermuten. Es ist eine lichte Bestandesstruktur zu erkennen, die auf einen insgesamt geringen Konkurrenzdruck hindeutet. Bestand (c) ist hingegen relativ stark pflegedringlich. Der Indikator liegt bei 0,65, was bei Betrachtung der dichten Bestandesstruktur auch plausibel erscheint. Bei der Methodik zur

Berechnung der Pflegedringlichkeit wird die Konkurrenz und bei vorhandenen Informationen die angestrebte Baumartenverteilung berücksichtigt. Ein Nachteil dieser Methode liegt darin, dass mit abnehmender Konkurrenzsituation auch die Pflegedringlichkeit geringer wird. Stocken keine Bäume mit einem BHD gleich oder größer 7 cm (Derbholzgrenze) auf dem Bestand, wird eine Pflegedringlichkeit von 0 berechnet. Da eine vorhandene Verjüngungssituation z. Z. von dem vorgestellten System nicht berücksichtigt wird, könnte es in diesem Fall sein, dass aktiv eine Pflanzung durchgeführt werden müsste, um einen neuen Bestand zu begründen, und dementsprechend sehr wohl Handlungsbedarf auf der Fläche besteht. Auch bei einer sehr geringen Bestockung wird ein geringer Pflegebedarf berechnet und folglich kein Handlungsbedarf ausgewiesen. Es könnte jedoch notwendig sein, eine Nachbesserung durchzuführen, um eine gewünschte Zielbestockung zu erreichen.

Bei der Berechnung der Endnutzungsdringlichkeit wird neben der statischen Durchmesserstruktur noch eine dynamische Größe bezüglich der zu erwartenden Zuwächse berücksichtigt. Dadurch wird erreicht, dass die Endnutzungsdringlichkeit erhöht wird, wenn vergleichsweise nur noch geringe Zuwächse von den zielstarken Bäumen geleistet werden. Die Integration weiterer „Strafterme“ könnte ein zu langes Hinauszögern der Endnutzung noch stärker berücksichtigen. Denkbar wäre die Berücksichtigung von Qualitätsmodellen, die beispielsweise für die Baumart Buche eine mögliche Entwertung durch Verkernung beschreiben (SCHMIDT et al. 2008). Wichtig für das vorgestellte System ist die Relation der Indikatorwerte zueinander, weniger die absolute Aussagekraft. Mittels der Indikatoren können zwei oder mehr Bestände untereinander, größere Auswertungseinheiten (Revierförsterei) oder verschiedenen Wandentwicklungsszenarien verglichen werden. Auf Basis dieses Vergleichs kann dann eine Entscheidung z. B. darüber getroffen werden, wo zuerst Maßnahmen ergriffen werden müssen.

5.2.3 Risiko von Sturmschäden

Da in das vorgestellte DSS eine Risikokomponente zur Abschätzung der Wahrscheinlichkeit von Sturmschäden integriert wurde, kann unter Einbeziehung eines DGMs einzelbestandesweise für jede vertretene Baumart und Bestandesschicht der zu erwartende geschädigte Vorratsanteil bei definierten Sturmeigenschaften abgeschätzt werden. Die Sturmeigenschaften werden über die Parameter *Richtung* und *Stärke* beschrieben, welche vom Anwender spezifiziert werden müssen. Neben der absoluten Aussage (mit wie viel Schadholz ist zu rechnen?) kann aber vor allem auch im bestandesweisen Vergleich abgeschätzt werden,

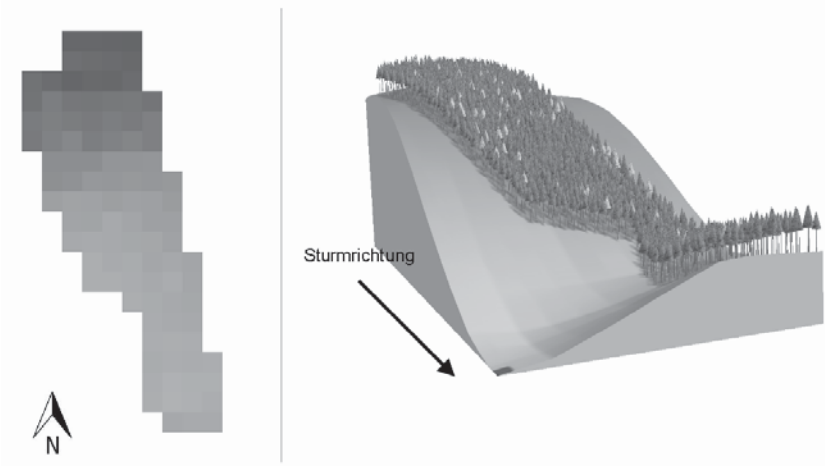
wo das Risiko für Sturmschäden am höchsten ist und somit auch präventive Maßnahmen oder Nutzungen am dringlichsten sind. Auf Basis des Modells kann ebenfalls das generelle Risiko für eine bestimmte Baumart abgeschätzt werden. Hierzu wird das Modell nicht auf die tatsächlichen Bestandesdaten (dg, hg) angewendet, sondern auf eine konstantes, fiktives Durchmessesser-Höhen-Tupel. Das Ergebnis dieser Berechnung zeigt vor allem in Abhängigkeit der Geländesituation, wo für eine bestimmte Baumart das Windwurfrisiko am höchsten ist. Diese Information kann beispielsweise bei Wahl der Zielbestockung (WET = Waldentwicklungstyp) eine entscheidungsrelevante Grundlage darstellen.

Abb. 58 zeigt eine Risikokarte für die Baumart Fichte auf Grundlage einer für alle Rasterzellen konstanten dg-hg-Paarung von 37 und 33.

Abb. 58: Windwurfrisikokarte für eine fiktive Fichtenbestockung, je dunkler eine Rasterzelle eingefärbt ist, umso geringer ist das Sturmschadenrisiko.

Je heller eine Rasterzelle eingefärbt ist, umso höher ist das Windwurfrisiko. Bei der Berechnung der dargestellten Karte wurde ein starker Sturm aus nördlicher Richtung unterstellt. Das Minimum beträgt 0.9 Prozent, das Maximum 99.8 Prozent. Im Mittel wird ein Windwurfrisiko von 51 Prozent ausgewiesen. Neben der Risikokarte für eine einzelne Baumart, kann das Windwurfrisiko und darauf aufbauend der Anteil des zu erwartenden Schadholzvorrates bestandesweise für eine reale oder simulierte Bestockung berechnet werden. In Abb. 59 ist die rasterbasierte Risikokarte (links) und eine 3D-Darstellung des Bestandes unter Berücksichtigung der Geländehöhe (rechts) dargestellt. Die Bestockung setzt sich aus Fichte (270 VFm) und Lärche (16,7 VFm) zusammen, wobei der dg bei 25,3 cm (Fi) bzw. 27,6 cm (ELä) liegt. Die hg beträgt 17,9 m bei der Fichte und 20,6 m bei der Lärche. Der

ausgewählte Bestand zieht sich von einer Kuppe auf der Lee-Seite hangabwärts bis in ein Tal. Entsprechend der unterschiedlichen Geländeexposition zeigt die Risikokarte erwartungsgemäß im Bereich der Kuppe das höchste Windwurfrisiko an, welches talwärts auf dem südexponierten Hang abnimmt. Die Berechnung des Schadholzanteils ergibt, dass bei den unterstellten Bedingungen 26 Prozent des Vorrats geworfen werden.



Art	Alter	Schicht	dg	hg	G	Vfm	N_ha
ELae (811)	46	1	27,641	20,598	1,814	16,717	30
Fi (511)	43	1	25,323	17,873	32,586	270,014	635

Abb. 59: Windwurfrisiko für einen konkreten Bestand, links: rasterbasierte Risikokarte, rechts:3D-Visialisierung des Bestands unter Berücksichtigung der Geländehöhe, unten: zugehörige Forsteinrichtungszeilen.

Risikomanagement (Risikoanalyse, Risikohandhabung, Kontrolle des Risikomanagements) ist für die Forstwirtschaft u. a. aufgrund der langen Produktionszeiträume eine wichtige Managementdisziplin (HOLTHAUSEN et al. 2004). Die Kombination der Handlungsdringlichkeit und der Wahrscheinlichkeit von Sturmschäden, liefert eine hilfreiche Information zum Aufbau einer bestandesweisen Rangfolge für waldbauliche Maßnahmen. Es kann modellhaft die Dringlichkeit ermittelt werden, in einem Bestand eingreifen zu müssen, um negative Auswirkungen aufgrund eines Pflegerückstandes, eines zu langen Hinauszögerns der Endnutzung und durch die Schäden eines möglichen Sturmereignisses abwenden zu können. Die Berechnung der bestandesweisen Handlungsdringlichkeit und des Sturmschadenrisikos deckt im Kontext der Risikoanalyse, welche sich in Risikoidentifizierung und Risikobewertung gliedert (HOLTHAUSEN et al. 2004), den Bereich der Risikobewertung ab. Da in dem vorgestellten System bislang nur die Eingriffsdringlichkeit in die automatisierte Generierung von Handlungsoptionen

(Optimierung) mit einbezogen wird, wird teilweise auch der zweite Schritt des Risikomanagements, die Risikohandhabung, berücksichtigt. Die Berücksichtigung der Wahrscheinlichkeit von Sturmschäden im Rahmen der Generierung von Handlungsalternativen, ist jedoch ohne weiteres umsetzbar, da hierfür alle technischen Vorraussetzungen (automatisierte Berechnung der Geländeexponiertheit, Verschneidung mit Bestandespolygonen) in dem vorgestellten System gegeben sind. Sowohl im Rahmen der Optimierung kurzfristiger Nutzungsentscheidungen (Kap. 5.4.2) als auch bei der mittelfristigen Ermittlung von optimalen Entwicklungspfaden (Kap. 5.4.1) wäre das Sturmschadenrisiko eine weitere sinnvolle Zielgröße. In Abhängigkeit der nutzerspezifischen Präferenzen könnten so Handlungsalternativen ermittelt werden, welche dazu beitragen, das gesamtbetriebliche Risiko des Auftretens von Sturmschäden zu minimieren.

5.3 Szenariosimulation/Pfadgenerierung

Im Rahmen des Anwendungsbeispiels werden vier Szenarien mit einer Simulationsdauer von jeweils 20 Jahren à 4 Perioden gerechnet. Die Szenarien *keine Eingriffe*, *starke Eingriffe*, *mittlere Eingriffe* und *schwache Eingriffe* werden über die Maßnahmenelemente des verwendeten Behandlungsmodells (Kap. 4.4.2) definiert und in Kombination mit dem Wuchsmodell TreeGrOSS in den Modellbeständen virtuell umgesetzt. Der Simulationszeitraum beträgt insgesamt 20 Jahre mit vier Eingriffsintervallen. Bei der Variante *keine Eingriffe* werden lediglich Zuwachs, Mortalität und Einwuchs bestandesweise für den definierten Zeitraum berechnet. Die wichtigsten Steuerparameter der übrigen Varianten sind Tab. 15 zu entnehmen. Die benötigte Rechenzeit je Variante (4001 Bestände 4 x 5 Jahre fortschreiben) liegt zwischen 24 und 28 Minuten. Die Entwicklung eines einzelnen Bestandes wird folglich durchschnittlich in ca. 0,39 Sekunden berechnet. Diese hohe Simulationsgeschwindigkeit ist vor allem auf die Parallelisierung der Simulation und die Optimierung des Lese- und Speichervorgangs (indizierte Tabellen, Batch-Update) zurückzuführen. Das von Duda (DUDA 2006) vorgestellte System benötigt beispielsweise 2 bis 3 Tage, um eine vergleichbare Simulation für ein Gebiet von ca. 12.000 Hektar durchzuführen.

Tab. 15: Steuerparameter der drei gerechneten Szenarien schwache, mittlere und starke Eingriffe.

	mittlere Eingriffe	starke Eingriffe	schwache Eingriffe
Schutz seltener Baumarten	ja	nein	ja
Habitatbäume (n ha)	1	0	3
Totholz (m^3 ha^{-1})	20	bis 10	40
Zielstärke (cm)	Ei 70, Bu 60 Fi 45, Ki 45	- 5 cm	+ 5 cm
Endnutzungsmasse pro Hieb (m^3 ha^{-1})	Min. 30 - 50 Max. 70 - 100	Min. 40 - 100 Max. 80 - 200	Min. 10 - 25 Max. 35 - 50 (Ei 100)
Durchforstungsbeginn			
Bestandesoberhöhe (m)	10 - 16	12 - 18	10 - 16
Durchforstungsmasse pro Hieb (m^3 ha^{-1})	25 - 70	35 - 90	15 - 35
Freistellungsgrad der Z-Bäume	stark	sehr stark	stark
Pflanzungen	ja	ja	nein

Die berechneten Szenarien weisen die in Tab. 16 dargestellten Rahmenwerte auf. Bei der Variante mit starken Eingriffen ist ein extremer Vorratsabbau zu verzeichnen. Nach 20 Jahren liegt der Vorrat nur noch bei 78 Vfm ha^{-1}. Bedingt durch die starken Eingriffe (364 Vfm ha^{-1}), fällt bei dieser Variante der mittlere jährliche Zuwachs am geringsten aus (8 Vfm $ha^{-1}a^{-1}$). Die höchsten mittleren Zuwächse erzielt die Variante *schwache Eingriffe* (12 Vfm $ha^{-1}a^{-1}$). Knapp darunter liegt der Zuwachs der moderaten Variante (11 Vfm $ha^{-1}a^{-1}$). Die bestandesbezogenen Diversitäts-Indikatoren (h, a) weisen teilweise voneinander abweichende Werte innerhalb einer Variante auf. Der mittlere h-Wert bei der Variante *starke Eingriffe* ist durch die starke Auflichtung der Bestände und eine entsprechende natürliche und gepflanzte Verjüngung vergleichsweise am höchsten. Das Artprofil (a) hingegen ist mit 0,57 am geringsten. Die Variante *mittlere Eingriffe* führt zu der höchsten Arten-Schichten-Diversität (0,7). Die BT-Durchmischung steigt mit zunehmender Eingriffsstärke. Dies ist dadurch zu erklären, dass auf ca. einem Viertel des betrachteten Waldgebiets Laubholzverjüngung bzw. -voranbauten vorhanden sind. Je stärker die Eingriffe ausfallen, umso eher wird der BT durch die Verjüngungsschichten bestimmt. Die BT-Zusammensetzung ändert sich dementsprechend dahingehend, dass immer mehr laubholzdominierte Bestandestypen auftreten und die BT-Durchmischung zunimmt. Einem ähnlichen Trend unterliegt erwartungsgemäß die Pflege- und Nutzungsdringlichkeit. Je stärker die Eingriffe ausfallen, umso geringer ist die Eingriffsdringlichkeit.

Tab. 16: Rahmenwerte für die vier simulierten Szenarien.

	Keine Eingriffe	Starke Eingriffe	Mittlere Eingriffe	Schwache Eingriffe
Vorrat [Vfm/ha]	511,8	78,3	310,0	423,1
G [m²/ha]	46,1	12,5	30,6	39,7
Zuwachs [Vfm $ha^{-1}a^{-1}$]	12,0	8,2	11,4	11,8
Habitatbäume [n ha^{-1}]	0	0	0,9	10,9
h-Wert	0,26	0,34	0,31	0,29
a-Wert	0,63	0,57	0,70	0,65
Totholzneubildung [Vfm ha^{-1}]	12	6	11	21
Artanzahl	2,5	3,2	2,7	2,7
Durchforstung [Vfm $ha^{-1}20a^{-1}$] / [Efm $ha^{-1}a^{-1}$]	0	5/0.2	55/2,2	17,8/0,7
Endnutzung [Vfm $ha^{-1}20a^{-1}$] / [Efm $ha^{-1}a^{-1}$]	0	359/14,4	135/5,4	56,6/2,7
Erlös [€ $ha^{-1}20a^{-1}$]*	0	13.128,1	7.057,0	2851,1
Kosten [€ $ha^{-1}20a^{-1}$]*	0	3.552,8	1.721,4	624,5
Abtriebswert 2023 [€]	13.193,6	760,0	7.438,5	10.542,6
BT-Durchmischung	0,38	0,52	0,46	0,45
Eingriffsdringlichkeit (Pflege/Endnutzung)	0,61/0,44	0,28/0,05	0,34/0,11	0,47/0,19

*Die Werte sind ohne Verzinsung hergeleitet. Eine Verzinsung erfolgt erst bei der Berechnung des *oets* im Rahmen der Optimierung.

Das Anwendungsbeispiel zeigt, dass die Simulationskomponente des Systems eine weitere Grundlage für die Entscheidung waldbaulicher Managementfragen darstellt. Es können verschiedene waldbauliche Szenarien definiert, berechnet und anhand ausgewählter Parameter verglichen werden. Somit können Auswirkungen verschiedener Handlungsoptionen modellhaft überprüft und eventuelle negative Folgen auf das reale System vermieden werden. Die verfügbaren entscheidungsrelevanten Indikatoren decken alle Waldfunktionen direkt oder indirekt ab, so dass auch eine multifunktional ausgerichtete Forstwirtschaft abgebildet werden kann. Werden viele Parameter in den Vergleich mit einbezogen, kann dieser auch durch ein geeignetes Verfahren systematisch durchgeführt werden (vgl. ALBERT u. HANSEN 2007). Erst die mittel- bzw. längerfristige Simulation (bis 30 Jahre) waldbaulicher Szenarien ermöglicht es auf Betriebsebene Auswirkungen verschiedener Behandlungskonzepte zu überprüfen (PRETZSCH 2001b). Die aus der Simulation gewonnenen Erkenntnisse stellen dann im Rahmen der Entscheidungsunterstützung eine Grundlage zum Überdenken und Adjustieren geplanter Behandlungskonzepte dar.

Das beschriebene System verarbeitet jeden einzelnen Bestand bzw. Stichprobenpunkt des jeweiligen Untersuchungsgebiets. Eine andere Möglichkeit besteht darin, Bestände nach definierten Kriterien zu stratifizieren (PRETZSCH 2001b) und lediglich je Stratum einen stellvertretenden Modellbestand zu simulieren. Dies hat den Vorteil, dass die benötige

Gesamtrechenzeit deutlich herabgesetzt wird. Ein Nachteil ist jedoch, dass dadurch ein Verlust, je nach Definition der Straten, der zeitlichen und vor allem der räumlichen Auflösung des Simulationssystems einhergeht. Sollen je Bestand im simulierten Zeitraum zu verschiedenen Zeitpunkten modellhaft Eingriffe durchgeführt werden, oder sollen in die Simulation Modelle integriert werden, welche räumliche Muster berücksichtigen, muss eine bestandesscharfe Simulation erfolgen. Die schnelle Rechenzeit aufgrund der Parallelisierung und Optimierung der Lese- und Schreibvorgänge ermöglicht es, mit dem vorliegenden System problemlos auf Betriebsebene bestandesscharf Simulationsrechnungen durchzuführen. Dies ist eine Vorraussetzung, um beispielsweise das Sturmschadenmodell anzuwenden oder eine bestandesspezifische WET-Vorgabe bei der Simulation zu berücksichtigen. Dies bietet vor allem auch in Kombination mit dem Optimierungssystem den Vorteil, dass bestandesweise Handlungsoptionen (Entwicklungspfade) ermittelt werden können und mit dem Ergebnis der Optimierung direkt konkrete Handlungsempfehlungen vorliegen und diese nicht erst aus den Ergebnissen einer stratenweisen Optimierung abgeleitet werden müssen (SÁNCHEZ OROIS 2003).

Die in das vorgestellte System integrierte GIS-Komponente, das Bestandesvisualisierungsmodul, sowie die verschiedenen tabellarischen und graphischen Auswertungsmöglichkeiten unterstützen den Anwender dabei,die Ergebnisse der Simulation aufzubereiten und aussagekräftig darzustellen. Die Anwendung weiterer Softwarepakete (GIS, Tabellenkalkulation, Datenbankeditor) entfällt somit. Dies erhöht die Praxistauglichkeit des Systems. Vor allem die Kopplung mit einem externen GIS setzt fundierte Softwarekenntnisse voraus.

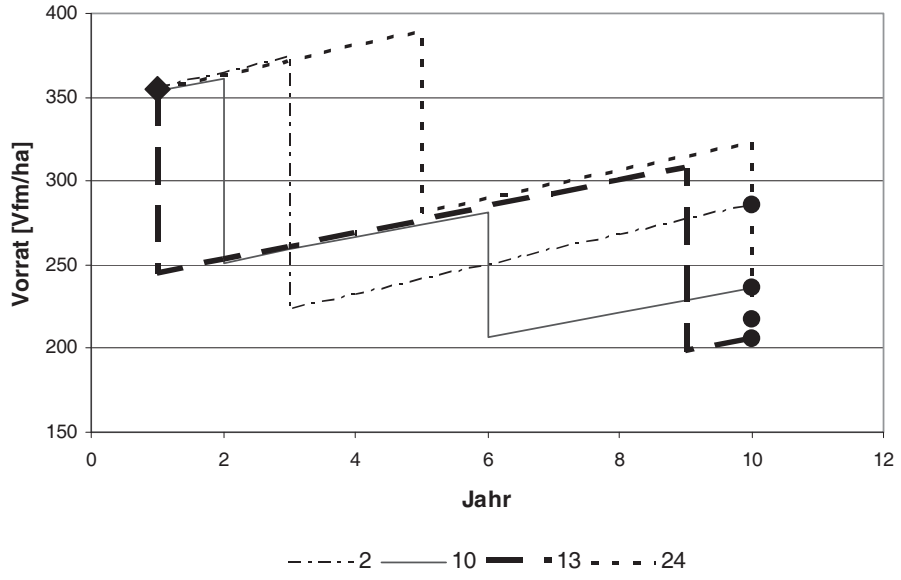
5.4 Optimierung

5.4.1 Optimierung der Eingriffsplanung

Dieser Optimierungsansatz soll für einen klassischen 10-jährigen Forsteinrichtungsturnus die Eingriffszeitpunkte (welches Jahr?) und die Eingriffsstärke der Pflege- bzw. Erntemaßnahmen bestandesweise so ermitteln, dass keine definierten Restriktionen verletzt werden und die nutzerspezifische Zielfunktion maximiert wird. Restriktionen sind die maximale u. minimale Gesamteingriffsmasse in 10 Jahren und die jährlichen maximalen und minimalen Eingriffsmassen. Die bestandesweise Zuweisung von Eingriffsstärke und Zeitpunkt wird in diesem Kontext als Entwicklungspfad verstanden. Die Zielfunktion setzt sich aus bis zu vier Teilnutzen zusammen. Die Gewichtung der Teilnutzen *konstante Eingriffsmassen* (*ef*),

Aggregation der jährlichen Eingriffe (*agg*), *ökonomischer Erfolg* (*oet*) und *Reduktion der Eingriffsdringlichkeit* (*pd*) ist nutzerspezifisch und wird durch einen Paarvergleich ermittelt oder direkt eingegeben. Das Verfahren liefert als Ergebnis eine Tabelle, welche für jeden Bestand des betrachteten Gebietes die Entnahmemenge und das Jahr des oder der vorgeschlagenen Eingriffe beinhaltet.

In Abb. 60 (oben) sind beispielhaft für nur einen Bestand (ID = 2) vier gültige Entwicklungspfade (Pfad 2, 10, 13 und 24, Kodierung vgl. Kap. 4.6.2) und die entsprechende Zeile der Ergebnistabelle (unten) mit den Daten des optimalen Pfads dargestellt.



ID	1. Eingriff	2. Eingriff	e1 [Vfm/ha]	e2 [Vfm/ha]	Endnutzung	*oet*	*pd*
2	Jahr 1	Jahr 9	110	110	Ja/Ja	7193,4	0,51

Abb. 60: Vier ausgewählte Entwicklungspfade (oben) zu Bestand 2 und die tabellarische Ausgabe(unten) des im Rahmen der nur auf den ökonomischen Erfolg ausgerichteten Optimierung.

Die fett hervorgehobene Linie zeigt den für Bestand 2 im Rahmen eines nur ertragsorientierten Optimierungslaufes (ökonomischer Erfolg) als optimal ausgewählten Pfad. Dieser sieht vor, dass in zwei Eingriffen (e1 und e2) jeweils ca. 110 Vfm entnommen werden. Die optimalen Eingriffszeitpunkte sind Jahr eins und Jahr neun (Abb. 60 unten). Im Rahmen des hier vorgestellten Anwendungsbeispiels wird das Verfahren mit sechs verschiedenen Gewichtungsszenarien auf das Untersuchungsgebiet angewendet. Vier der durchgeführten Optimierungsläufe werden mit einer Gewichtung der Zielfunktion durchgeführt, die jeweils nur einen der vier Teilnutzen berücksichtigt. Diese Läufe dienen vor allem der Validierung des Optimierungssystems. Zu den Läufen, die jeweils nur den Teilnutzen *pd* und *oet*

berücksichtigen, können die möglichen Maxima und Minima aus den einzelnen Extremwerten der Bestandesparameter durch vollständige Enumeration berechnet und zur Beurteilung des Lösungsverhaltens des Verfahrens verwendet werden. Durch die Normierung der Teilnutzen *ef* und *agg* ist zumindest der theoretisch mögliche Maximalwert bekannt. Bei den verbleibenden zwei Optimierungsläufen werden zwei (*ef* und *oet*) bzw. alle Teilnutzen gleich stark gewichtet berücksichtigt (Tab. 17). Die sechs verschiedenen Gewichtungsszenarien werden einmal ohne und einmal mit Einschränkung der jährlichen und der gesamten Eingriffsmassen gerechnet. Dadurch kann gezeigt werden, wie groß der Nutzenentgang durch die Einhaltung von Restriktion ausfällt. Bei den Läufen mit Restriktion wird die minimale jährliche Eingriffsmenge auf 2 Efm $ha^{-1}a^{-1}$ festgelegt.

Tab. 17: Gewichtung der Teilnutzen für die durchgeführten Optimierungsläufe A, B, C, D, E u. F.

Lauf	**ef**	**agg**	**oet**	**pd**
A	1	0	0	0
B	0	1	0	0
C	0	0	1	0
D	0	0	0	1
E	0,5	0	0,5	0
F	0,25	0,25	0,25	0,25

Die minimale Eingriffsmenge des gesamten Simulationszeitraums beträgt 25 Efm ha^{-1}. Maximal dürfen jährlich 7 Efm ha^{-1}, insgesamt 60 Efm ha^{-1}entnommen werden. Da bei diesem Optimierungsansatz vergleichsweise der größte Suchraum vorliegt, fällt auch die benötigte absolute Rechenzeit am höchsten aus. Die Generierung der Entwicklungspfade dauert ca. 25 Minuten. Der eigentliche Suchprozess läuft bei gegebener Hardwareausstattung zwischen 30 und 40 Sekunden.

Bei Lauf A (möglichst zeitlich gleichverteilte Nutzungsmassen) sind keine Auswirkungen der Restriktionen zu beobachten, da hier die jährliche Entnahmemenge zur Erreichung einer optimalen Gleichverteilung bei ca. 4,5 Efm liegt. Der Lauf ohne Restriktion liefert die gleichen Ergebnisse wie der Lauf mit eingeschränkter Nutzung. Es wird nahezu eine absolute Gleichverteilung erreicht ($ef > 0{,}999$), so dass jährlich 10 Prozent der Gesamtnutzungsmasse des berücksichtigten Zeitraums von 10 Jahren anfallen. Bei den übrigen Läufen ist eine Abnahme der Lösungsqualität durch die Einführung der Restriktionen zu beobachten (vgl. Tab. 18 und Tab. 19). Lauf A, C und D liefern ohne Restriktionen Lösungen nahe dem

globalen Optimum (oet, agg, pd > 0,999). Die Läufe mit Restriktion zeigen eine Abnahme der Lösungsqualität von 8 (D) bis 32 (B) Prozent. Da die Läufe A, C und D ohne limitierte Eingriffsmassen Lösungen nahe dem bekannten globalen Optimum erreichen, kann davon ausgegangen werden, dass das Verfahren stabil parametrisiert wurde und die entsprechenden Einbußen hinsichtlich der Lösungsqualität auf die Beschränkung der Eingriffsmengen zurückzuführen sind.

Lauf C (hohe räumliche Aggregation der jährlichen Nutzungen) erreicht ohne Restriktion ein Maximum von 0,46. Mit Einschränkung der Nutzung resultiert lediglich ein Maximum von 0,31. Zu beachten ist dabei, dass die Nutzungszeitpunkte aller Pfade einiger Bestände nicht alle Jahre abdecken. Es kann vorkommen, dass Bestände in einem oder mehreren Jahren des betrachteten Zeitraums modellhaft überhaupt nicht genutzt werden. Hinzu kommt, dass die Nutzungen z. T. zeitlich gekoppelt sind (vgl. Kap. 4.6.2) Dies kann u. U. dazu führen, dass der Suchalgorithmus eine Lösung ermittelt, welche nicht in allen Jahren zu einer hohen Aggregation der genutzten Bestände führt. In Abb. 61 sind die jährlich genutzten Flächen dunkel eingefärbt. Es ist zu beobachten, dass in den Jahren eins, drei, sechs und acht vergleichsweise nur geringe Aggregationen erzielt werden (0,05 bis 0,15). In den übrigen Jahren ist eine deutlich höhere Aggregation der Bestände mit Nutzung zu verzeichnen (0,35 bis 0,95). In den Läufen E und F werden zwei bzw. alle vier Teilnutzen zu gleichen Anteilen berücksichtigt. Lauf E führt zu einem Maximum von 0,9 bzw. 0,89 mit Einschränkung der Nutzung. Bei Einbeziehung aller Teilnutzen (Lauf F) wird ein Optimum von 0,75 bzw. 0,72 (Tab. 18 und Tab. 19) erreicht.

Neben einem Nutzenentgang (Verschlechterung des Zielfunktionswertes) durch Berücksichtigung von Restriktionen ist auch zu erwarten, dass bei Berücksichtigung mehrerer Teilnutzen diese im jeweiligen Vergleich zu den Läufen mit Berücksichtigung von nur einem Teilnutzen geringer ausfallen. Durch den Einsatz eines geeigneten Optimierungsverfahrens können gerade diese Opportunitätskosten möglichst minimiert werden. Es wird also der bestmögliche Kompromiss gefunden, wenn zwei oder mehr konkurrierende Teilnutzen optimiert werden sollen. In Tab. 18 und Tab. 19 ist der Nutzenentgang durch Berücksichtigung mehrerer Teilziele prozentual angegeben. Die prozentualen Einbußen beim Teilnutzen *ef* fallen insgesamt am geringsten aus (1 bis 5,9 Prozent.). Die größten Einbußen sind beim Teilnutzen *agg* zu verzeichnen (bis 30,4 Prozent). Insgesamt ist zu beobachten, dass die Opportunitätskosten erwartungsgemäß mit der Anzahl berücksichtigter Teilnutzen

höher werden und die Opportunitätskosten bei den Läufen mit Nutzungseinschränkung geringer ausfallen.

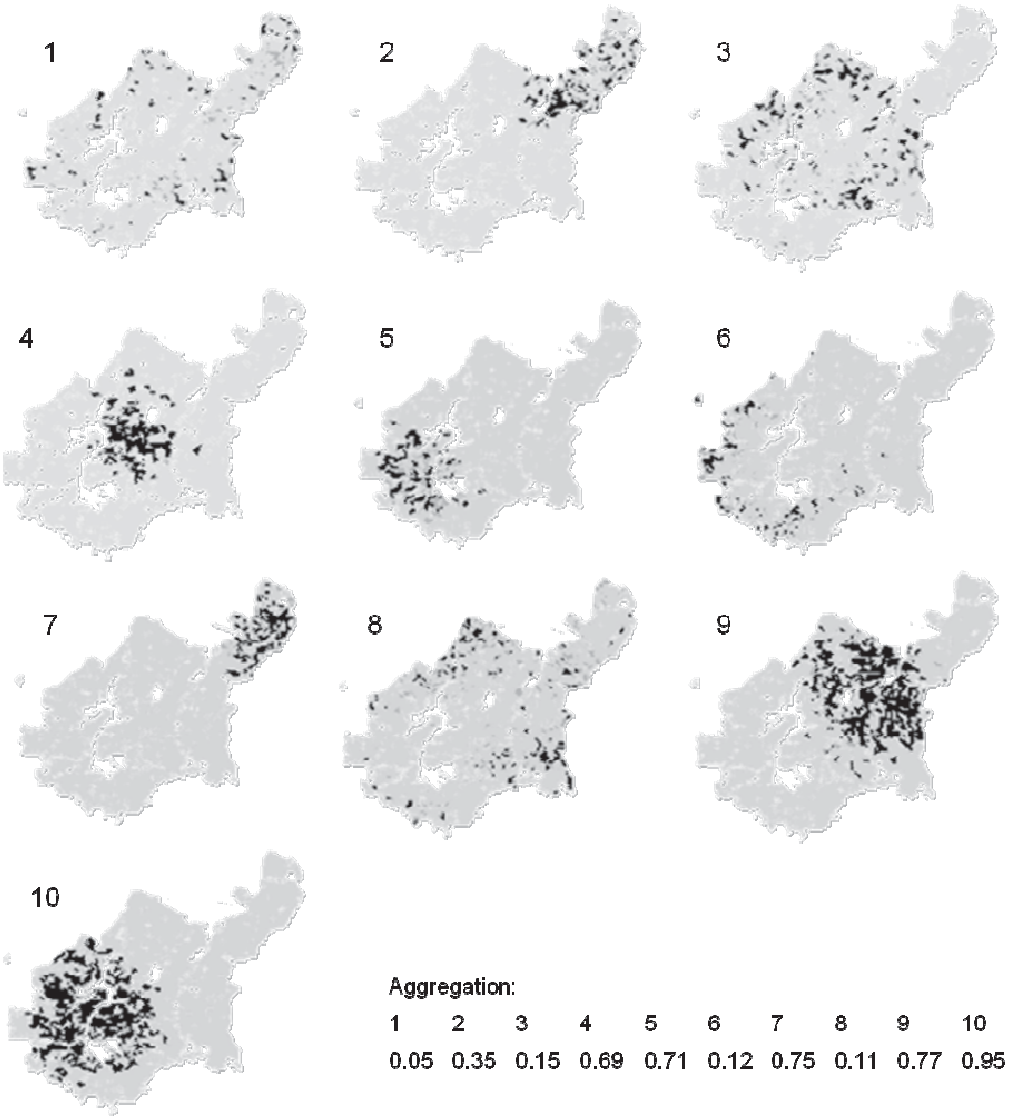


Abb. 61: Räumliche Verteilung der jährlich genutzten Flächen und der daraus resultierende Teilnutzen agg basierend auf Lauf B ohne Restriktion.

Tab. 18: Vergleich der erreichten Teilnutzen in den jeweils nur auf einen Teilnutzen ausgerichteten Läufen und den Läufen E und F (ohne Restriktion), sowie die je Teilnutzen resultierenden prozentualen Einbußen.

Teilnutzen	Teilnutzen aus jew. A, B, C, D	Teilnutzen E	Teilnutzen F
ef	>0.999	0,97 (-2,9%)	0,94 (-5,9 %)
oet	>0.999	0,83 (-16,9%)	0,86 (-13,9 %)
agg	0.46		0,32 (-30,4 %)
pd	>0,999		0,88 (-11,9%)
		0.9	**0.75**

Tab. 19: Vergleich der erreichten Teilnutzen in den jeweils nur auf einen Teilnutzen ausgerichteten Läufen und den Läufen E und F (mit Restriktion), sowie die je Teilnutzen resultierenden prozentualen Einbußen.

Teilnutzen	Teilnutzen aus jew. A, B, C, D	Teilnutzen E	Teilnutzen F
ef	>0.999	0.99 (-1%)	0,99 (-1 %)
oet	0,78	0,78 (-0 %)	0,77 (-1,3 %)
agg	0,31		0,27 (-12,9)
pd	0,92		0,86 (-6,5)
		0,89	**0,72**

5.4.2 Optimierung der kurzfristigen Holzbereitstellung

Im Rahmen der kurzfristigen Optimierung sollen Cluster von Beständen ausgewählt werden, welche definierte Sortimente (z. B. konkrete Käuferanfragen) zum aktuellen Zeitpunkt bereitstellen können und gegebene Restriktionen nicht verletzen. Restriktiv können z. B. eine zu steile Hangneigung einzelner Bestände oder ein Nutzungsverbot wirken. Ziel der Optimierung ist es, Bestände zu identifizieren, die die gewünschte Mindestmenge des oder der definierten Sortimente bereitstellen können und möglichst dicht beieinander liegen sowie gleichzeitig eine hohe Eingriffsdringlichkeit aufweisen. Für das konkrete Anwendungsbeispiel werden zwei Zielsortimente definiert, welche Buchenlangholz mit einem Mindestmittendurchmesser von 40cm bzw. 55cm und einem Zopf von 12cm vorsehen. Es werden drei Zielfunktions-Varianten berechnet: Berücksichtigung der Eingriffsdringlichkeit und der räumlichen Nähe (EDRN), Berücksichtigung nur der räumlichen Nähe (RN) und Berücksichtigung nur der Eingriffsdringlichkeit (ED). Abb. 62 zeigt die Optimierungsergebnisse für die Suchanfrage nach den definierten Zielsortimenten. Gesucht werden Bestände (dunkel eingefärbt), die insgesamt 2500 EFm des Zielsortiments unter Einhaltung der restriktiven Maximaleinschlagsmengen bereitstellen können. Die Varianten, welche sowohl die Eingriffsdringlichkeit als auch die räumliche Nähe der Bestände berücksichtigt, weist hinsichtlich der Teilindikatoren im Vergleich zu den Varianten RN und ED mittlere Werte auf. Der Indikator zur Aggregation liegt zwischen 0,36 und 0,66 bei der Variante RN bei 0,66. Der Abstandsindikator ist für die Suche nach Beständen mit Buchenlangholz mit einem Mindestmittendurchmesser von 55cm (0,1 - 0,48) deutlich größer im Vergleich zu den Suchergebnissen nach dem schwächeren Sortiment (0,076 - 0,18).

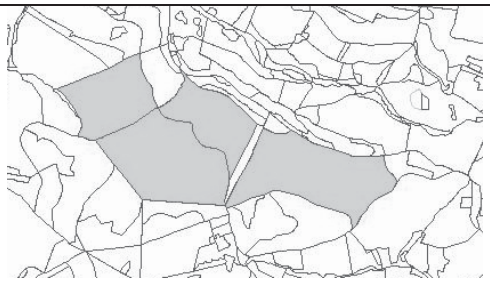
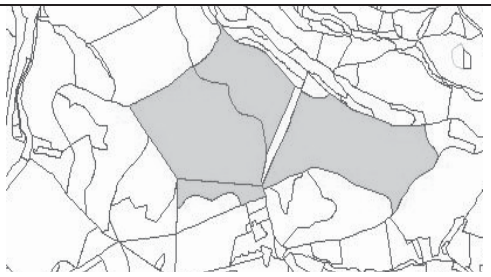
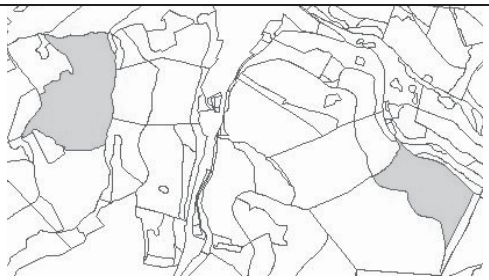
Zieldefinition	Buche, lang, MD ≥ 55, Z12, 2500 Fm	Buche, lang, MD ≥ 40, Z12, 2500 Fm
EDRN	 Aggregation: 0.36, Distanz: 0.16, Eingriffsdringlichkeit: 0.84, Rechenzeit: 514ms	 Aggregation: 0.66, Distanz: 0.13, Eingriffsdringlichkeit: 0.94, Rechenzeit: 623ms
RN	 Aggregation: 0.66, Distanz: 0.1, Eingriffsdringlichkeit: 0.9, Rechenzeit: 518ms	 Aggregation: 0.66, Distanz: 0.076, Eingriffsdringlichkeit: 0.75, Rechenzeit: 572ms
ED	 Aggregation: 0.0, Distanz: 0.48, Eingriffsdringlichkeit: 1.0, Rechenzeit: 566ms	 Aggregation: 0.0, Distanz: 0.18, Eingriffsdringlichkeit: 1.0, Rechenzeit: 833ms

Abb. 62: Ergebnismatrix für die Auswahl von Beständen, die Buchenabschnitte liefern.

Dies erklärt sich aus der Tatsache, dass mehr Bestände das schwächere Sortiment beinhalten und die absolute Menge des schwächeren Sortiments pro Bestand im Durchschnitt höher ist. Dadurch erhöht sich die Wahrscheinlichkeit, dass die ausgewählten Bestände dichter beieinander liegen. Die Eingriffsdringlichkeit bei der Variante RN ist mit 0,75 bzw. 0,9 noch vergleichsweise hoch. Das spezifizierte Sortiment wird fast ausschließlich durch zielstarke Bäume gebildet, so dass die Bestände, welche dieses Sortiment beinhalten, eine hohe Endnutzungsdringlichkeit aufweisen. Die Variante ED weist für beide Suchläufe

erwartungsgemäß die geringste Aggregation (0,0) und den höchsten Distanzindikator (0,48/0,18) auf. Die Eingriffsdringlichkeit liegt bei 1 und erreicht somit den maximal möglichen Wert. Die benötigte Rechenzeit liegt zwischen 0,8 und 0,5 Sekunden und fällt somit sehr gering aus. Die hohe Suchgeschwindigkeit wird durch die Vorauswahl potentieller Bestände ermöglicht. Dadurch ist die Suchraumgröße im Vergleich zu den beiden anderen vorgestellten Optimierungsvarianten deutlich geringer. Hinzu kommt, dass die Zielfunktion einen weniger komplexen Aufbau aufweist, da lediglich zwei Teilnutzen berücksichtigt werden, so dass auch die Zielfunktionswertberechnung schneller abläuft. Zu beachten ist, dass nicht immer zwangsläufig das gleiche Ergebnis auf eine Suchanfrage geliefert wird. Es wird in Kap. 3.7.1 gezeigt, dass PSA in 97 Prozent aller Suchläufe das globale Optimum gefunden hat. Liegt jedoch eine multikriterielle Bewertungsfunktion vor, um die aktuelle Bestandesauswahl zu beurteilen, kann es sein, dass kein eindeutiges globales Optimum existiert, da gerade bei einem Auswahlproblem durch Kompensationseffekte verschiedene Bestandeskombinationen zu gleichen Zielfunktionswerten führen können.

5.4.2.1 Eingriffsreihenfolge

Durch die Bestimmung der geringsten Distanz zwischen allen ausgewählten Beständen mittels der Kombination des Dijkstra-Algorithmus, bzw. der euklidischen Distanzberechnung und Simulated Annealing, resultiert neben der minimalen Gesamtdistanz auch die Rangfolge, in welcher die Bestände am besten abzuarbeiten sind, um einen möglichst geringen Umsetzaufwand zu erreichen. Wird nur eine geringe Anzahl von Beständen ausgewählt, ist diese Information zu vernachlässigen. Wird jedoch eine Bestandesauswahl vorgeschlagen, die viele gering aggregierte Bestände beinhaltet, kann die Information dazu beitragen Zeit und Kosten einzusparen. Dies ist meist dann der Fall, wenn die gesuchten Sortimente selten sind und nicht konzentriert in einem Bereich des Betriebs vorkommen.

Im folgenden Beispiel (Abb. 63) wurde nach 2000 Fm Kiefern-Industrieholz (oben) bzw. 4000 Fm Buchen-Industrieholz (unten) gesucht, wobei die Kiefer mit einem sehr geringen Anteil am Vorrat des Untersuchungsgebiets beteiligt ist und eine deutlich schlechtere Aggregation im Vergleich zur Buche erzielt werden kann. Die Pfeile zeigen die optimale Abarbeitungsreihenfolge an und können optional zu den ausgewählten Beständen in der Kartendarstellung angezeigt werden. Liegen Geodaten zur Erschließung vor, wird die Berechnung der Abstände zwischen zwei Beständen, die keine gemeinsame Grenze besitzen

über die Länge der kürzesten Route in dem Wegenetz zwischen den jeweiligen Beständen berechnet. Die optimale Route kann ebenfalls visualisiert werden.

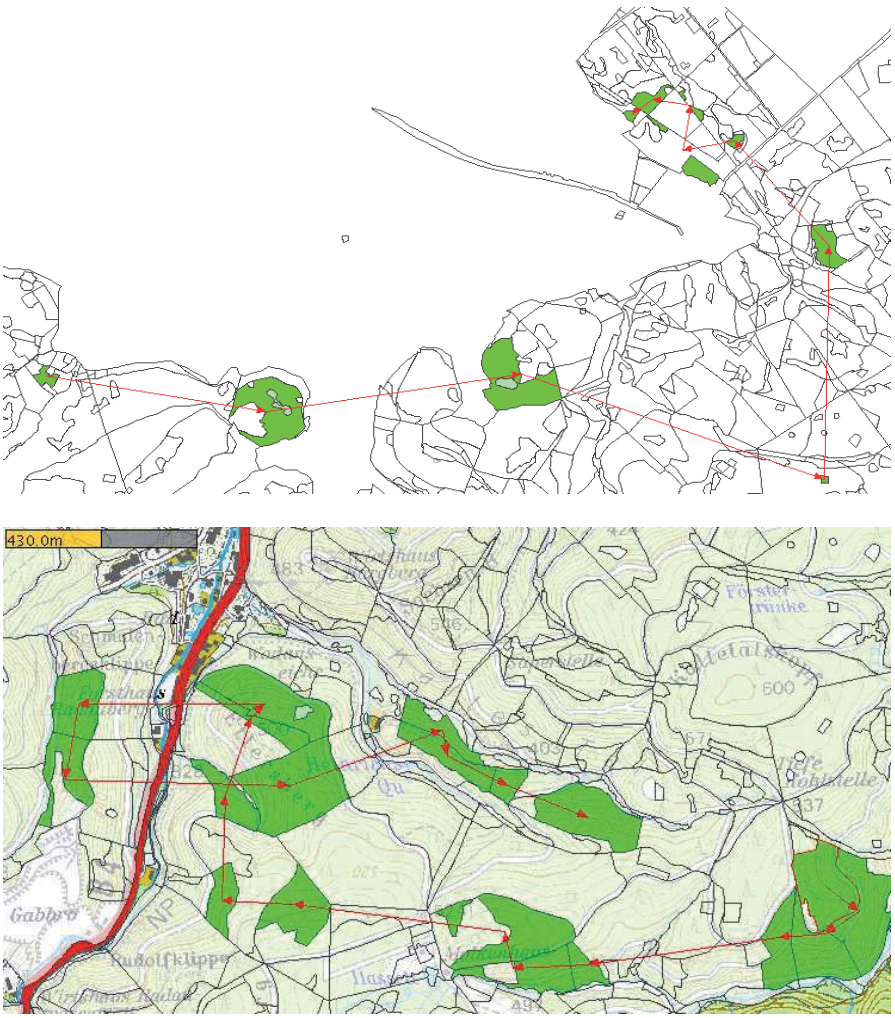


Abb. 63: Vorschlag zur Abarbeitung der ausgewählten Bestände auf Basis der euklidischen Abstände.

In Abb. 64 ist die kürzeste Route zwischen drei ausgewählten Beständen dargestellt (dicke Linie). Der Vorteil bei Verwendung der Wegstrecken liegt darin, dass in einem sehr rauen Gelände die Verhältnisse der tatsächlich zwischen zwei Beständen zurückzulegenden Strecke realistischer eingeschätzt werden kann. Bei Berücksichtigung der Wegstrecken auf Basis eines digitalen Erschließungssystems kann der Optimierungsprozess somit feiner differenzierte Lösungen ermitteln.

Abb. 64: Teilausschnitt des Wegenetzes und eine optimale Route zwischen drei ausgewählten Beständen. Liegen Daten zur Erschließung vor, wird die euklidische Distanz durch die kürzesten Routen ersetzt. Die Bestimmung erfolgt mit dem Dijkstra-Algorithmus.

5.4.3 Optimale Auswahl von Naturschutzflächen

Ziel bei der Anwendung der Komponente O*ptimalen Auswahl von Naturschutzflächen* ist es, eine bestimmte Fläche des Betriebs aus der regulären Nutzung zu nehmen. Der Hintergrund ist dabei meist naturschutzfachlich begründet. In diesem Kapitel werden die Ergebnisse der beispielhaft durchgeführten Ermittlung von mehreren Naturschutzkulissen dargestellt und diskutiert. Die ermittelte Fläche kann aus einem oder mehreren Beständen gebildet werden, welche verschiedene definierte Eigenschaften aufweisen müssen. Das Optimierungsverfahren wird eingesetzt, um in Abhängigkeit der Nutzervorgaben, den zu erwartenden wirtschaftlichen Verlust zu minimieren und/oder verschiedene naturschutzfachliche Qualitäten zu maximieren. In diesem Zusammenhang entsteht meist ein Zielkonflikt dadurch, dass die geforderten Flächeneigenschaften und die Minimierung des wirtschaftlichen Verlustes konträre Parameter darstellen.

Im Rahmen des Anwendungsbeispiels werden mehrere Kulissen unter Berücksichtigung verschiedener Restriktionen und Zieldefinitionen generiert und diskutiert. Der Anwender kann die Eigenschaften der auszuwählenden Flächen anhand der unterstützten Restriktionen und Parameter der Zielfunktion festlegen. Berücksichtigt werden vor allem solche Parameter, welche ausgewiesene Indikatoren für die Habitatqualität darstellen (vgl. Kap. 4.6.3). Die

folgenden Abbildungen (Abb. 65 bis Abb. 70 – Kulisse 1 bis 6) zeigen die definierten Restriktionen und die Ergebnisse der sechs ermittelten Kulissen sowie die Zielgrößen *Nutzenentgang* und *Totholzneubildung*. Bei der Ermittlung der Kulissen wurden alle Bestände des Untersuchungsgebiets als mögliche Lösungskandidaten zugelassen.

Min. Blockgröße: 100ha

Verlust [€ $30a^{-1}ha^{-1}$]: 10.165,00
Totholzneubildung [$m^3/ha/a$]: 1,2

Abb. 65: Kulisse 1.

Ein bereits bestehender Schutzstatus (FFH, Naturschutzgebiet, Naturreservat etc.) wurde nicht berücksichtigt. Kulisse 1 und 2 wurden bis auf die Blockgröße mit keinen weiteren Restriktionen ermittelt. Bei der Generierung aller vorgestellten Kulissen wurde die Gesamtgröße auf 400 Hektar festgelegt. Die Größe der einzelnen zusammenhängenden Blöcke muss bei Kulisse 1 mindestens 100 ha bei Kulisse 2 mindestens 400 ha betragen. Der Aggregations-Index liegt bei beiden Kulissen nahe 1. Es werden dementsprechend 3 (Kulisse 1) bzw. 1 (Kulisse 2) zusammenhängende Blöcke gebildet. Der für 30 Jahre zu erwartende Nutzenentgang liegt in beiden Fällen bei ca. 10.200 € ha^{-1}. Bei einem völligen Verzicht auf waldbauliche Eingriffe liegt die durchschnittliche jährliche Totholzneubildung bei ca. 1,2 $m^3ha^{-1}a^{-1}$ (Kulisse 1) bzw. 1,8 m^3ha^{-1} (Kulisse 2).

Bei der Generierung von Kulisse 3 wurde eine minimale Blockgröße von 50 Hektar vorgegeben. Die Zielfunktion wurde so parametrisiert, dass der zu erwartende Nutzenentgang für 30 Jahre minimiert wird. Weitere Anforderungen an die Kulisse wurden nicht spezifiziert. Es werden vier Blöcke (Hotspots) ausgewiesen. Diese führen erwartungsgemäß im Vergleich mit Kulisse 1 oder Kulisse 2 zu einem geringeren Nutzenentgang von 1.811 € $30a^{-1}ha^{-1}$. Die Kulisse weist vergleichsweise eine noch kleinere Totholzneubildung (0,26 $m^3ha^{-1}a^{-1}$) auf. Wird die Restriktion *Blockgröße* ebenfalls aus dem Zielsystem eliminiert und nur noch der Nutzenentgang minimiert, kann das Optimierungssystem in dem gegebenen Untersuchungsgebiet eine Kulisse (Kulisse 4) ermitteln, welche überhaupt keinen

Nutzenentgang verursacht. Wird keine Mindestblockgröße gefordert, kann dass System beliebige Bestände auswählen, welche nicht zwangsläufig direkt benachbart sind. Im Kontext der Minimierung des Nutzenentgangs werden durch das Optimierungssystem solche Flächen ausgewählt, die eine sehr junge oder gar keine Bestockung aufweisen.

Min. Blockgröße: 400ha

Verlust [€ $30a^{-1}ha^{-1}$]: 10.216,00
Totholzneubildung [$m^3ha^{-1}a^{-1}$]: 1,8

Abb. 66: Kulisse 2.

Min. Blockgröße: 50ha

Verlustminimierung

Verlust [€ $30a^{-1}ha^{-1}$]: 1.811,00
Totholzneubildung [$m^3ha^{-1}a^{-1}$]: 0,26

Abb. 67: Kulisse 3.

Das mittlere Bestandesalter der ausgewählten Bestände beträgt ca. 17 Jahre bei einem mittleren Derbholzvorrat von ca. 24 VFm. Dieses Ergebnis kann auch so interpretiert werden, dass die Einhaltung einer Mindestblockgröße von 50 ha innerhalb der simulierten 30 Jahre ca. 1.800 € ha^{-1} kostet. Das geringe Alter führt ebenfalls zu einer geringen Totholzneubildung innerhalb der simulierten 30 Jahre. Diese beträgt bei Kulisse 3 und 4 lediglich 0,26 bzw. 0,0 $m^3ha^{-1}a^{-1}$.

Min. Blockgröße: >0

Verlustminimierung

Verlust [€ $30a^{-1}ha^{-1}$]: 0
Totholz [$m^3ha^{-1}a^{-1}$]: 0

Abb. 68: Kulisse 4.

Wird in das Zielsystem zusätzlich zu der Verlustminimierung die Maximierung der Totholzneubildung aufgenommen, resultiert Kulisse 4 (Abb. 69). Es werden 7 Blöcke oder Hotspots ausgewiesen, die die geforderten Restriktionen erfüllen. Bei Einhaltung der geforderten Blockgröße von 50 ha liegt die Totholzneubildung bei 2,4 $m^3ha^{-1}a^{-1}$. Die Verluste belaufen sich in 30 Jahren auf ca. 6.400 € ha^{-1}. Im Vergleich zu Kulisse 3 (Blockgröße 50 ha, nur Verlustminimierung) ist der Verlust deutlich höher. Die zusätzliche Maximierung der Totholzneubildung kostet entsprechend ca. 4.600 € $ha^{-1}30a^{-1}$ (= 6.400 minus 1.800). Dafür steigt die Totholzneubildung von 0,26 auf 2,4 $m^3ha^{-1}a^{-1}$, was einer Steigerung von etwa 850 Prozent entspricht.

Min. Blockgröße: 50ha

Verlustminimierung
Maximierung Totholz

Verlust [€ $30a^{-1}ha^{-1}$]: 6.399,00
Totholz [$m^3ha^{-1}a^{-1}$]: 2,4

Abb. 69: Kulisse 5.

Bei der Berechnung von Kulisse 6 (Abb. 70) werden zwei weitere Restriktionen eingeführt. Neben der Blockgröße (mind. 50 ha) muss die ausgewählte Fläche ein Mindestbestandesalter von 100 Jahren und einen Laubholzanteil von mindestens 80 Prozent aufweisen. Das Zielsystem berücksichtigt weiterhin die Verlustminimierung und die Maximierung der

Totholzneubildung. Die ausgewählte Fläche gliedert sich in zwei Blöcke, welche im Nordosten des Untersuchungsgebiets liegen. Dies entspricht den Erwartungen, da in diesem Bereich die Baumartenzusammensetzung der Bestände laubholzdominiert ist. Der Laubholzanteil auf der ausgewählten Fläche beträgt ca. 92 Prozent und liegt somit über den geforderten 80 Prozent.

Min. Blockgröße: 50ha
min. Alter 100 Laubholzanteil 80%

Verlustminimierung
Maximierung Totholz

Verlust [€ $30a^{-1}ha^{-1}$]: 18.174,00
Totholz [$m^3ha^{-1}a^{-1}$]: 0,8

Abb. 70: Kulisse 6.

Das durchschnittliche Alter des Hauptbestands beträgt 129 Jahre und die Restriktion Mindestalter wird ebenfalls nicht verletzt. Auffallend im Vergleich mit den Kulissen 1 - 5 ist der deutlich höhere wirtschaftliche Verlust. In dreißig Jahren könnten auf den ausgewählten laubholzdominierten Beständen ein erntekostenfreier Erlös von 18.174 € ha^{-1} erzielt werden.

Das Anwendungsbeispiel zeigt, dass die Optimierungskomponente unter Einhaltung der geforderten Restriktionen solche Bestände auswählt, die das gegebene Zielsystem optimieren. Weiterhin wird deutlich, dass mit der Anzahl der Restriktionen und der Teilziele die Lösungsgüte der einzelnen Teilziele zurückgeht. Soll beispielsweise der wirtschaftliche Verlust minimiert und die Totholzmenge maximiert werden, kann der Fall auftreten, dass die Bestände, in denen der höchste wirtschaftliche Verlust bei einem Nutzenverzicht entstehen würde gleichzeitig die Bestände sind, in welchen das meiste naturschutzrelevante Totholz anfällt. Das Optimierungssystem ermittelt dann einen „Kompromiss“, der je nach Gewichtung des Zielsystems die Bestandesauswahl so trifft, dass alle Teilziele bestmöglich erreicht werden. Da die Größe der zusammenhängenden Blöcke ein wichtiges Habitatkriterium darstellt, wurde die Blockgröße als Restriktion in das System mit aufgenommen. Eine Erweiterung der Indikatoren bzw. Restriktionen würde dazu beitragen, dass die Möglichkeit, Habitate zu beschreiben, verbessert wird. Zur Zeit werden beispielsweise die Zerschneidung durch Gewässer oder Strassen, die Flächenkontinuität oder die Naturnähe noch nicht

berücksichtigt. Je nach Spezies, die von dem Nutzungsverzicht auf der ausgewählten Fläche profitieren soll, können diese oder andere Faktoren jedoch von Bedeutung sein (ROTH et al. 2006). Um die Zerschneidung zu berücksichtigen, wäre es z. B. denkbar einen Indikator zu berechnen, welcher den Widerstand beschreibt, den eine Art bei der Ausbreitung innerhalb eines Hotspots überwinden müsste und diesen in das Zielsystem mit aufzunehmen. Die Naturnähe der jeweiligen Fläche kann über den Abgleich der Anteile der vorkommenden Baumarten mit den Baumarten der PNV (potentiell natürlichen Vegetation) abgeleitet werden.

Ein Vorteil der Optimierungskomponente zur Flächenauswahl ist der flexible Aufbau. Sollen beispielsweise Hotspots für zwei verschiedene Habitattypen (mit verschiedenen Eigenschaften) ermittelt werden, ist das System in der Lage diese getrennt voneinander zu ermitteln. In diesem Fall können keine gemeinsamen Flächen der Hotspots mit verschiedenen Habitattypen entstehen. Eine weitere Möglichkeit besteht darin, Verschneidungen zuzulassen. Das Ergebnis kann in diesem Fall Flächen bzw. Teilflächen ausweisen, die sowohl den ersten als auch den zweiten Habitattyp aufweisen. Der Vorteil dabei ist, dass alle Habitattypen mit der gewünschten Fläche vertreten sind und die ausgewählte Fläche insgesamt geringer ausfällt, was wiederum die finanziellen Einbußen verringern kann.

5.4.4 Lösungsqualität

In den vorigen Kapiteln wurden Ergebnisse der konkreten Anwendung des Optimierungssystems dargestellt. Dabei lag der Schwerpunkt vor allem auf der Beschreibung und Diskussion der generierten Handlungsalternativen. In diesem Kapitel wird die Lösungsgüte (Geschwindigkeit und Qualität) der im Rahmen des Anwendungsbeispiels durchgeführten Optimumsuchen dargestellt. In Kap. 3.7.1 wurden geeignete Optimierungsverfahren anhand konstruierter Optimierungsprobleme mit bekannter optimaler Lösung verglichen und darauf aufbauend eine Entscheidung getroffen, welches oder welche Verfahren in das DSS integriert werden sollen (Kap. 4.6). Da die parallelisierte Variante des Simulated Annealing (PSA) sehr gute Lösungsqualitäten mit einer hohen Robustheit und einer praxistauglichen Lösungsgeschwindigkeit generiert, wurde dieses Verfahren bei der Implementierung der Optimierungskomponenten verwendet.

Anhand der Beurteilung der Lösungsgüte auf Basis der vorgestellten Anwendungsbeispiele soll geprüft werden, ob die entwickelten Optimierungskomponenten auch für ein reales Problem verlässliche Ergebnisse generieren. Alle verwendeten Zielssysteme transformieren die einzelnen Teilnutzen so, dass diese nur Werte zwischen 0

und 1 annehmen können. Somit kann die Lösungsqualität direkt anhand des erreichten Zielfunktionswertes, bzw. den Werten der Teilnutzen, beurteilt werden. Generiert eine Optimierungskomponente eine Lösung mit einem zugehörigen Zielfunktionswert nahe oder gleich 1, kann davon ausgegangen werden, dass eine qualitativ hochwertige Lösung nahe oder gleich dem globalen Optimum (im definierten Zielsystem) ermittelt wurde. Bei der Beurteilung des Optimierungserfolgs ist auch das Startniveau von Interesse. Die Differenz des Zielfunktionswertes der generierten optimalen Lösung zu dem Startniveau bzw. dem Zielfunktionswert der Initiallösung ist ein Indikator für die Verbesserung der Lösungsqualität durch Anwendung des Optimierungsverfahren gegenüber einer zufällig aus dem Suchraum ausgewählten Lösung.

In Tab. 20 sind die Parameter *Optimum*, *Initiallösung* und *Standardabweichung des Optimums* zur Beurteilung der Lösungsqualität dargestellt. Die Parameter *Anzahl Aufrufe der Zielfunktion*, *Zeit* und *Zeit-Anteile* geben Auskunft über den benötigten Rechenaufwand zur Lösung des Optimierungsproblems. Die Parameter sind bis auf die Standardabweichung gemittelte Werte aus den jeweils 100 durchgeführten Suchläufen. Der Verlauf des Suchpfads (einzelne Zwischenlösungen im Lösungsraum → single pass heuristic) ist teilweise zufällig, so dass Optimumsuchen mit gleicher Parametrisierung zu verschiedenen Ergebnissen führen können.

Tab. 20: Parameter zur Beurteilung der je Optimierungskomponente durchgeführten 100 Optimierungsläufe, die benötigte Rechenzeit wird einmal gesamt und einmal aufgeteilt in Rechenzeit für Simulation/Datenaufbereitung und Rechenzeit für die Optimierung angegeben.

	Holzbereitstellung	Auswahl von Nat.-Schutzflächen	Mittelfristige Bestandesbehandlung
Mittleres Optimum	0,999	0,98	0,89
Initiallösung	0.35	0.21	0.01
Mittlere Standardabweichung Optimum	0,002	0,003	0,001
Mittlere Anzahl Aufrufe der Zielfunktion (C)	1085,2	9184,3	13681,8
Mittlere Rechenzeit (Simulation + Optimumsuche) [s]	0,58 (0,18 + 0,40)	1895,4 (1,4 + 1894)	1490,1 (33,1 + 1457)
Zeit-Anteile Simulation - Optimumsuche [%]	69 - 31	99,9 - 0,1	97,8 - 2,2

Deshalb werden zur Beurteilung der Lösungsgüte die einzelnen Suchläufe mit den drei vorgestellten Komponenten (Holzbereitstellung, Auswahl von Naturschutzflächen,

mittelfristige Bestandesbehandlung) jeweils einhundert mal wiederholt. Durchschnittlich werden durch die verschiedenen Komponenten Optima zwischen 0,999 und 0,89 erreicht. Der geringste Wert (0,89) wird bei der Optimierung der mittelfristigen Behandlung erzielt. Dies ist damit zu erklären, dass bei dieser Problemstellung je nach Gewichtung und Definition der Restriktionen die höchsten Zielkonflikte der einzelnen Teilnutzen resultieren. Dennoch ist die Lösungsqualität als sehr gut einzustufen, da eine wesentliche Verbesserung des Gesamtnutzens im Vergleich zu der mittleren Initiallösung (0,011) erzielt wird. Die Initiallösungen sind bei den Auswahlproblemen (Holzbereitstellung, Auswahl von Naturschutzflächen) deutlich besser (0,35, 0,21). Dies kann damit erklärt werden, dass der Suchraum zu diesen Problemen insgesamt kleiner ist und die Anzahl der Bestände, welche die Lösung bilden, insgesamt geringer ist. Bei dem kombinatorischen Problem (mittelfristige Bestandesbehandlung) muss das Optimierungsverfahren alle Bestände des Untersuchungsgebiets berücksichtigen. Darauf beruht auch der deutlich höhere Rechenaufwand. Die Kosten- bzw. Zielfunktion wird durchschnittlich ca. 13700 mal aufgerufen. Das Suchverfahren benötigt 33 Sekunden. Bei den Auswahlproblemen beträgt die Rechenzeit für die Optimierung lediglich 1,4 bzw. 0,18 Sekunden. Die Aufbereitung benötigter Indikatoren oder die Simulation der gültigen Entwicklungspfade benötigt deutlich mehr Rechenzeit, als die eigentliche Optimierung. Der Anteil der Optimumsuche an der Gesamtrechenzeit liegt zwischen 31 Prozent (Holzbereitstellung) bis 0,1 Prozent (Auswahl von Naturschutzflächen). Die Gesamtlaufzeit ist bei der Optimierung der mittelfristigen Bestandesbehandlung mit knapp 25 Minuten am längsten. Die Optimierung der Auswahl von Beständen zur Bereitstellung definierter Zielsortimente dauert hingegen durchschnittlich lediglich ca. 0,6 Sekunden.

Zusammenfassend kann festgestellt werden, dass alle drei Optimierungskomponenten gute Lösungen generieren, welche in einer praxistauglichen Rechenzeit ermittelt werden. Die geringe Standardabweichung der erzielten Optima zeigt, dass die Lösungsgüte nur in geringen Maße variiert, so dass auch von einer hohen Konstanz der Lösungsgüte ausgegangen werden kann.

5.5 Fazit

Die Anwendungsbeispiele zeigen, dass das vorgestellte DSS mit den eingangs definierten Ebenen *Bereitstellung entscheidungsrelevanter Indikatoren*, *Szenariosimulation und -Vergleich* sowie *Ermittlung optimaler Handlungsalternativen* ein breites Spektrum an

Informationen bietet, welche für die forstliche Planung eine zusätzliche Entscheidungsgrundlage darstellen können. Die hohe Geschwindigkeit der Simulation und der Optimierung sind eine wichtige Vorraussetzung für den praktischen Einsatz der Software.

Die Berechnung von Indikatoren zu den verschiedenen Waldfunktionen auf Basis gegebener Inventurdaten ermöglichen dem Anwender eine Beurteilung der aktuellen, gesamtbetrieblichen (dem Stand der Datengrundlage entsprechenden) Situation. Die selben Parameter werden ebenfalls für simulierte waldbauliche Szenarien berechnet. So können Veränderungen zu einem Ausgangszustand und die Auswirkungen der Maßnahmen quantifiziert und beurteilt werden (PRETZSCH 2001b). Darüber hinaus können verschiedene waldbauliche Strategien miteinander verglichen werden. Dieses Variantenstudium kann dazu beitragen, eine bevorzugte Waldbaustrategie zu ermitteln. Hinsichtlich der Auswertungsmöglichkeiten sind dem Nutzer somit bezüglich der Simulation und dem Vergleich von Waldentwicklungsszenarien lediglich Grenzen durch die verwendeten Modelle gesetzt. Er kann im Rahmen der gegebenen Möglichkeiten völlig frei verschiedene Szenarien definieren und simulieren.

Darüber hinaus steht mit der Optimierungskomponente eine Anwendung zur Verfügung, welche für verschiedene waldbauliche Fragestellungen zu variablen Zieldefinitionen Lösungen generieren kann. Zum einen kann für die kurzfristige Entscheidung, welche Bestände zur Bereitstellung definierter Sortimente ausgewählt werden sollen, eine optimale Auswahl generiert werden. Zum anderen kann im Rahmen der mittelfristigen Planung eine Behandlungsmatrix generiert werden, welche eine nutzerdefinierte Zielfunktion optimiert. Weiterhin kann eine optimale Flächenauswahl zur Umwidmung in Naturschutzflächen ermittelt werden. Die Ergebnisse der Auswahl eines geeigneten Optimierungsverfahrens und des Anwendungsbeispiels zeigen, dass das System plausible und qualitativ hochwertige Lösungen generiert. Die Optimierungskomponenten generieren direkt Handlungsoptionen, ohne dass der Anwender verschiedene Alternativen in Eigenregie ermitteln und vergleichen muss. Das Optimierungssystem greift auf ein mehrkriterielles Zielsystem zurück, welches nutzerspezifisch parametrisiert wird. Somit wird das Programm einer multifunktionalen, betriebsspezifischen forstwirtschaftlichen Ausrichtung gerecht. Vor allem die optionale Einbeziehung von Raum-Zeit-Mustern ist ein wichtiger Aspekt im Kontext der Optimierung der Eingriffsplanung. Unterschiedliche Kombinationen von Bestandesentwicklungspfaden ergeben, zu bestimmten Zeitpunkten, unterschiedliche

räumliche Muster der Nutzung und Verjüngung auf Betriebsebene, welche naturschutzfachliche und ökonomische Ziele auf unterschiedliche Weise erfüllen (CHEN u. GADOW 2002, CHEN u. GADOW 2008). Das Optimierungssystem kann die Zeitpunkte der Eingriffe in einzelne Bestände so bestimmen, dass die gesamtbetriebliche Zielfunktion optimiert wird. Beispielsweise kann es günstiger sein, in einen Bestand später einzugreifen als es eine bestandesweise Optimierung der Eingriffsplanung ergeben würde, da dadurch eine bessere Blockbildung ermöglicht wird. Diese wiederum führt zu einem effizienteren Maschineneinsatz.

6 Diskussion

Wie andere Wirtschaftsbereiche unterliegt auch die Forstwirtschaft einem Strukturwandel. Immer größere Waldflächen werden von immer weniger Personal betreut (MERKER 2006). Gleichzeitig nimmt die Bedeutung des Waldes zu. So müssen nicht nur ökonomische Aspekte, sondern beispielsweise auch die Naturschutz- oder Erholungsfunktionen des Waldes bei der betrieblichen Planung mit berücksichtigt werden. Decision-Support-Systeme (DSS) können dabei helfen, die aus diesem Sachverhalt resultierenden, komplexen Planungsprobleme zu lösen.

Bisher liegen jedoch keine allgemeinen, vollständig entwickelten Systeme für die forstliche Praxis vor (TEUFEL et al. 2006). Viel versprechende Systeme, wie beispielsweise das standorts- und bestandesspezifische DSS-WuK (Decision Support System Wald und Klimawandel) befinden sich z. Z. noch in der Entwicklungs- bzw. Überarbeitungsphase (FZW 2011). Oder es werden Systeme beschrieben, welche nur für eine eingeschränkte Anzahl von Baumarten oder Bestandestypen eingesetzt werden können (z. B. SÁNCHEZ OROIS 2003, HINRICHS 2006a, DEGENHARDT 2006). Das von Duda entwickelte Softwaresystem hingegen ist mit den wichtigsten Baumarten und den meisten Bestandestypen Nordwestdeutschlands kompatibel, verfügt jedoch über keine Nutzeroberfläche (DUDA 2006), so dass die praktische Anwendung erschwert wird. Hinzukommt, dass eine Komponente zum automatisierten Erzeugen von optimalen Handlungsempfehlungen fehlt. Stang beschreibt ein System, welches die betriebsumfassende Maximierung des Ertragswerts über die Steuerung des zeitlichen Einschlagverhaltens vornimmt (STANG 2008, STANG u. KNOKE 2009). Das System wurde in MS Excel realisiert. Die Simulation waldbaulicher Szenarien und eine entsprechende Datenaufbereitung müssen in externen Programmen durchgeführt werden. Die verwendete Zielfunktion ist dabei rein auf den Ertrag ausgerichtet. Andere Waldfunktionen müssen bei

der Simulation als Restriktion berücksichtigt werden. Neben der Aufteilung des Systems auf mehrere Programme hat es den Nachteil, dass keine nutzerspezifische mehrkriterielle Zielfunktion verwendet wird. Ein vollständiges, jedoch noch nicht als Stand-Alone-Software verfügbares DSS stellt Sodke vor (SODKE et al. 2006). Das System bietet die Möglichkeit, waldbauliche Szenarien zu simulieren und die Ergebnisse in einer Datenbank abzuspeichern. Darüber hinaus können mit einer mehrkriteriellen Bewertungs- und einer Optimierungskomponente optimale Behandlungskonzepte ermittelt werden.

Sowohl Silva Support als auch das von Stang beschriebene System leiten optimale Eingriffsmethoden und Zeitpunkte unter Berücksichtigung verschiedener Restriktionen ab. Räumliche Aspekte, wie auch von Chen und Gadow beschrieben (CHEN u. GADOW 2002, CHEN u. GADOW 2008), werden nicht berücksichtigt.

Das in dieser Arbeit entwickelte System kann je nach Nutzervorgaben räumliche Aspekte bei der Optimierung mit einbeziehen. Darüber hinaus liegt hiermit ein autark lauffähiges System vor, welches alle in Kap. 2 definierten Komponenten eines DSS umfasst und keine zusätzliche Software erfordert, um benötigte Eingangsdaten oder Ergebnisse aufzubereiten und darzustellen. Vor allem die GIS-Komponente ermöglicht die direkte Kartendarstellung verschiedener Ergebnisse und Indikatoren und darüber hinaus die Aufbereitung und Berechnung raumbezogener Daten. Beispielsweise kann das System für die Sturmschadenrisikomodellierung benötigte TOPEX-Werte zu einem vorliegenden Geländemodell automatisch berechnen. Somit wird die Datenaufbereitung für den Anwender auf ein Minimum reduziert und der Einsatz weiterer Software entfällt. Das Entscheidungsunterstützungspotenzial des vorgestellten Systems beruht auf drei Ebenen: (1) Berechnung und Schätzung entscheidungsrelevanter Parameter, (2) Simulation, Auswertung und Vergleich verschiedener waldbaulicher Szenarien und (3) Ermittlung optimaler Handlungsalternativen. Der Anwender kann so entscheiden, wie hoch der Grad der Entscheidungsunterstützung durch das Systems ausfallen soll. Die Bereitstellung dieser drei Ebenen ist vor allem für die Akzeptanz der Software wichtig. Bei der Anwendung von Optimierungsverfahren (Ebene 3) ist es für den Anwender kaum möglich, den Lösungsweg nachzuvollziehen. Zwar wird eine qualitativ hochwertige Handlungsoption generiert, welche jedoch schwer zu begründen und zu kommunizieren sein kann. Deshalb ist es wichtig, dass der Anwender eigenständig Handlungsalternativen generieren und vergleichen kann (SCHMIDT et al. 2006b). Dies ist mit dem vorgestellten DSS problemlos möglich. Die in der

Literatur beschriebenen forstlichen DSS mit integrierter Optimierungskomponente sind meist auf ein spezielles Optimierungsproblem fokussiert und lassen keine weiteren Auswertungen oder frei definierte Simulationen (Pfadkombinationen) zu. Meist wird die Eignungsprüfung und Anpassung eines Optimierungsverfahrens sowie die Ergebnisaufbereitung speziell für das zugrunde liegende Optimierungsproblem vorgenommen. Das in der vorliegenden Arbeit entwickelte System wurde mit drei Optimierungskomponenten zu verschiedenen Problemstellungen ausgestattet. Aufgrund des objektorientierten Aufbaus des Optimierungssystems, der universellen Einsetzbarkeit und hohen Robustheit des entwickelten metaheuristischen Verfahrens (PSA), können weitere Optimierungskomponenten zu verschiedenen kombinatorischen oder selektiven Problemstellungen leicht nachträglich implementiert und integriert werden.

Ein weiterer Vorteil des vorgestellten Systems liegt in seiner Flexibilität hinsichtlich der benötigten Ausgangsdaten sowie der Auswertungsmöglichkeiten. Es können grundsätzliche Einzelbaumdaten (BHD, Höhe etc.) oder Bestandesdaten (dg, hg etc.) sowie Stichproben (stratifizierte Betriebsinventur) oder flächige Inventuren (Forsteinrichtung) verarbeitet werden. Über eine Plugin-Lösung können beliebig viele Einleseroutinen für verschiedene Datenstrukturen hinzugefügt werden. Die drei in Sodtke (SODTKE 2004) vorgestellten Entwürfe forstlicher DSS (KSP_DSS, DSS Silva Support, DSS LesHIS) benötigen beispielsweise zusätzliche Darstellungs- bzw. Auswertungskomponenten (z. B. ein externes GIS) und sind auf eine spezifische Struktur der benötigten Datengrundlage angewiesen. Einige der vorgestellten DSS zielen auf eine einzelbestandesweise Entscheidungsunterstützung bzw. Optimierung ab (z. B. DEGENHARDT 2006, HINRICHS 2006a, SODTKE 2004). Gerade bei der Optimierung auf betrieblicher Ebene ist es jedoch oft nötig einzelbestandesweise Optima aufzugeben, um ein gesamtbetrieblich optimales Ergebnis zu erzielen.

6.1 Parallelisierung

Durch kostengünstige Multiprozessor-Desktoprechner, Cluster-Konzepte und Technologien wie Hyperthreading oder Multicore-Prozessoren sind parallele Rechenstrukturen allgegenwärtig. Die effiziente Ausnutzung der damit verfügbaren Rechenleistung ist jedoch nur durch den Einsatz paralleler Programmiertechniken möglich (RAUBER u. RÜNGER 2007). Bei einer sequentiellen Programmstruktur eines Simulationssystems würden auf einem modernen PC mit bis zu sechs Kernen (BENZ 2010) lediglich ein Kern durch die vorgestellte

Software ausgelastet werden. Das Rechenpotenzial der übrigen Kerne bliebe ungenutzt. Die Verarbeitungsgeschwindigkeit kann dann sogar verglichen mit einem Einzelkernrechner langsamer ausfallen, da die Rechenleistung eines einzelnen Kerns meist geringer ausfällt als die eines Single-Core-Prozessors. Deshalb wurden bei der Implementierung des vorgestellten Systems verschiedene Techniken entwickelt, um parallele Hardwarestrukturen voll ausnutzen zu können und somit die rechenintensiven Programmabschnitte deutlich zu beschleunigen. Theoretisch hängt nach dem eher pessimistischen Amdahlschen Gesetz[11] (AMDAHL 1967) der SpeedUp (die Beschleunigung im Vergleich zu einem rein sequentiellen System) direkt von dem Verhältnis sequentieller und paralleler Programmanteile sowie der Anzahl zur Verfügung stehender Prozessoren ab.

Im vorgestellten System liegen zwei rechenintensive Prozesse (Simulation und Optimierung) vor. Um bei der Simulation und Optimierung schnelle Antwortzeiten zu erzielen, kommen zwei verschiedene Strategien zum Einsatz. Zum einen wird der Suchraum des Optimierungsprozesses eingegrenzt, indem das sog. Mehrpfadprinzip angewendet wird. Dieses Prinzip sieht vor, dass für einzelne Planungseinheiten (Bestände) eine endliche Anzahl an möglichen Behandlungspfaden definiert wird. Zum anderen werden so viele Rechenvorgänge wie möglich parallelisiert. Zerlegbare Probleme werden dementsprechend simultan, je nach Hardwareverfügbarkeit von mehreren Prozessorkernen oder von mehreren Prozessoren berechnet. Beispielsweise lässt sich die bestandesweise Simulation von Behandlungsvarianten gut parallelisieren. Die Gesamtmenge der zu berechnenden Bestände wird durch die Anzahl der zur Verfügung stehenden Recheneinheiten (Prozessoren/Kerne) geteilt und jede Untermenge parallel berechnet. Oder ein Masterprozess verteilt zu berechnende Bestände auf Recheneinheiten, die gerade die Simulation eines Bestandes abgeschlossen haben und entsprechend freie Rechenkapazitäten besitzen. Nach letztgenanntem Prinzip arbeitet das in dieser Arbeit entwickelte Simulationssystem. Es hat sich gezeigt, dass eine deutliche Geschwindigkeitssteigerung bis zur Aufteilung auf so viele Threads, wie Recheneinheiten zur Verfügung stehen erzielt wird. Der erreichte SpeedUp liegt bei den zu Testzwecken verwendeten Multiprozessor-Systemen zwischen 2 und 4. Die komplette Simulation (Bestände einlesen, simulieren, speichern) wird dementsprechend bis zu

[11] Amdahlsches Gesetz: SpeedUp=1/((1-P)+(P/N)), wobei N= Anzahl zur Verfügung stehender Prozessoren, P= paralleler Programmanteil, 1-P= sequentieller Programmanteil.

viermal schneller durchgeführt. Legt man das Amdahlsche Gesetz zugrunde, so läst sich ein paralleler Anteil von ca. 70 bis 85 Prozent ableiten.

Die Parallelisierung der Simulation führt jedoch dazu, dass deutlich größere Datenmengen (je nach dem Grad der Parallelisierung) pro Rechenzeit anfallen, welche in die angebundene Datenbank gespeichert werden müssen. Dies kann zu einem Engpass führen, so dass das gesamte Simulationssystem warten muss, bis wieder freie Lese/Schreib-Kapazitäten zur Verfügung stehen. Je nach Parallelisierungsgrad und Prozessorgeschwindigkeit, übersteigt die anfallende Datenrate die Schreibgeschwindigkeit der Festplatte. Um diesen „Flaschenhals“ zu vermeiden, wurde das vorgestellte System mit einer sog. Double-Buffer-Strategie ausgestattet. Dabei werden die Simulationsergebnisse jeweils in einem von zwei Buffern (Speicherbereichen) mit definierter Größe im Arbeitsspeicher abgelegt. Ein Buffer wird solange benutzt, bis dessen Kapazität erreicht ist. Ist dies der Fall, wird der zweite (freie) Buffer zum Zwischenspeichern genutzt und der Inhalt des ersten Buffers in die Datenbank geschrieben. Dies verringert die Wartezeit des Simulationssystems auf freie Schreibkapazitäten. Ein weiterer Vorteil der Buffer-Strategie ist, dass eine größere Datenmange (mehrere Bestände) mit einer Datenbankabfrage (Batch-Vorgang) übertragen wird, und somit der Kommunikationsoverhead verringert werden kann. Unabhängig von der Datenmenge müssen bei der Kommunikation zwischen Java-Programm und Datenbank zu jeder Abfrage Metadaten ausgetauscht und eine entsprechende Schnittstelle geöffnet werden. Dies kostet Rechenzeit, welche durch den Batch-Vorgang reduziert wird. Das Management der (parallel) eingehenden Anfragen der einzulesenden Bestände wird dem Datenbanksystem überlassen. Trotz der vielen Schreibvorgänge lässt sich ein hoher Parallelisierungsgrad (70 – 85 Prozent s. o.) und somit eine schnelle Gesamtabarbeitungszeit der Simulation waldbaulicher Szenarien erreichen.

6.2 Optimierung

Das entwickelte parallele Simulated Annealing (PSA), welches im Rahmen des Vergleichs verschiedener Verfahren die insgesamt besten Ergebnisse lieferte, arbeitet deutlich schneller als die sequentiell strukturierten Varianten des Simulated Annealing. In einigen vorgestellten Systemen zur Optimierung forstlicher Problemstellungen werden modifizierte metaheuristische Verfahren verwendet (z. B. modified accelerated Simulated Annealing (SEO 2005), modifizierter Tabu-Search-Algorithmus (PRETZSCH 2006)). Die Modifikationen dienen

der Anpassung an das spezielle Problem, um höhere Lösungsgeschwindigkeiten und/oder -Qualitäten zu erzielen.

Für das in der vorliegenden Arbeit beschriebene Optimierungsmodul wird eine parallel arbeitende Variante des Simulated Annealing entwickelt. Die sehr hohe Lösungsqualität - in 97 bzw. 98 Prozent der Suchläufe wird das globale Optimum gefunden - beruht hauptsächlich auf der Parallelisierungsstruktur. Diese entspricht der Type1-Parallelisierung (vgl. Kap. 3.5). Das Type2-Parallelisierungsverfahren hat den Nachteil, dass nicht nur die Suche aufgeteilt wird, sondern auch der Suchraum oder das Bewertungsproblem an sich. Daraus resultieren mehrere Teillösungen, aus denen ein sinnvolles Gesamtergebnis generiert werden muss. Nicht alle Optimierungsprobleme sind geeignet, derart zerlegt zu werden. Denkbar wäre die einfachste Variante, eine Type3-Parallelisierung durchzuführen. Diese berechnet simultan das gleiche Problem und wählt nach Beendigung aller einzelnen Optimierungsläufe das beste Ergebnis aus. Der Nachteil im Vergleich zur Type1-Parallelisierung ist jedoch, dass erst am Ende der Suche aus den Einzellösungen das beste Ergebnis ausgewählt wird und kein Geschwindigkeitsvorteil entsteht. Bei der Type1-Parallelisierung hingegen wird nach jedem inneren Schleifendurchlauf die beste Teillösung identifiziert und als neue Ausgangslösung für alle Teilsuchprozesse verwendet, so dass eine Kommunikation zwischen den einzelnen Suchprozessen schon während der Suche besteht, was die Lösungsqualität deutlich verbessern kann. Eine andere Möglichkeit besteht darin, aus den einzelnen Teillösungen, ähnlich den genetischen Verfahren, durch Rekombination eine neue Lösung zu erzeugen.

Die Parallelisierung ermöglicht eine weitere Verbesserung des PSA. PSA kann so mit zwei verschiedenen Nachbarschaftsrelationen ausgestattet werden. Die Nachbarschaftsrelation ist bei den Lokale-Suche-Verfahren mitentscheidend für den Optimierungserfolg (HEINONEN u. PUKKALA 2004, PUKKALA 2006). Durch die Verwendung von zwei verschiedenen Nachbarschaftsrelationen kann während der gesamten Suche sowohl eine hohe Diversifikation als auch eine stetige Intensivierung erzielt werden. Dadurch werden lokale Optima noch besser umgangen, der Suchraum insgesamt umfassender durchsucht (Diversifikation) und gute Zwischenlösungen durch geringfügige Änderungen auf eine mögliche Verbesserung untersucht (Intensivierung). Dass die Verwendung paralleler Strukturen die Lösungsgeschwindigkeit- und Qualität deutlich verbessern kann, zeigte schon Janaki Ram 1996 anhand des Traveling-Saleseman-Problems und des Job-Shop-Scheduling-Problems (JANAKI RAM et al. 1996).

Die Struktur des entwickelten PSA ermöglicht weiterhin neben der Verwendung von zwei verschiedenen Nachbarschaftsrelationen, die Verwendung von zwei verschiedenen Abkühlungsstrategien. Diese beeinflussen ähnlich wie die Nachbarschaftsfunktionen die Intensivierung und Diversifikation der Suche, so dass die eine positive Beeinflussung des Suchverlaufs resultiert. Dieses Phänomen wird auch von Miki bestätigt (MIKI et al. 2002). Die Autoren beschreiben das so genannte Temperature Parallel Simulated Annealing und können eine deutliche Verbesserung der Suchergebnisse gegenüber dem klassischen Simulated Annealing erzielen.

Das im vorgestellten System implementierte PSA (parallele Simulated Annealing) arbeitet mit einer Aufteilung der Suche auf zwei Threads. Eine Verbesserungsmöglichkeit dieses Verfahrens besteht darin, den Grad der Parallelisierung (Threadanzahl), ähnlich wie bei der parallelen Simulation, den gegebenen Hardwareressourcen automatisch anzupassen, so dass bei z. B. vier verfügbaren Prozessorkernen vier parallele Suchthreads gestartet werden. Um das PSA-Verfahren noch effizienter zu gestalten, könnten Modifikationen, die sich auch bei den sequentiellen Verfahren bewährt haben, eingesetzt werden. Denkbar wären z. B. zusätzliche Abbruchkriterien der inneren oder äußeren Schleife oder die Verwendung eines Gedächtnisspeichers. Bei letzterer Modifikation müsste jedoch geprüft werden, ob die Verwendung eines Speichers zum Suchverlauf nicht die Rechenzeit insgesamt negativ beeinflusst, da die gespeicherten Informationen mit den aktuellen Lösungen verglichen werden müssen. Je nach Information und Datenstruktur kann dieser Abgleich viel Rechenzeit beanspruchen.

Neben der Verbesserung des verwendeten Verfahrens bietet auch das Gesamtkonzept der Optimierungskomponente verschiedene Ausbaumöglichkeiten. Die benötigte Zeit zur Simulation eines fünf Jahre umfassenden Szenarios dauert für 2000 Bestände 100 bis 470 Sekunden. Das vorgestellte System beruht auf einer Beschränkung der möglichen Entwicklungspfade, so dass vor dem eigentlichen Suchlauf die Simulation bestandesweise für alle definierten Szenarien nur einmal durchgeführt werden muss. Die Ergebnisse werden aggregiert und im Arbeitsspeicher vorgehalten, so dass sehr schnelle Antwortzeiten der Bewertungsfunktion erzielt werden können. Dementsprechend schnell arbeitet das entwickelte System die eigentliche Optimumsuche ab. Würde die Simulation direkt in den Optimierungsprozess (die Zielfunktion) integriert werden, würde die Lösungsgeschwindigkeit wesentlich geringer ausfallen. Eine mögliche Verbesserung des Systems besteht

dementsprechend darin, die zur Simulation benötigte Rechenzeit herabzusetzen und so eine intervallweise Variation der Behandlungsvarianten für jeden einzelnen Bestand zu ermöglichen (z. B. CHEN u. GADOW 2008). D. h., die Simulation einzelner Bestände wird während des Optimierungsprozesses ein oder mehrmalig durchgeführt, um so variable Entwicklungspfade zur Laufzeit abzubilden und zu bewerten. Dies erhöht die Flexibilität des Systems, vergrößert aber auch den Suchraum. Zielgrößen (Teilkomponenten der Zielfunktion) könnten jedoch durch die Hintereinanderreihung verschiedener Varianten besser optimiert werden. Es könnte z. B. die Kombination von drei eingriffsstarken Behandlungsintervallen mit einem eingeschobenen Intervall, welches keine Nutzung vorsieht, insgesamt einen größeren Ertrag erzielen, als mit vier eingriffsstarken Intervallen, da durch eine reine Zuwachsperiode z. B. der Gewinn durch den Zuwachs der Bäume die verzinsten Erlöse übersteigt. Die Implementierung eines solchen Systems ist aber nur dann sinnvoll, wenn die Simulationsgeschwindigkeit des verwendeten Einzelbaumwuchsmodells deutlich verbessert werden könnte.

Eine weitere mögliche Verbesserung der Optimierungskomponente besteht darin, dem Benutzer nicht nur eine (die optimale) Lösung anzubieten, sondern weitere Lösungen mit einer ähnlich hohen Lösungsqualität als zusätzliche Handlungsalternativen aufzuzeigen. Die drei vorgestellten Optimierungsansätze liefern lediglich die beste Lösung. Diese ist zwar unter Berücksichtigung aller Restriktionen und der Zielfunktion eine qualitativ hochwertige Lösung, jedoch kann durch dem System unbekannte Umstände eine andere Lösung aus Sicht des Anwenders besser geeignet sein. Unter diesem Gesichtspunkt ist es sinnvoll, nicht nur eine (die beste gefundene) Lösung sondern z. B. die besten fünf gefundenen Lösungen als Ergebnis anzubieten, so dass sich dem Nutzer eine Wahlmöglichkeit zwischen mehreren Optionen eröffnet.

Da das Optimierungssystem sehr flexibel aufgebaut wurde, ist es denkbar, die implementierte Bibliothek auch in anderen Bereichen des DSS einzusetzen. So könnte beispielsweise im Rahmen einer Sortierung der einzelnen Modellbäume eine optimale Aushaltung ermittelt werden. Ein weiterer möglicher Einsatzbereich besteht in der Datenergänzung bzw. dem Aufbau der virtuellen Bestände. Liegen keine Baumkoordinaten vor oder stehen lediglich Forsteinrichtungsdaten zur Verfügung, müssen die Stammfußkoordinaten generiert werden. Die räumliche Verteilung der Bäume hat beim Einsatz eines positionsabhängigen Wuchsmodells einen großen Einfluss auf die

konkurrenzbasierten Teilmodelle (Zuwachs, Mortalität). So könnte z. B. unter der Annahme, dass die Baumkonstellation eines Bestandes einen möglichst konkurrenzarmen Zustand (durch optimale Pflege oder natürliche Auslese) aufweist, unter Zuhilfenahme der Optimierungskomponente, eine Baumverteilung generiert werden, aus der ein möglichst geringer Konkurrenzdruck für jeden Baum resultiert. Liegen Informationen zur Mischungsform mehrere Baumarten und/oder Altersklassen vor, könnten diese in Kombination mit entsprechenden Indizes zusätzlich bei der Stammfußkoordinaten-Generierung mit berücksichtigt werden. Hierzu wäre eine Zielfunktion zu implementieren, welche sowohl die Konkurrenz minimiert und den entsprechenden Mischungs-Index entweder maximiert oder ebenfalls minimiert (je nach Mischungsform: aggregiert oder durchmischt).

6.3 Nutzeroberfläche

Durch die Implementierung der parallelen Rechenstrukturen konnte die zur Szenariosimulation und zur Optimierung benötigte Rechenzeit im Vergleich zu anderen Systemen deutlich gesenkt und dadurch gleichzeitig die Anwenderfreundlichkeit gesteigert werden. Hinsichtlich der Nutzerfreundlichkeit spielt auch die Gestaltung der Nutzeroberfläche (GUI) eine wichtige Rolle. Diese muss übersichtlich gestaltet und intuitiv bedienbar sein, um eine Akzeptanz der Software seitens der Anwender zu erzielen. Die in der vorliegenden Arbeit vorgestellten Front-End-Konzepte versuchen diesem Anspruch gerecht zu werden, indem Richtlinien und Styleguides zur Oberflächengestaltung berücksichtigt werden. Die Browser-basierte Variante zeichnet sich durch den übersichtlichen und einheitlichen Aufbau der GUI aus. Diese wird als HTML-Seite auf einem Server dynamisch generiert, ahmt aber die Fenster-Metapher gängiger Betriebssysteme nach, so dass sich der Anwender intuitiv auf der Oberfläche zurechtfinden kann. Beide Konzepte informieren den Anwender ausführlich über System- oder Anwendungsfehler. Dadurch wird erreicht, dass der Nutzer möglichst selbstständig die richtige Vorgehensweise zur Aufgabenbewältigung ableiten kann. Um die Benutzerfreundlichkeit noch zu verbessern, könnte z. B. ein Praxistest durchgeführt werden und entsprechend des Feedbacks der Testuser die Oberflächengestaltung überarbeitet werden.

6.4 GIS-Komponente

Die Implementierung einer auf das vorgestellte System zugeschnitten GIS-Komponente bietet mehrere Vorteile. Die GIS-Komponente kann speziell für den serverseitigen Einsatz angepasst werden. Zur Kartendarstellung oder Aufbereitung relevanter Geoinformationen für

die Szenariosimulation und die Optimierung wird keine zusätzliche GIS-Software benötigt. So bleibt das komplett in Java implementierte System völlig betriebssystemunabhängig. Darüber hinaus resultiert hieraus ein Geschwindigkeitsgewinn, da auch alle raumbezogenen Berechnungen und die weiteren Systemkomponenten in der selben JVM laufen. Somit sind keine Schnittstellen für einen Datenaustausch und eine Datentypenumwandlung sowie ein damit verbundener erhöhter Ressourcenbedarf erforderlich. Die GIS-Komponente beinhaltet unter anderem Funktionen zur Verschneidung von Polygonen. Dies stellt einen weiteren Anknüpfungspunkt zur Ausweitung der Simulationsmöglichkeiten dar. In die Szenariosimulation könnte so die Verschneidung von vorhandenen oder geplanten Naturschutzflächen mit den momentanen Wirtschaftsflächen integriert werden. Dadurch kann z. B. ein potentieller Nutzenentgang abgeschätzt werden. Ein anderes Anwendungsfeld stellt die Holzlagerung und Abfuhrlogistik dar. In die vorhandene GIS-Komponente kann leicht eine Funktion zum Verbuchen von Holzpoltern und deren Koordinaten integriert werden. Unter Einbeziehung der vorgestellten Optimierungsverfahren kann darauf aufbauend, beispielsweise eine optimale Holzabfuhrkarte generiert werden.

Neben den technischen Vorzügen, ergibt sich für den Anwender vor allem der Vorteil, dass er kein zusätzliche GIS erwerben, erlernen und anwenden muss. Viele der vorgestellten forstlichen Entscheidungsunterstützungssysteme beruhen auf einer Kombination verschiedener Softwaremodule und Programme (z. B. DUDA 2006, SODKE et al. 2006, STANG 2008), welche Expertenwissen voraussetzen, um die gewünschten Ergebnisse zu generieren und entsprechend darzustellen. Die dem vorgestellten System beigefügte GIS-Komponente trägt wesentlich dazu bei, die Anwendung für den Praktiker zu erleichtern und die Akzeptanz der Systems zu erhöhen.

6.5 Entscheidungsrelevante Indikatoren

Um bei der Planung waldbaulicher Maßnahmen im Kontext einer mehrkriteriell ausgerichteten Forstwirtschaft alle Teilziele möglichst gut abzudecken, sollte ein entsprechendes Entscheidungsunterstützungssystem Indikatoren zu den verschiedenen Waldfunktionen bzw. Teilzielen berechnen können (ALBERT 2007, ALBERT u. HANSEN 2007, SODKE et al. 2006). Auf Grund der Verwendung eines einzelbaumorientierten Wuchsmodells können zu dem Bereich der naturalen Ausstattung sehr viele Indikatoren abgeleitet werden (Vorrat, Zuwachs, Grundfläche, Vorratsstruktur etc.). Durch die beiden eingeführten Indikatoren *Nutzungs-* und *Pflegedringlichkeit* stehen zwei geeignete Parameter zur

Verfügung, um die Dringlichkeit eines Eingriffs zu quantifizieren. Wie bei den ökonomischen Größen *Abtriebswert* und *erntekostenfreier Erlös* steht das Vergleichspotential der Indikatoren im Vordergrund. Sie dienen im Rahmen eines Variantenvergleichs oder der Optimierung in erster Linie dazu, zu beurteilen, ob eine Variante im Vergleich zu einer anderen Variante besser oder schlechter bezüglich des jeweiligen Indikators einzustufen ist. Um z. B. die absolute Aussagekraft der ökonomischen Indikatoren zu verbessern, könnten diese auf Basis einer modellhaft durchgeführten Sortierung der genutzten (Modell-) Bäume und einer hinterlegten aktuellen, sortenspezifischen Preismatrix berechnet werden. Auch der Bereich Naturschutz ist insgesamt gut abgedeckt. Auf Basis verschiedener Diversitäts-Indizes und den Ergebnissen des Totholzmodells, sowie einem integrierten Habitatbaumkonzept, kann das Teilziel Naturschutz gut abgebildet werden. Der Erholungswert von Waldgebieten kann bislang nur auf Basis indirekter Indikatoren, wie beispielsweise dem Abwechslungsreichtum der Bestandestypen oder der Altersstruktur abgeleitet werden. Das System beinhaltet jedoch keine Modelle oder Funktionen, um den Erholungswert direkt zu quantifizieren.

Vor dem Hintergrund des Klimawandels werden auch immer extremere Wetterereignisse diskutiert (Trockenheit, Stürme etc.). Solche Ereignisse stellen ein Risiko für Waldflächen dar. Je nach Bestockungsform können sich Extremereignisse unterschiedlich auswirken und entsprechenden Schaden verursachen. In dem vorgestellten System wurde das von Schmidt (SCHMIDT et al. 2010) entwickelte Modell zur Abschätzung des Risikos von Sturmschäden implementiert. Das Modell ermöglicht es, für verschiedene Bestockungen (Baumart, Höhe), Geländesituationen und Bodenverhältnisse (Vernässung) das Sturmschadensrisiko zu prognostizieren. Dieser Indikator kann bei der Planung forstlicher Maßnahmen helfen, besonders gefährdete Bestände zu identifizieren und entsprechend zu reagieren. So kann das Risiko durch Sturmschäden verringert werden. Neben Sturmschäden bestehen jedoch noch weitere Risiken für Waldflächen. Eine mögliche Verbesserung des vorgestellten Systems besteht darin, weitere Risikoindikatoren (Waldbrand, Insektenkalamitäten) zu integrieren und somit den Aspekt des Risikomanagements zu verbessern.

6.6 Modellauswahl

Einen wesentlichen Einfluss auf die Qualität der Entscheidungsunterstützung haben die verwendeten Modelle. Die Grenzen der Modelle stellen gleichzeitig die Grenzen der

Möglichkeiten bei der Entscheidungsunterstützung dar. Sie bestimmen, welche entscheidungsrelevanten Indikatoren berechnet werden können, wie hoch die Auflösung, Realitätsnähe und Verlässlichkeit der produzierten Ergebnisse ist. Zur Simulation der Bestandesentwicklung kommt in dem vorgestellten System der einzelbaumorientierte Waldwachstumssimulator BWIN bzw. TreeGrOSS zum Einsatz. Die Anwendung von Waldwachstumssimulatoren zur Durchführung von großräumigen Holzaufkommensprognosen hat in den letzten Jahren immer mehr an Bedeutung gewonnen (BÖSCH 1995, STERBA 1996, KÄNDLER et al. 2005, RÜTHER et al. 2008a, RÜTHER et al. 2008b). Sie sind mittlerweile als optimale Werkzeuge für waldbauliche Simulationen anerkannt. Zu ihren Vorteilen zählt, dass hoch aufgelöste Einzelbauminformationen genutzt werden können, ohne dass Informationen durch eine Aggregierung zu Bestandeswerten verloren gehen (PRETZSCH 2001b).

Ein gerade bei längerfristigen Simulationen wichtiger Faktor wird zurzeit vom Simulationssystem noch nicht umfassend berücksichtigt. Abiotische und biotische Störungen (Kalamitäten, Spätfröste usw.) stellen eine erhebliche Veränderung der Waldstruktur dar und können durch verschiedene Bewirtschaftungsstrategien im Vorfeld oder im Nachhinein maßgeblich beeinflusst bzw. unterschiedlich aufbereitet werden. In dem vorgestellten System ist bislang nur ein Modell zur Abschätzung des Sturmschadenrisikos integriert. Die Implementierung weiterer Störungsmodelle und Funktionen zur Bewertung der Resistenz einzelner Bestände gegenüber den Störfaktoren würde die Entscheidungsunterstützung bezüglich der Bestandesstabilität weiter verbessern.

Eine weitere Ausbaumöglichkeit des Systems besteht darin, das Wachstum der Bäume standort- und klimasensitiv abzubilden. Noch während der Erstellung der vorliegenden Arbeit wurde in das vorgestellte Softwaresystem eine Komponente integriert, welche unter Verwendung eines Wasserhaushaltsmodells und eines klimasensitiven Bonitätsmodells (ALBERT u. SCHMIDT 2010) unter Berücksichtigung verschiedener Klimaszenarien das Einzelbaumwachstum anpasst. Zunächst werden nur die Baumarten Buche und Fichte unterstützt. In naher Zukunft ist mit einer Parametrisierung des Bonitätsmodells für die übrigen Hauptbaumarten zu rechnen. Damit kann das Wachstum der Hauptbaumarten klimasensitiv modelliert und beispielsweise die Auswirkungen von verschiedenen Klimaszenarien auf die Bestandesentwicklung simuliert werden.

6.7 Praxisbezug

Das im Rahmen der vorliegenden Arbeit entwickelte Softwaresystem stellt mit den beiden implementierten Front-End-Konzepten für verschiedene Nutzergruppen ein praktikables, einfach steuerbares Entscheidungsunterstützungssystem dar. Durch die Möglichkeiten der Auswertung eines aktuellen Zustands eines Forstbetriebs, der Simulation von verschiedenen Waldentwicklungsszenarien und der Optimierung mehrerer waldbaulicher Fragestellungen, weist das System bezüglich der Einbindung in Entscheidungsprozesse eine hohe Flexibilität auf. Es liegt damit ein für den nordwestdeutschen Raum gültiges System vor, welches erstmalig alle zu einem vollständigen, forstlichen DSS gehörenden Komponenten in einem Softwaresystem vereinigt. Durch die Parallelisierungsstrategien werden die komplexen Rechenvorgänge erfolgreich beschleunigt, so dass die Wartezeiten auf die Ergebnisse der jeweiligen Anfrage (Simulation, Optimierung) auf ein praxistaugliches Maß reduziert werden konnten. Durch die strikte Trennung der implementierten Software in ein Funktionspaket und die verschiedenen Nutzeroberflächen, ist es ohne weiteres möglich, nur das Funktionspaket in bestehende Softwarekonzepte zu integrieren. Die FENA[12] hat bereits Teile des Funktionspaketes in ihr System zur Auswertung der in Hessen begonnenen Betriebsinventur auf Stichprobenbasis integriert. Das Stadtforstamt Göttingen erprobt zur Zeit den Einsatz der vorgestellten webbasierten Programmversion, welche auf einem für den Außeneineinsatz tauglichen Laptop installiert wurde. Neben der Anwendung in der Praxis ist das System auch in Lehre und Forschung einsetzbar. An der Fachhochschule Eberswalde wird im Rahmen der Lehre ein Modellbetrieb mit dem WaldPlaner analysiert und verschiedene waldbauliche Szenarien gerechnet und diskutiert (ALBERT u. HANSEN 2007). An der Nordwestdeutschen Forstlichen Versuchsanstalt wurde und wird die Software in verschiedenen Forschungsprojekten eingesetzt, um Waldentwicklungsszenarien zu rechnen und darauf aufbauende Analysen durchzuführen (RÜTHER et al. 2007, RÜTHER et al. 2008a, RÜTHER et al. 2008b, MEESENBURG et al. 2009). Aktuell steht der Einsatz der oben beschriebenen Version gekoppelt mit einem klimasensitiven Bonitätsmodell und einem Wasserhaushaltsmodell im Vordergrund (ALBERT u. SCHMIDT 2010, MEESENBURG et al. 2010, SUTMÖLLER et al. 2010).

[12] Hessen-Forst, FENA: Hessen-Forst Forsteinrichtung und Naturschutz

7 Zusammenfassung

Aufgrund der Dynamik natürlicher und anthropogen geschaffener Rahmenbedingungen sowie den steigenden Anforderungen an die Funktionen des Waldes in Deutschland, ist die Planung waldbaulicher Maßnahmen eine zunehmend komplexe Aufgabe. Hinzu kommt, dass vor dem gleichen Hintergrund klassische Planungswerkzeuge wie die Ertragstafeln an Gültigkeit verlieren (GADOW 2006). Aus diesem Informations- und Instrumentendefizit ergab sich bereits in den 1990er Jahren der Bedarf einer Neuausrichtung der klassischen Forstplanung (FRANZ 1987, GADOW 1991). Stehen zusätzliche Informationen und bessere Planungswerkzeuge zur Verfügung, können entsprechend qualitativ hochwertigere Entscheidungen getroffen werden. Der zusätzliche Informationsbedarf zeigt sich z. B. in der verstärkten Durchführung von Waldinventuren (Bundeswaldinventur, Landeswaldinventur, WZE etc.). Hinsichtlich der Planungswerkzeuge kommen immer häufiger mischbestandsfähige Waldwachstumssimulatoren zum Einsatz. Diese Simulatoren sind in der Regel als reines Funktionspaket erhältlich oder stehen als auf den Einzelbestand fokussierte Anwendungen zur Verfügung. Systeme zur Verarbeitung aller Bestände eines Forstbetriebs mit integrierter Möglichkeit zur Generierung von Handlungsalternativen sowie darauf aufbauende Verfahren zur Ableitung von Handlungsempfehlungen in Verbindung mit einer benutzerfreundlichen Oberfläche und annehmbaren Rechenzeiten befinden sich z. Z. noch in der Entwicklungsphase (TEUFEL et al. 2006). Ein Defizit verfügbarer Anwendungen besteht darin, dass nicht alle entscheidungsrelevanten Parameter berücksichtigt werden (TEUFEL et al. 2006). Folglich besteht ein Handlungsbedarf für die Entwicklung und Verbesserung entscheidungsorientierter, modellbasierter Planungswerkzeuge (SCHÖLLER u. SPORS 2001). Um eine multifunktionale Waldwirtschaft und nutzerspezifische Waldbaustrategien effektiv abzubilden und modellhaft umsetzen zu können, bedarf es eines Systems, welches aus allen Bereichen der verschiedenen Waldfunktionen (Nutzung, Schutz, Erholung) Indikatoren zum aktuellen Waldzustand aber auch für einen zukünftigen Zeitraum bestandesweise für den gesamten Betrieb aufbereiten bzw. abschätzen kann.

Die Planung forstlicher Maßnahmen unterliegt der Gliederung der Waldflächen in einzelne Planungseinheiten bzw. Bestände (Abteilung, Unterabteilung, Unterfläche) (GADOW 2003). Die Planung auf Ebene der durch unterschiedliche Standortbedingungen und durch Unterschiede in der historischen Nutzung resultierenden Einzelbestände ist mit der Planung auf Waldlandschaftsebene untrennbar verbunden. Die Bestandesentwicklung ist durch eine Abfolge forstlicher Eingriffe und deren Auswirkungen auf das Ökosystem und den

Betriebserfolg bestimmt (HINRICHS 2006b). Aus der Anzahl der einzelnen Planungseinheiten und der Vielzahl der verschiedenen Handlungsoptionen resultiert ein großer Handlungsraum. Daher ist es sinnvoll, den Handlungsraum systematisch nach optimalen Handlungsalternativen durchsuchen zu lassen.

Von verschiedenen in der Literatur vorgestellten forstlichen Entscheidungsunterstützungssystemen mit integriertem Optimierungsmodul werden diese Anforderungen schon teilweise erfüllt (ÖHMAN u. ERIKSSON 2002, HINRICHS 2006a, DEGENHARDT 2006, PUKKALA 2006, SODTKE et al. 2006, REDSVEN et al. 2007, CHEN u. GADOW 2008). Ziel der vorliegenden Arbeit ist es, ein für den nordwestdeutschen Raum gültiges, praktikables Entscheidungsunterstützungssystem (DSS) zu entwickeln, welches bei der kurz- und mittelfristigen Planung für einen Fortbetrieb und untergeordnete Einheiten unterstützend eingesetzt werden kann und verschiedene grundlegende Funktionen (GIS, Anbindung gängiger Datenbanksysteme, Simulation von Wachstum und Eingriffen, Optimierung verschiedener waldbaulicher Fragestellungen) in einem benutzerfreundlichen System vereinigt. Dem Anwender wird so eine dreistufige Informationsbasis geboten. Auf der ersten Stufe werden entscheidungsrelevante Parameter zu den verschiedenen Waldfunktionen auf Basis von Inventur- oder Einrichtungsdaten generiert (Status quo). Die zweite Stufe bietet die Möglichkeit zur Simulation waldbaulicher Szenarien auf Basis des Status quo und schafft so die Grundlage für einen Variantenvergleich. Im Rahmen der dritten Stufe generiert das System optimale Handlungsoptionen zu verschiedenen waldbaulichen Fragestellungen unter Beachtung nutzerspezifischer Zielvorstellungen. Die Steuerung des Systems erfolgt aus einer anwenderfreundlichen Nutzeroberfläche, unter welcher die Ergebnisse in Form von Tabellen, Graphiken und Karten dargestellt werden. Zur Erstellung des Systems wird vorbereitend zunächst ein Konzept entwickelt, welches sich nach dem Grundaufbau eines Entscheidungsunterstützungssystems (DSS) richtet. Ein DSS besteht in der Regel aus den drei Subsystemen *Dialogmanagement*, *Datenbankmanagement* sowie *Modell-*, *Methoden-* und *Ergebnismanagement* (SPRAGUE u. CARLSON 1982). Da im Rahmen des Variantenvergleichs und der Generierung von Handlungsoptionen die Entwicklung der einzelnen Bestände simuliert werden und optimale Handlungsalternativen ermittelt werden sollen, werden ein Wuchsmodell, ein Maßnahmenmodell, ein System zur multikriteriellen Bewertung und ein geeignetes Optimierungsverfahren benötigt. Die Modelle werden vom Modellmanagement gesteuert und die Ergebnisse durch das Ergebnismanagement aufbereitet. Die Nutzerinteraktion und die Ergebnisdarstellung werden vom Dialogmanagement übernommen.

Bei der Auswahl eines geeigneten Wuchsmodells muss beachtet werden, dass die nötige Auflösung zum Abbilden von Maßnahmen und eine Rückkopplung der Maßnahmen auf die Bestandesentwicklung möglich ist. Es liegen bereits verschiedene valide Waldwachstumsmodelle vor, wobei die Auflösung vom Biom- bis hin zum prozessorientierten Modell auf Baumkompartimentsebene reicht (PRETZSCH 2001b). Für die Verwendung in dem hier beschriebenen DSS sind statistische einzelbaumorientierte Modelle gut geeignet, da sie die geforderte Auflösung und Rückkopplungsfähigkeit aufweisen und die benötigte Rechenzeit zur Prognose der Bestandesentwicklung z. B. deutlich geringer ist als es bei Prozessmodellen mit vergleichbarer Auflösung der Fall ist. Zu den bekanntesten, europäischen einzelbaumorientierten Wuchsmodellen zählen BWinPro (bzw. TreeGrOSS) (NAGEL 2005), MOSES 3.0 (HASENAUER et al. 2006), PrognAus 2.2 (LEDERMANN 2006), SILVA (PRETZSCH et al. 2006), STAND (PUKKALA u. MIINA 2006) und SIBYLA (FABRIKA u. DURSKY 2006).

In das im Rahmen der vorliegenden Arbeit vorgestellte DSS wird die in Java programmierte Variante TreeGrOSS des Simulator BWinPro integriert. Ergänzt wird das Wuchsmodell mit dem auf der Reihung von verschiedenen Maßnahmenelementen (Pflanzung, Durchforstung, Endnutzung, Habitatbaumauswahl etc.) beruhenden Maßnahmenmodell von Duda (DUDA 2006) und einem Totholzmodell nach Meyer (MEYER 2009). Um die bestandesweise Simulation von Entwicklungspfaden zu beschleunigen, wird ein Schwerpunkt der Arbeit auf die Parallelisierung des Simulationssystems gelegt. Das entwickelte System ist in der Lage bei gegebener Hardwareausstattung (Hyper Threading, Mehrkern-CPU, Multiprozessorsystem) die Prognose der Bestandesentwicklung simultan zu berechnen und somit insgesamt wesentlich schneller durchzuführen. Dabei wird die Simulation einzelner Bestände auf mindestens zwei sog. Threads aufgeteilt. Zur Minimierung der Rechenzeit trägt auch das Datenbankmanagementsystem bei, welches erstellte Tabellen für einen schnelleren Zugriff automatisch indiziert und die Schreibvorgänge in einem Batch-Vorgang kapselt, so dass der Datenaustausch mit der angebundenen Datenbank beschleunigt wird. Insgesamt kann eine deutliche Steigerung der Simulationsgeschwindigkeit erzielt werden. Verglichen mit dem von Duda (DUDA 2006) vorgestellten sequentiellen einzelbaumorientierten Simulationssystem wird eine Steigerung der Rechengeschwindigkeit um den Faktor 100 erreicht.

Sowohl für den Ausgangszustand als auch für die simulierten Varianten werden Indikatoren berechnet, welche bei der Entscheidungsunterstützung von Interesse sind. Die

Auswahl der Indikatoren orientiert sich an der Liste der Pan-European Indicators for Sustainable Forest Management (MCPFE 2003). Durch die verwendeten Modelle stehen auf Einzelbaumebene Parameter zur Verfügung, die zu verschiedenen entscheidungsrelevanten Indikatoren aggregiert werden können (z. B. Vorrat, dynamischer Totholzvorrat, Nutzungsmassen, Habitatbaumanzahl, Abtriebswert, ekf. Holzerlös etc.). Darüber hinaus werden verschiedene Verfahren entwickelt, um Indikatoren abzuleiten, die nicht direkt aus den Modelldaten hervorgehen. Hierzu zählen die Berechnung der Betriebstypendurchmischung auf Landschaftsebene, die Pflege- und Endnutzungsdringlichkeit einzelner Bestände, ein Indikator zur Beschreibung der Aggregation mehrerer Bestände und ein Indikator zur Quantifizierung der Konstanz von Eingriffen unter Berücksichtigung mehrerer Eingriffsintervalle.

Ein weiterer Schwerpunkt der Arbeit liegt auf der Erarbeitung des Optimierungssystems. Dieses generiert unter Berücksichtigung nutzerspezifischer Zielvorstellungen Maßnahmenpläne zu den drei Problemstellungen:

- Optimale Bestandesauswahl zur Bereitstellung definierter Sortimente (z. B. nach einer konkreten Käuferanfrage)
- Optimierung der Nutzung und Pflege (für einen klassischen 10jährigen Einrichtungsturnus)
- Optimale Auswahl von Naturschutzflächen

Dabei wird auf die simulationsbasierte Optimierung zurückgegriffen. Es werden dementsprechend die verschiedenen Modelle (Wachstum, Maßnahmen, Totholz) mit einem Optimierungsverfahren gekoppelt. Hierzu können metaheuristische Verfahren eingesetzt werden, da diese nur geringe Anforderungen an die Struktur des zu lösenden Problems stellen. Sie sind geeignet für komplexe, mathematisch nicht lösbare Optimierungsprobleme mit sehr großen Lösungsräumen durch gezielte Steuerung (meta) von Heuristiken optimale Lösungen zu finden. Eine Heuristik ist eine meist auf Erfahrungswissen basierende Regel zur Verbesserung eines Initialzustandes. Metaheuristische Verfahren sind oft natürlichen Prozessen nachgeahmt. Es können Verfahren der *Lokalen Suche* und der *Rekombinatorischen Suche* unterschieden werden. Verfahren der ersten Gruppe generieren Zwischenlösungen bis hin zur Optimallösungen durch Variation einer aktuellen Lösung. Rekombinatorische Verfahren versuchen durch die Kombination mehrerer Lösungen eine neue, bessere Lösung

zu generieren. Neben diesen klar abgegrenzten Gruppen gibt es noch eine Reihe verschiedener Verfahren, die Eigenschaften beider Gruppen besitzen (Hybride). Klassische Verfahren der Lokalen Suche sind z. B. *Simulated Annealing* oder *Tabu Search*. Zur rekombinatorischen Suche gehört beispielsweise der *Genetische Algorithmus*. Bei geeigneter Parametrisierung können diese Verfahren hochwertige Lösungen generieren. Es ist jedoch nicht garantiert, dass mit einem vertretbaren Rechenaufwand das globale Optimum ermittelt wird. Um ein geeignetes Verfahren für das vorgestellte DSS zu identifizieren, werden zunächst Vertreter der klassischen Verfahren modifiziert, um deren Lösungsgeschwindigkeit- und/oder -qualität zu verbessern. Dabei liegt ein Schwerpunkt auf der Steuerung des Suchabbruchs und der Parallelisierung der Verfahren. Zum Vergleich der klassischen und modifizierten Verfahren werden zwei hypothetische Optimierungsprobleme mit bekannter optimaler Lösung konstruiert, deren Struktur den Problemen entspricht, die im Rahmen der Entscheidungsunterstützung gelöst werden sollen. Auf Basis der erreichten Lösungsqualitäten und des dazu benötigten Rechenaufwandes wird dann eine Entscheidung getroffen, welche der Verfahren in das Entscheidungsunterstützungssystem integriert werden sollen. Verglichen werden die Verfahren Simulated Annealing (SA), Tabu Search (TS), der genetische Algorithmus (GA), die modifizierten Varianten des Simulated Annealing (SAM → mit Gedächtnis; SAG → mit Verkürzung der inneren Schleife; SAMG → mit Gedächtnis und Verkürzung der inneren Schleife), paralleles Simulated Annealing (PSA), sowie die parallelisierte Variante des Genetischen Algorithmus (PGA). Der Vergleich ergibt, dass das parallelisierte Simulated Annealing für beide Probleme insgesamt am besten abschneidet. In 97 bis 98 Prozent der durchgeführten Testläufe wurde das globale Optimum gefunden. Die dabei benötigte Rechenzeit ist deutlich kürzer als die durchschnittlich benötigte Zeit aller betrachteten Verfahren. Da im Rahmen der Quantifizierung der räumlichen Nähe ausgewählter Beständen bei gegebener Datenlage (digitalisiertes Wegenetz) die kürzeste Route zwischen mehreren Beständen ermittelt wird, ist es notwendig eine weiteres Optimierungsverfahren zu verwenden. Der speziell für Routing-Probleme ausgelegt Dijkstra-Algorithmus wird modifiziert und implementiert. Im Gegensatz zu den metaheuristischen Verfahren liefert dieses Verfahren immer das globale Optimum. Durch Modifikation des Abbruchkriteriums konnte die Lösungsgeschwindigkeit fast verdoppelt werden.

Als Benutzerschnittstelle werden zwei Front-End Lösungen implementiert: eine Einzelplatzanwendung und eine Webanwendung. Um die Effizienz und die Akzeptanz der

Software zu steigern, werden Richtlinien zur Gestaltung von Benutzeroberflächen aufgezeigt und in den Designprozess mit einbezogen. Abschließend wurde ein Anwendungsbeispiel vorgestellt und die Ergebnisse hinsichtlich der Berechnung der entscheidungsrelevanten Indikatoren, der Variantensimulation und der Optimierung diskutiert. Das beispielhaft betrachtete Waldgebiet umfasst ca. 16.000 ha und beinhaltet 4001 Planungseinheiten bzw. Bestände. Daraus resultiert im Rahmen der Optimierung ein sehr großer Suchraum. Es wird gezeigt, dass die Optimierungskomponente verlässliche Ergebnisse liefert. Mit der vorgestellten Software stehen zwei Systeme zur Verfügung, die alle für ein DSS essentiellen Komponenten in einem System vereinigen. Das Datenbankmanagement, die Simulationskomponente, die benötigten GIS-Funktionen, das Wuchs- und Behandlungsmodell und die Optimierungskomponenten sind in der Programmiersprache Java implementiert und in einem Softwarepaket zusammengefasst worden. So steht dem Anwender unter einer einheitlichen Oberfläche die gesamte Funktionalität des DSS zur Verfügung, und die Einbeziehung anderer Softwarelösungen (z. B. externes GIS) entfällt.

Weitere Entwicklungspotentiale bestehen hinsichtlich der mittelfristigen Optimierung. Das vorgestellte DSS generiert für einen mittelfristigen Zeitraum für alle Planungseinheiten des jeweiligen Untersuchungsgebiets mehrere mögliche waldbauliche Szenarien. Für jeden Bestand resultieren daraus so viele verschiedene Entwicklungspfade, wie Szenarien vorgegeben wurden. Eine bestandesweise Variation der Szenarien im Simulationsintervall ist im Rahmen einer vorgezogenen Berechnung nicht praktikabel, würde aber die Flexibilität und die Lösungsqualität des Systems erhöhen. Die bestandesweise Variation müsste vom Optimierungsverfahren während des Lösungsprozesses ermittelt werden, so dass auch die Simulation der jeweiligen Entwicklungspfade dynamisch erfolgen würde. Weiterhin unterliegt das Simulationssystem den Grenzen der verwendeten Modelle. Denkbar wäre, durch Integration weiterer Modelle (z. B. Abbildung weiterer Extremereignissen, klimasensitives Baumwachstum) die Möglichkeiten des Variantenstudiums zu erweitern und die Realitätsnähe des Prognose- und Optimierungssystems zu verbessern.

Insgesamt erfüllt das System alle eingangs formulierten Anforderungen. Es liegt damit ein für den nordwestdeutschen Raum gültiges System vor, welches erstmalig alle zu einem autarken, forstlichen DSS gehörenden Komponenten in einem Softwaresystem vereinigt.

8 Verzeichnisse

8.1 Abbildungen

8.2 Tabellen

9 Referenzen

9.1 Literatur

Albert, M. (2007): Waldwachstumssimulatoren zur mehrkriteriellen Szenariobewertung: ein Ansatz für eine benutzerfreundliche und flexible Bewertung. Tagungsband der Jahrestagung der Sektion Ertragskunde im DVFFA vom 21.-23.05.2007 in Alsfeld-Eudorf, 43-56.

Albert, M. und Hansen, J. (2007): Ein Entscheidungsunterstützungssystem für die multifunktionale Forstplanung auf Landschaftsebene. Forst u. Holz, 62. Jg., 12, 14-18.

Albert, M. und Schmidt, M. (2010): Climate-sensitive modelling of site-productivity relationships for Norway spruce (Picea abies (L.) Karst.) and common beech (Fagus sylvatica L.). Forest Ecology and Management, Vol. 259, Issue 4, 739-749.

Alho, J., M., Korhonen, P., Leskinen, P. (2002): Measurement of preferences in multiple criteria evaluation. Multi-objective forest planning, Kluwer, Dordrecht, 21-36.

Alter, S.L. (1980) Decision Support Systems: Current Practice and Continuing Challenges, Reading, MA: Addison-Wesley.

Amdahl, G. (1967): Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities. In: AFIPS Conference Proceedings. 30, 483–485.

Bailey, T.C. und Gatrell, A.C. (1995): Interactive Spatial Data Analysis. Addison-Wesley Longman, Harlow, Great Britain.

Balakrishnan, P.V. und Jacob, V. S. (1996): Genetic algorithms for product design. Management Science, 42, 1105-1118.

Balas, E. und Vazacopoulos, A. (1998): Guided local search with shifting bottleneck for job shop scheduling. Management Science, 44, 262–275.

Bang-Jensen, J. und Gutin, G. (2007): Digraphs: Theory, Algorithms and Applications. Springer Berlin Heidelberg, 754 S.

Battiti, R. und Protasi, M. (1997): Reactive Search, A history-base heuristic for MAX-SAT. ACM Journal of Experimental Algorithmics 2, Article 2.

Battiti, R. und Tecchiolli, G. (1994): The reactive tabu search. ORSA Journal on Computing Vol. 6, N. 2, 126–140.

Bayrische Landesanstalt für Wald und Forstwirtschaft (LWF) (2005): Holzaufkommensprognose für Bayern. LFW Wissen 50, Freising, 73.

Beard, D. V., Smith, D. K., Denelsbeck, K. M. (1996): Quick and dirty GOMS: A case study of computed tomography interpretation. Human-Computer Interaction, 11, 157-180.

Benz, B. (2010): Kernschau. Performance und Eigenschaften aktueller Prozessoren. c't 7/2010, Heise, 136-145.

Bettinger, P. und Zhu, J. (2006). A new heuristic method for solving spatially constrained forest planning problems based on mitigation of infeasibilities radiating outward from a forced choice. Silva Fennica 40(2): 315–333.

Bertsekas, D., Tsitsiklis, J., Wu, C. (1997): Rollout algorithms for combinatorial optimization. Journal of Heuristics, 3, 245–262.

Blum, C. und Roli, A. (2003): Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Surveys 35, 268–308.

Böckmann, T. (2004): Möglichkeiten und Grenzen von Waldwachstumsmodellen aus Sicht der forstlichen Praxis. In Hanewinkel u. v. Teufel (Hrsg.) Waldwachstumsmodelle für Prognosen in der Forsteinrichtung. Freiburger Forstliche Forschung, FVA Baden-Württemberg, 50, S. 95-102

Bonczek, R. H., Holsapple, C.W. and Whinston, A.B. (1981): Foundations of Decision Support Systems, Academic Press, New York.

Bösch, B. (1995): Ein Informationssystem zur Prognose des künftigen Nutzungspotentials. Forst und Holz 50 (19).

Brennan, J. J. und Elam, J. (1986): Enhanced Capabilities for Model-Based Decision Support Systems. In: Sprague, R.H.W. und Watson, H.J. (Ed.): Decision Support Systems. Putting Theory into Practice, Prentice-Hall, London, 130-137.

Bundesministerium für Verbraucherschutz, Ernährung und Landwirtschaft (BMVEL) (Hrsg.) (2004): Die zweite Bundeswaldinventur – BWI 2. Das Wichtigste in Kürze. Bonn, 87 S.

Bundesrat (Hrsg.) (2001): Entschließung des europäischen Parlaments zu der Mitteilung der Kommission über den Stand der Wettbewerbsfähigkeit der Holz verarbeitenden Industrie und verwandter Industriezweige in der EU. Bundesdrucksache 113/01, 10 S.

Cahon, S., Melab, N., Talbi, E.-G. (2004): ParadisEO: A Framework for the Reusable Design of Parallel and Distributed Metaheuristics. Kluver, Journal of Heuristics, 10, 357–380.

Canadian Council Of Forest Ministers (1997): Criteria and Indicators of Sustainable Forest Management in Canada. Technical Report. Ottawa, 137 S.

Chakrapani, J., Skorin-Kapov, J. (1993): Massively parallel tabu search for the quadratic assignment problem. Annals of Operations Research, 41, 327-341.

Chen, B.W. (2003): Optimization in Forest Planning. Dissertation, Fakultät für Forstwissenschaften und Waldökologie, Georg-August-Universität Göttingen. Cuvillier, Göttingen. 140 S.

Chen, B.W. u. Gadow, K. v. (2002): Timber Harvest Planning with Spatial Objectives using the Method of Simulated Annealing. Forstwiss. Centralblatt 121, 25-34.

Chen, B.W. u. Gadow, K. v. (2008): Combining spatial and other objectives in forest design. Metsanduslikud Uurimused (Forestry Studies) 48, 30-40.

Christensen, M., Hahn, K., Mountford, E. P., Ódor, P., Standóvar, T., Rozenbergar, D., Diaci, J., Wijdeven, S., Meyer, P., Winter, S. & Vrska, T. (2005): Dead wood in European beech (Fagus sylvatica) forest reserves. Forest Ecology and Management 210, S.267-282.

Christensen, S. und Oppacher, F. (2001): What can we learn from No Free Lunch? In: Spector L. et. al. (Hrsg): Proceedings of the Genetic and Evolutionary Computation Conference, Morgan Kaufmann, San Francisco, 1291-1226.

Cordeau J.-F. und Laporte, G. (2002): Tabu Search Heuristics for Vehicle Routing Problem. 15, GERAD, Montreal, Canada.

Cordeau, J.-F., Laporte, G., Pasin, F. (2008): Iterated Tabu Search for the Car Sequencing Problem. European Journal of Operational Research 191, 945-956.

Cormen, T. H., Leiserson, C., Rivest, R. L., Stein, C. (2001): Introduction to Algorithms. 2. Auflage. MIT Press 1202.

Crainic, T.G. und Toulouse, M. (2003): Parallel strategies for metaheuristics. Handbook of Metaheuristics, F. Glover und G. Kochenberger, eds., Kluwer Academic Publishers.

Czeranka, M. und Ehlers, M. (1997): GIS als Instrument zur Entscheidungsunterstützung. Geo-Informations-Systeme, Vol. 10, No. 2, 9-17.

Davidon, W. C. (1959): Variable Metric Method for Minimization. Technical Rep. 5990, Argonne National Laboratory, Argonne, IL.

Davis, G.B. (Hrsg.) (1997): The Blackwell encyclopedic dictionary of management information systems. Oxford[u. a.]: Blackwell. 263 S.

Davis, T.E., Principe, J.C. (1991): A simulated annealing like convergence theory for the simple genetic algorithm. In: Belew R., Booker, L. (Hrsg.): Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann, San Fransisco, 174–182.

Degenhardt, A. (2006): Verfahren zur Ableitung optimaler Behandlungsvarianten in Kiefernreinbeständen. In: Degenhardt, A. u. Wunn, U.: Beiträge von der 18. Jahrestagung der Sektion Biometrie und Informatik des DVFFA in Trippstadt. Die Grüne Reihe, 120-128.

Dengler, A. (1992): Waldbau. Baumartenwahl, Bestandesbegründung und Bestandespflege. 6. Auflage (bearbeitet von Röhrig, E. und Bartsch, N.), Parey, Berlin (2), 314 S.

Diestel, R. (2011): Graphentheorie, 3. Springer, Berlin, 345 S.

Dijkstra, E.W. (1959): A note on two problems in connection with graphs, Numerische Mathematik, 1:269-271.

Dittmar, O., Knapp, E. u. Lembcke, G. (1986): DDR-Buchenertragstafel 1983. IFE-Berichte.

Döbbeler, H. (2004): Simulation und Bewertung von Nutzungsstrategien unter heutigen und veränderten Klimabedingungen mit dem Wuchsmodell SILVA 2.2. Dissertation an der Fakultät für Forstwissenschaften und Waldökologie der Georg-August-Universität Göttingen, 227 S.

Döbbeler, H. und Spellmann, H. (2002): Methodological Approach to Simulate and Evaluate Silvicultural Treatments under Climate Change. Forstw. Cbl. 121, Supplement 1: 52-69.

Dorigo, M. und Stützle, T. (2004): Ant Colony Optimization. MIT Press, Cambridge.

Droste, S. und Wiesmann, D. (2002): On the design of problem-specific evolutionary algorithms. In: Ashish, G., Shigeyoshi, T. (Hrsg.): Advances in Evolutionary Computing. Springer, Berlin Heidelberg, 153–173.

Duda, H. (2006): Vergleich forstlicher Managementstrategien. Dissertation Universität Göttingen, S. 180 (http://webdoc.sub.gwdg.de/diss/2006/duda/) Duin, C. und Voß, S. (1999): The pilot method: A strategy for heuristic repetition with application to the Steiner problem in graphs. Networks 34, 181–191.

Dueck, G. und Scheuer, T. (1990): Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing. Journal of Computational Physics, 90, 161–175.

ESRI (1998): ESRI Shapefile Technical Description. Environmental Systems Research Institute, Inc., White Paper No. GS-35F-5D86H, USA, 28 S.

Fabrika, M. und Dursky, J. (2006): Implementing Tree Growth Models in Slovakia. Sustainable Forest Management, Springer/Berlin Heidelberg, 315-342.

Fink, A. (2004): Supply chain coordination by means of automated negotiations. Proceedings of the 37th Hawaii International Conference on System Sciences, IEEE.

Fisher, M. L., Rinnooy Kan, A.H.G. (1988): The design, analysis and implementation of heuristics. Management Science, 34, 263–265.

Fogel, L. J., Owens, A J., Walsh, M. J. (1966): Artificial Intelligence through Simulated Evolution. Wiley, New York.

Franz, F. (1987): Zum Aufbau eines neuzeitlichen Informationssystems für die Forstwirtschaft. Forstarchiv, 58. Jg., 131-137.

Füldner, K. (1995): Strukturbeschreibung von Buchen-Edellaubholz-Mischwäldern. Dissertation, Fakultät für Forstwissenschaften und Waldökologie, Georg-August-University Göttingen. Cuvillier, Göttingen. 146 S.

Gadow, K., v. (1991): Integration von Einzel- und Gesamtnutzungsplanung in der Forsteinrichtung. AFJZ 162 (4), 72-75.

Gadow, K. v. (2003): Waldstruktur und Wachstum. Universitätsverlag Göttingen, 241 S.

Gadow, K. v. (2005): Forsteinrichtung. Analyse und Entwurf der Waldentwicklung. Universitätsverlag Göttingen, 342 S.

Gadow, K. v. (2006): Forsteinrichtung. Adaptive Steuerung und Mehrpfadprinzip. Universitätsverlag Göttingen, 163 S.

Gadow, K. v. und Pukkala, T. (2008): Designing Green Landscapes. Managing Forest Ecosystems, Vol. 15, Springer, 290 S.

Garey, M. R. und Johnson, D.S. (1979): Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, New York.

Geoffrion, A.M. (1983): Can MS/OR Evolve Fast Enough? Interfaces, 13:1, 10-25.

Gerdes, I., Klawonn, F., Kruse, R. (2004): Evolutionäre Algorithmen. Vieweg+Teubner, 252 S.

Glover, F. (1963): Parametric combinations of local job shop rules. Chapter IV, ONR Research Memorandumg No. 117, GSIA, Carnegie Mellon University, Pittsburgh.

Glover, F. (1990): Tabu search Part II. ORSA Journal on Computing. 2, 1, 4–32.

Glover, F. (1986): Future paths for integer programming and links to artificial intelligence. Comput. Oper. Res. 13, 533–549.

Goldberg, D. E. (2002): The Design of Innovation. Genetic Algorithemns and Evolutionary Computation. Kluwer, Dordrech.

Gosavi, A. (2003): Simulation-based optimization: parametric optimization techniques and reinforcement learning. Kluwer Academic Publishers, Boston, 554 S.

Goss, S., Aron, S., Deneubourg, J. L., Pasteels, J. M. (1989): Self-organised shortcuts in the argentine ant. Naturwissenschaften 76, 579–581.

Haase, K. (1999): Modellgestützte Personaleinsatzplanung im Einzelhandel. Zeitschrift für Betriebswirtschaftslehre 69, 233–244.

Hanewinkel, M. (2001): Neuausrichtung der Forsteinrichtung als strategisches Managementinstrument. AFJZ 172. Jg. 202-211.

Hansen, J. (2006): Der WaldPlaner – Ein System zur Entscheidungsunterstützung in einer nachhaltigen Forstwirtschaft. In: Degenhardt, A. u. Wunn, U.: Beiträge von der 18. Jahrestagung der Sektion Biometrie und Informatik des DVFFA in Trippstadt. Die Grüne Reihe, 112-119.

Hansen, P. und Mladenović, N. (2001): Variable neighbourhood search – Principles and applications. European Journal of Operational Research, 130, 449–467.

Hansen, W. J. (1971): User Engineering Principles for Interactive Systems. AFIPS Conference Proceedings 39, AFIPS Press, Mondale, NJ, 523-532.

Hart, P., Nilsson, N., Raphael, B. (1968): A Formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Trans. Sys. Sci. Cybern. 2, 100-107.

Hasenauer, H. (2004): ITM – Implementing Tree Growth Models as Forest Management Tools. Final Report.

Hasenauer, H. (2006): Sustainable Forest Management: Growth models for Europe. Springer, Berlin Heidelberg, 398.

Hasenauer, H., Kindermann, G., Steinmetz, P. (2006): The Tree Growth Model MOSES 3.0. Sustainable Forest Management, Springer, Berlin/Heidelberg, 64-70.

Heckel, P. (1984): The Elements of Friendly Software Design. Warner Books, New York.

Heinen, E. (1976): Grundfragen der entscheidungsorientierten Betriebswirtschaftslehre. Goldmann, München.

Heinen, E. (1991): Industriebetriebslehre als entscheidungsorientierte Unternehmensführung. In Heinen, E. (Hrsg.): Industriebetriebslehre. 9. Auflage. Gabler, Wiesbaden, 3-69.

Heinonen, T. und Pukkala, T. (2004): A comparison of one- and two-compartment neighborhoods in heuristic search with spatial forest management goals. Silva Fenn 38, 3, 319-332.

Hinrichs, L. (2006a): Untersuchungen zur Simulation von Behandlungspfaden für Buchen-Fichten-Mischbestände. Diss., Universität Göttingen. 123 S.

Hinrichs, L. (2006b): Automatisierte Generierung von Behandlungspfaden für Mischbestände. Allg. Forst- u. J.-Ztg., 177. Jg., 5, 77-85.

Holland, J.H. (1975): Adaptation in natural and artificial systems. Ann Arbor, 1, University of Michigan Press.

Holten, R. und R. Knackstedt (1997): Führungsinformationssysteme. Historische Entwicklung und Konzeption. Arbeitsberichte des Instituts für Wirtschaftsinformatik der Westfälischen Wilhelms-Universität Münster, Nr. 55, Münster.

Holthausen, N., Hanewinkel, M., Holecy, J. (2004). Risikomanagement in der Forstwirtschaft am Beispiel des Sturmrisikos. Forstarchiv 75 (4), 149-157.

Hoos, H. H. und Stuetzle, T. (2004): Stochastic Local Search – Foundations & Applications. Morgan Kaufmann, San Francisco.

Hundeshagen, J., C. (1826): Die Forstabschätzung auf neuen wissenschaftlichen Grundlagen. H. Laupp, Tübingen.

Hurme, E., Kurttila, M., Mönkkönen, M., Heinonen, T., Pukkala, T. (2007): Maintenance of flying squirrel habitat and timber harvest: a site-specific spatial model in forest planning calculations. Landscape Ecology, 22, 243–256.

Igel, C. und Toussaint, M. (2004): A No-Free-Lunch Theorem for Non-Uniform Distributions of Target Functions. JMMA, Springer Niederlande, 313-322.

James, T., Rego, C., Glover, F. (2005): Sequential and parallel path-relinking algorithms for the quadratic assignment problem. IEEE Intelligent Systems, 20(4), 58-65.

James, T., Rego, C., Glover, F. (2009): A Cooperative Parallel Tabu Search Algorithm for the Quadratic Assignment Problem. European Journal of Operational Research, 195, 3, 810-826.

Janaki Ram, D., Sreenivas, T., H., Ganapathy Subramaniam, K. (1996): Parallel Simulated Annealing Algorithms. Journal Of Parallel And Distributed Computing, 37, 207–212.

Johnson, D. und McGeoch, L. (2002): Experimental analysis of heuristics for the STSP. In: Gutin, G., Punnen, A. P. (Hrsg.): The Traveling Salesman Problem and Its Variations. Kluwer Academic, Boston.

Jungnickel, D. (1987): Graphen, Netzwerke und Algorithmen. B.I.-Wissenschaftsverlag, Mannheim, Leipzig, Wien, Zürich.

Kändler, G., Bösch, B., Schmidt, M. (2005): Wesentliche Ergebnisse der zweiten Bundeswaldinventur in Baden-Württemberg - Rückblick und Ausblick. Forst und Holz 60 (2): 45-49.

Keen, P. G. W. und Scott-Morton, M. S. (1978) Decision Support Systems: An Organizational Perspective, Reading, MA: Addison-Wesley.

Keeney, R. L. und Raiffa, H. (1976): Decisions with Multiple Objectives. Cambridge University Press, 569 S.

Keeney, R. L. und Sicherman, A. (1976): Assessing and Analyzing Preferences Concerning Multiple Objectives: An Interactive Computer Program. Behavioral Science, 21, 2, 173-182.

Klein, R., Scholl, A. (2004): Planung und Entscheidung. Vahlen, München.

Kolda, T. G.; Lewis, R. M.; Torczon, V., (2003): Optimization by direct search: New perspectives on some classical and modern methods. SIAM Rev., 45: 385-482.

Korhonen, P., J. (1986): A hierarchical interactive method for ranking alternatives with multiple qualitative criteria. European Journal of Operational Research, 24, 2, S. 265-276.

Kronauer, H. (2007): Ruhe nach dem Sturm? Schäden durch Orkan Kyrill. AFZ- Der Wald 62 (5): 250-251.

Kurttila, M., Pukkala, T., Loikkanen, J. (2002): The performance of alternative spatial objective types in forest planning calculations: a case for flying squirrel and moose. Forest Ecology and Management 166, 245-260.

Laguna, M. und Marti, R. (2003): Scatter Search: Methodology and Implementations in C. Kluwer, Boston.

Lappi, J. (1992): JLP: A linear programming package for management planning. Finnish Forest Research Institute, Research Papers 414, 134 S.

Ledermann, T. (2006): Description of PrognAus for Windows 2.2. Sustainable Forest Management Springer Berlin/Heidelberg, 71-78.

Ledermann, T. und Neumann, M. (2009): Prognose des Waldwachstums und des Nutzungspotenzials. BFW-Praxisinformation 18, 5 – 7.

Leitão, A. B., Miller, J. Ahern, J., McGarigal, K. (2006): Measuring Landscapes: A Planners Handbook. Island Press, Washington, D.C.

Lexer, M. J. (2000): Ein multi-attributives Nutzenmodell zur Unterstützung der waldbaulichen Entscheidungsfindung dargestellt am Beispiel sekundärer Fichtenwälder. Forstw. Cbl. 119,377 – 394.

Lotfi, V., Stewart, T. J., and Zionts, S. (1992): An Aspiration-Level Interactive Model for Multiple Criteria Decision Making. Computers and Operations Research, Vol. 19, No. 7, S. 671-681.

Lourenco, H. R., Martin, O., Stützle, T. (2003): Iterated Local Search. In: Glover, F., Kochenberger, G.A. (Hrsg.): Hand-book of Metaheuristics. Kluwer, Boston, 321–353.

Marti, R., Laguna, M., Glover, F. (2006): Principles of scatter search. European Journal of Operational Research, Vol 169, 2. Ausg., Elsevier, 359-372.

Mayhew, D. (1992): Principles and Guidelines in Software User Interface Design. Prentice-Hall, New Jersey.

Merker, K., (2006): Bewirtschaftung des Staatswaldes in einer globalen Wirtschaft. Forst u. Holz 61 (7): 250 - 252.

Merkle, D., Middendorf, M. (2003): An ant algorithm with global pheromone evaluation for scheduling a single machine. Applied Intelligence, 18, 105–111.

Meesenburg, H., Hentschel, S., Sutmöller, J., Jansen, M., Ahrends, B., Döring, C., Rüping, U. (2009): SILVAQUA & SILVAQUAplus, Abschlussbericht, Pilotprojket zur Umsetzung der EG-WRRL in Niedersachsen.

Meesenburg, H., Sutmöller, J., Hentschel, S. (2010): Retrospective and prospective evaluation of water budgets at Lange Bramke, Harz Moutains, Germany: effects of plant cover and climate change. In: Status and Perspectives of Hydrology in Small Basins (Proceedings of the Workshop held at Goslar-Hahnenklee, Germany, 30 March-2 April 2009), IAHS Publ. 336, 239-244.

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M. N., Teller, A.H., Teller, E. (1953): Equations of State Calculations by Fast Computing Machines. J. Chemical Physics, vol. 21 1087-1092.

Meyer, P., Wevell von Krüger, A., Steffens, R., Unkrig, W. (2006): Naturwälder in Niedersachsen - Schutz und Forschung. Band 1. Hrg.: Nordwestdeutsche Forstliche Versuchsanstalt, Göttingen u. Niedersächsische Landesforsten, Braunschweig, 339 S.

Meyer, P., Menke, N., Nagel, J., Hansen, J., Kawaletz, H., Paar, U., Evers, J. (2009): Entwicklung eines Managementmoduls für Totholz im Forstbetrieb. Abschlussbericht DBU, 106 S.

Miki, M., Hiroyasu, T., Kasai, M., Ono, K., Jitta, T. (2002), Temperature Parallel Simulated Annealing with Adaptive Neighborhood for Continuous Optimization Problem. Computational Intelligence and Applications, 149-154.

Möhring, B., Rüping, U., Leefken, G., Ziegler, M. (2006): Die Annuität – ein „missing link“ der Forstökonomie, Allgemeine Forst- und Jagdzeitung, 177/2, 21-29.

Moning, C. und Müller, J. (2009): Critical forest age thresholds for the diversity of lichens, molluscs and birds in beech (Fagus sylvatica L.) dominated forests. Ecological Indicators, 9, 5, 922-932.

Mrosek, M., Kies, U., Schulte, A. (2005): Clusterstudie Forst und Holz Deutschland 2005. Holz-Zentralblatt, 113/84, 1113.

Müller, J., Hothorn, T., Pretzsch , H. (2007): Long-term effects of logging intensity on structures, birds, saproxylic beetles and wood-inhabiting fungi in stands of European beech Fagus sylvatica L. Forest Ecology and Management 242, 297–305.

Müller-Using, B. (2005): Totholzdynamik eines Buchenbestandes im Solling, Berichte des Forschungszentrums Waldökosysteme, Reihe A, Bd. 193. Göttingen.

Nagel, J. (1999): Einsatz von Managementmodellen in der Forsteinrichtung. Arbeitskreis Forsteinrichtung Jahrestagung 7. bis 8.10.99 in Templin, S 79-81.

Nagel, J. (2005): TreeGrOSS eine Java basierte Softwarekomponente zur Waldwachstumsmodellierung für Forschung. Lehre und Praxis. Deutscher Verband Forstlicher Forschungsanstalten - Sektion Forstliche Biometrie und Informatik, 15. Tagung Freiburg 9.-10. Oktober 2003, 33-37.

Nagel, J. (2009): Waldwachstumssimulation mit dem Softwarepaket TreeGrOSS – Neuerungen, Erweiterungsmöglichkeiten und Qualitätsmanagement. In: Römisch, K., Nothdurft, A., Wunn, U. (Hrsg.): Tagungsband der gemeinsamen Jahrestagung der Sektion Forstliche Biometrie und Informatik im DVFFA (20. Tagung) und der AG Ökologie u. Umwelt in der Intern. Biometr. Gesell., 22.-24.09.2008 in Freiburg. Die Grüne Reihe, 174-178.

Nagel, J. Duda, H., Hansen, J. (2006): Forest Simulator BWINPro7. Forst u. Holz 61, 427-429.

Niedersächsische Landesforsten (1987): Anweisung zur Betriebsregelung (Forsteinrichtung) in den Niedersächsischen Landesforsten B.A.87, Niedersächsische Landesforsten 20 S.

Nohl, W. (2001): Landschaftsplanung: ästhetische und rekreative Aspekte: Konzepte, Begründungen und Verfahrensweisen auf der Ebene des Landschaftsplans. Berlin, Patzer, 248 S.

Nowicki, E. und Smutnicki, C. (1996): A fast taboo search algorithm for the job-shop problem. Management Science 42, 2, 797–813.

Oaks, S. und Wong, H. (2004): Java Threads, 3, O’Reilly Media.

Oesten, G. und Roeder, A. (2002): Management von Forstbetrieben I. Kessel: Remagen-Oberwinter., 363 S.

Öhman, K. und Eriksson, L. O., 2002: Allowing for spatial consideration in long-term forest planning by linking linear programming with simulated annealing

Osman, I. H. (1993): Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. Ann. Oper. Res. 41, 421–451.

Osman, I.H., Laporte, G. (1996): Metaheuristics: A bibliography. Ann. Oper. Res. 63, 513–623.

Pearl, J. (1984): Heuristics: Intelligent Search Strategies for Computer Problem Solving. Addison-Wesley, Reading, Massachusetts 382 S.

Picot, A. und M. Maier (1993): Information als Wettbewerbsfaktor. In: Preßmar, D. (Hrsg.): Informationsmanagement. Schriften zur Unternehmensführung, Band 49, Gabler, Wiesbaden.

Pielou, E.C. (1966). The measurement of diversity in different types of biological collections. Volume 13, 131-144.

Pretzsch, H. (1996): Strukturvielfalt als Ergebnis waldbaulichen Handelns. Allgemeine Forst- und Jagdzeitung, 167/11, 213-221.

Pretzsch, H. (1992): Konzeption und Konstruktion von Wuchsmodellen für Rein- und Mischbestände. Forstliche Forschungsberichte München (115), 332 S.

Pretzsch, H. (2001a): Waldwachstumsforschung. Ihre Bedeutung für die Nutzung und den Schutz von Waldökosystemen. Beitr. Forstwirtsch. u. Landsch. ökol. 35 (3), 138-144.

Pretzsch, H. (2001b): Modellierung des Waldwachstums. Parey/ Berlin, 341 S.

Pretzsch, H. (2002): Application and Evaluation of the Growth Simulator SILVA 2.2 for Forest Stands, Forest Estates and Large Regions. Forstw. Cbl. 121, Supplement 1, 28-51.

Pretzsch, H., Biber, P., Ïurský, J., Sodtke, R. (2006): The Individual-Tree-Based Stand Simulator SILVA. Sustainable Forest Management, Springer Berlin/Heidelberg, 78-84.

Pretzsch, H., Kahn, M., Dursky, J. (1998): Stichprobendaten für die Entwicklungsprognose und die Nutzungsplanung. Anwendungsbeispiel: Stadtwald Traunstein. Allgemeine Forstzeitschrift 53 (25). 1552–1558.

Pukkala, T. (1987): Simulation model for natural regeneration of Pinus sylvestris, Picea abies, Betula pendula and Betula pubenscens. Silva Fennica (21), S. 37-53.

Pukkala, T. (2002): Introduction to multi-objective forest planning. Multi-objective forest planning, Kluwer, Dordrecht, 1-26.

Pukkala, T. (2006): The Use of Multi-Criteria Decision Analysis and Multi-Objective Optimisation in Forest Planning. Sustainable Forest Management, Springer Berlin/Heidleberg, 263-284.

Pukkala, T. und Mina, J. (2006): A Decision Support System for the Management of Even-Aged Stands in Finland. Sustainable Forest Management, Springer Berlin/Heidleberg, 85-90.

Rauber, T. und Rünger, G. (2007) Parallele Programmierung .2., Springer, Berlin.

Redsven, V., Hirvelä, H., Härkönen, K., Salminen, O., Siitonen, M., (2007): MELA 2007 Reference Manual. The Finnish Forest Research Institute, 642 S.

Ross, P., Hart, E., Corne, D. (2003): Genetic algorithms and timetabling. Advances in Evolutionary Computing: Theory and Applications. Springer New York, 755–771.

Roth, M., Waterstraat, A., Klenke, R. (2006): Ökologische und evolutionsbiologische Wirkungen der Segmentierung in Landschaften und der Zerschneidung in Habitaten. - In: Baier, H., Erdmann, F.; Holz, R., Waterstraat, A. [Hrsg.]: Freiraum und Naturschutz - Die Wirkungen von Störungen und Zerschneidungen in der Landschaft. Springer. Berlin, Heidelberg, New York. 143-150.

Rothlauf, F. (2002): Representations for Genetic and Evolutionary Algorithms. Springer, Berlin Heidelberg.

Rowe, J. E., Vose, M. D., Wright, A. H. (2002): Group properties of crossover and mutation. Evolutionary Computation, 10, 151–184.

Roy, B. (1973): How Outranking Relation Helps Multiple Criteria Decision Making. In Cochrane, J., L. und Zeleny, M. [Hrsg.]: Multiple Criteria Decision Making. University of South Carolina Press, Columbia, SC, 179-201.

Rudolph, G. (1997): Convergence Properties of Evolutionary Algorithms. Schriftenreihe Forschungsergebnisse der Informatik, Bd. 35, Kovac, Hamburg.

Rüther, B., Hansen, J., Ludwig, A., Spellmann, H., Nagel, J., Möhring, B., Dieter, M. (2007): Clusterstudie Forst und Holz Niedersachsen. Beiträge aus der Nordwestdeutschen Forstlichen Versuchsanstalt, Band 1.

Rüther, B., Hansen, J., Ludwig, A., Spellmann, H., Nagel, J., Möhring, B., Lüpke, N. v., Schmidt-Walter, P., Dieter, M. (2008a): Clusterstudie Forst und Holz Schleswig-Holstein. Eigendruck, Göttingen, 78.

Rüther, B., Hansen, J., Spellmann, H., Nagel, J., Möhring, B., Schmidt-Walter, P., Dieter, M. (2008b): Clusterstudie Forst und Holz Sachsen-Anhalt, Nordwestdeutsche Forstliche Versuchsanstalt (Hrsg.), 60.

Ruzicka, R. (1988): SIMUL_R - eine Simulationssprache mit speziellen Befehlen zur Modelldarstellung und -analyse, Informatik Fachberichte 179, Proceedings des 5. Symposiums Simulationstechnik, Springer, Aachen.

Ruzicka, R. (1989): Environments for SIMUL_R. 3rd European Simulation Congress, Proceedings, Edinburgh.

Ruzicka, R. (1993): SIMUL_R PARALLEL - Hardware-in-the-loop Simulation mit Transputern unter Windows, Fortschritte in der Simulationstechnik. Band 6, Proceeding des 8. Symposiums Simulationstechnik, Vieweg, Berlin.

Saaty, T. L. (1980): The Analytic Hierarchy Process: Planning, Priority Setting, Re-source Allocation, New York.

Saaty, T. L. (2000): Fundamentals of Decision Making and Priority Theory with the Analytic Hierarchy Process, Vol. VI, 2. Auflage, Pittsburgh.

Sachsenforst (2008): Biotopbäume und Totholz. Markenzeichen naturnaher Wälder, Fördermittel der Richtlinie Wald und Forstwirtschaft (WuF 2007) für den Waldnaturschutz. Staatsbetrieb Sachsenforst, http://www.smul.sachsen.de/sbs/download/Biotopbaeume.pdf

Sagl, W. (1995): Bewertung in Forstbetrieben. Berlin [u. a.]: Blackwell-Wissenschaftsverlag. 396 S.

Sánchez Orois, S. (2003): Untersuchungen zur optimalen Steuerung der Waldentwicklung. Dissertation, Institut für Waldinventur und Waldwachstum, Universität Göttingen, 126 S.

Sander, J. (2000): Betriebssteuerung mit moderner Informationstechnik und Aufgaben der Forsteinrichtung in der Niedersächsischen Landesforstverwaltung. Forst und Holz 55 (5). 150–153.

Schelhaas, M. J. (2008): Impacts of natural disturbances on the development of European forest resources: application of model approaches from tree and stand levels to large-scale scenarios. Alterra Scientific contributions 23. Alterra, Wageningen, 10 - 17.

Schmidt, M. (2001): Prognosemodelle für ausgewählte Holzqualitätsmerkmale wichtiger Baumarten. Dissertation, Fakultät für Forstwissenschaften u. Waldökologie der Georg-August-Universität, Göttingen, 296 S.

Schmidt, M., Bayer, J., Kändler, G. (2006a): Sturm „Lothar“ - Ansatz einer inventurbasierten Risikoanalyse. Jahrestagung der Sektion Ertragskunde im DVFFA 2006 in Staufen, 5-20.

Schmidt, M, Böckmann, T., Nagel, J. (2006b): The Use of Tree Models for Silvicultural Decision Making. Sustainable Forest Management, Springer Berlin/Heidleberg, 237-261.

Schmidt, M. und Hansen, J. (2007): Validierung der Durchmesserzuwachsprognose des Wachstumssimulators BWINPro für Fichte und Buche für den Bereich der alten Bundesländer. Tagungsband der Jahrestagung der Sektion Ertragskunde im DVFFA vom 21.-23.05.2007 in Alsfeld-Eudorf, 164-179.

Schmidt, M., Nowack, S., Riebeling, R. (2008): Methodische Ansätze und Ergebnisse zur Quantifizierung des Rotkerns in Hessen. Ergebnisse angewandter Forschung zur Buche, Universitätsverlag Göttingen, 267-290.

Schmidt, M., Hanewinkel, M., Kändler, G., Kublin, E., Kohnle, U. (2010): An inventory-based approach for modeling single tree storm damage - experiences with the winter storm 1999 in southwestern Germany. Canadian Journal of Forest Research, NRC Research Press, Canada, Volume 40, 8, 1636-1652.

Schober, R. (1995): Ertragstafeln wichtiger Baumarten. J. D. Saarländers Verlag. Frankfurt a. M.

Schöller, W. und Spors, H.-J. (2001): Einführung von Steuerungselementen in der Landesforstverwaltung NRW. Allgemeine Forstzeitschrift 56 (14). 740–742.

Schulte, A. (Hrsg) (2003): Clusterstudie Forst und Holz NRW. Ministerium für Umwelt und Naturschutz, Landwirtschaft und Verbraucherschutz NRW.

Schultz, J. und Mertens, P. (2000): Untersuchung wissensbasierter und weiterer ausgewählter Ansätze zur Unterstützung der Produktionsfeinplanung – ein Methodenvergleich. Wirtschaftsinformatik, 42, 56–65.

Seo, J.- H. (2005): Modelling Applications for Optimizing Forest Development. Cuvillier Verlag Göttingen, 153 S.

Shannon, C. E. und Weaver, W. (1949): The Mathematical Theory of Communication. Univ. of Illinois Press, Urbana. 3-91.

Shneiderman, B. (2001): User Interface Design. Deutsche Ausgabe, mitp. 704 S.

Simpson, E. H. (1949): Measurement of diversity. Nature, 163, 688 S.

Söderberg, I. U. und Ledermann, T. (2003): Algorithms for simulating thinning and harvesting in five European individual-tree growth simulators: a review. Computers and electronics in Agriculture (39), 115-140.

Sodtke, R., Schmidt, M., Fabrika, M., Nagel, J., Dursky, J., Pretzsch, H. (2004): Anwendung und Einsatz von Einzelbaummodellen als Komponenten von entscheidungsunterstützenden Systemen für die strategische Forstbetriebsplanung. Forstarchiv 75, 51-64.

Sodtke, R. M., Utschig, H., Pretzsch, H. (2006): A Decision Support System for Multi-Criteria Forest Estate Planning, Integration a Forest Growth Simulator, Fuzzy-Interface Techniques and a Heuristic Optimization Approach. Sustainable Forest Management, Springer Berlin/Heidelberg, 211-233.

Spellmann, H. (1998): Überführung als betriebliche Aufgabe. M. Hanewinkel (Hrsg.),Überführung von Altersklassenwäldern in Dauerwälder. Freiburger Forstliche Forschung, Heft 8: 194 - 217.

Spellmann, H. (2002): Indikatoren einer nachhaltigen und multifunktionalen Forstwirtschaft. DFV Journal 1/2002. 1-3.

Speidel, G. (1972): Planung im Forstbetrieb. Parey, Hamburg.

Sprague, R. H., Jr. und Carlson, E. D. (1982): Building Effective Decision Support Systems, Englewood Cliffs, NJ: Prentice Hall.

Stang, S. (2008): Optimierung der Forstbetriebsplanung zur Bewertung von Nutzungseinschränkungen. Allg. Forst Z. Waldwirtsch. Umweltvorsorge, 17, 905–907.

Stang, S. und Knoke T. (2009): Optimierung der Hiebsatzplanung zur Quantifizierung von finanziellen Ertragseinbußen durch den Klimawandel am Beispiel des Forstbetriebes der Stadt Zittau. Waldökologie, Landschaftsforschung und Naturschutz, 8, 89-94.

Sterba, H. 1996: Holzaufkommen in Österreich. Die Sägeindustrie Österreichs (CD und Begleittext), Wirtschaftskammer Österreich, Wien.

Sterba, H., Vospernik, S., Söderbergh, I., Ledermann, T. (2006): Harvesting Rules and Modules for Predicting Commercial Timber Assortments. Sustainable Forest Management, Springer Berlin/Heidelberg, 111-123.

Stöcker, M. (2008): GIS – mehr als nur ein Werkzeug zur Erstellung forstlicher Kartenwerke. AFZ, 6, 282 – 283.

Sutmöller, J., Hentschel, S., Hansen, J., Meesenburg, H. (2010): Coupled forest growth-hydrology modeling as an instrument for the assessment of effects of forest management on hydrology in forest catchments. ADGEO, Copernicus Publications.

Taillard, E. D. (1991): Robust Taboo Search for the Quadratic Assignment Problem. Parallel. Computing. 17, 443–455

Taillard, E. D., Gambardella, L.-M., Gendreau, M., Potvin, J.-Y. (2001): Adaptive memory programming: A unified view of metaheuristics. European Journal of Operational Research, 135, 1–16.

Teufel, K., v., Hein, S., Kotar, M., Preuhsler, E. P., Puumalainen, J., Weinfurter, P. (2006): End User Needs and Requirements. Sustainable Forest Management, Springer Berlin/Heidelberg, 19-38.

Tremer, N. (2008): Untersuchung zur Verjüngung von Waldbeständen in Nordwestdeutschland. Cuvillier Verlag Göttingen, 149 S.

Ullenboom, C. (2007): Java ist auch eine Insel. 7., aktualisierte Auflage Galileo Computing, 1492 S.,ISBN 978-3-8362-1146-8

Van Laarhoven, P. J. M. und Aarts, E. H. L. (1987): Simulated Annealing: Theory and Applications. Reidel, Dordrecht.

Vose, M. (1999): The Simple Genetic Algorithm: Foundations and Theory. Complex Adaptive Systems. MIT Press.

Voss, S. und Gutenschwager, K. (2001): Informationsmanagement. Springer.

Walrath, K., Campione, M., Huml A. (2004): The JFC Swing Tutorial. 2. Auflage. Addison-Wesley Professional.

Wanka, R. (2006): Approximationsalgorithmen Eine Einführung. 1. Auflage, Vieweg u. Teubner, 206 S.

Weber, W. (1995): Unternehmensstrategie privater Forstbetriebe. AFZ 4/1995, 177 – 180.

Wegener, I. (2003): Towards a theory of randomized search heuristics. In: Rovan, B. und Vojtáš, P. (Hrsg.): Mathematical Foundations of Computer Science, Lecture Notes in Computer Science 2747, Springer, Berlin Heidelberg, 125–141.

Weicker, K. (2007): Evolutionäre Algorithmen. 2. Auflage, Vieweg u. Teubner, 313 S.

Weise, W. (1880): Ertragstafel für die Kiefer. Springer, Berlin, 156 S.

Wiedemann, E. (1949): Ertragstafeln der wichtigsten Holzarten. Schaper Verlag, Hannover.

Wittmann, W. (1959): Unternehmung und unvollkommene Information: Unternehmerische Voraussicht – Ungewissheit und Planung. Westdeutscher Verlag, Köln.

Wollborn, P. und Böckmann, T. (1998): Ein praktikables Modell zur Strukturierung des Vorrates aus Ertragstafelschätzungen. Forst und Holz, 53, 18, 547-540.

Wolpert, D. H. und Macready, W.G. (1997): No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1, 67–82.

Wooldridge, M.J. (2002): Introduction to Multi-Agent Systems. Wiley, Chichester.

9.2 Web-Links

Forschungszentrum Waldökosysteme (FZW), Georg-August-Universität Göttingen (2011): http://www.dss-wuk.de/impressum, 15.12.2011.

GeoTools (2009): http://geotools.codehaus.org/, 15.05.2009.

ILOG (2009): http://www.ilog.com/, 15.05.2009.

Metaheuristics Network (2009): http://www.metaheuristics.net/, 18.05.2009.

MCPFE (2003): Improved Pan-European Indicators for Sustainable Forest Management, MCPEF Liaison Unit Vienna, http://www.mcpfe.org, 18.05.2009.

SAS (2008): http://www.sas.com/technologies/analytics/optimization/or/index.html, 15.05.2009.

Sun (2009a): http://java.sun.com/j2se/1.4.2/docs/guide/jdbc/getstart/GettingStartedTOC.fm.html, 15.05.2009.

Sun (2009b): http://www.sun.com/software/products/appsrvr/index.jsp, 15.05.2009.

The Apache Software Foundation (2009): http://tomcat.apache.org/index.html, 15.05.2009.