Technische Hochschule Brandenburg
University of Applied Sciences

Fachbereich Informatik und Medien

MASTERARBEIT

Entwicklung einer prototypischen Web-Applikation zur optimierten Menüplanung unter Verwendung von terminologischem Wissen

Vorgelegt von: Franziska Krebs

am: 07.03.2017

zum

Erlangen des akademischen Grades

MASTER OF SCIENCE

(M.Sc.)

Erstbetreuer: Dipl.-Inform. Ingo Boersch

Zweitbetreuer: Prof. Dr. rer. nat. Rolf Socher

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit zum Thema

Entwicklung einer prototypischen Web-Applikation zur optimierten Menüplanung unter Verwendung von terminologischem Wissen

vollkommen selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe. Die Arbeit wurde in dieser oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Brandenburg/Havel, den 07.03.2017

Unterschrift

Danksagung

Zuvorderst danke ich Ingo Boersch, der mich bereits in zahlreichen Projektarbeiten betreute und dabei stets Raum und Zeit für Diskussionen fand. Während des gesamten Master-Studiums verhalf er mir mehrmals zur gedanklichen Ordnung und war dabei eine nie versiegende Quelle an Ermutigung, Bekräftigung und Expertise.
Prof. Socher danke ich für seine prägnante Art, die mir wohl ewig ein Vorbild sein wird.
Ein besonderer Dank gilt Dr. Andreas Billig, der mich seit dem Beginn meiner Tätigkeit am Fraunhofer FOKUS unnachgiebig die wissenschaftliche Denkweise lehrte. Wie kein anderer vermittelte er mir die Notwendigkeit, komplexe Sachverhalte stets auf das Wesentliche zu reduzieren. Die Möglichkeit, mich in dieser Disziplin zu trainieren, bekam ich während mehrerer Hochschulprojekte, welche geschickt in die industrieorientierten Arbeiten am Institut eingebettet wurden. Auch dieser glückliche Umstand ist ihm zu verdanken.
Zuletzt - jedoch keinesfalls im Geringsten - danke ich meinem Lebensgefährten Tobias Franz. Mit niemandem hätte ich die vergangenen sechs Jahre lieber am Schreibtisch verbracht als mit ihm. Sein Rückhalt und seine Unterstützung während des gesamten Studiums waren von unschätzbarem Wert.

Zusammenfassung

Die Gestaltung eines Menüplans ist mit Schwierigkeiten verbunden, wenn die Bedürfnisse und Geschmäcker mehrerer Personen berücksichtigt werden müssen. Im Rahmen dieser Arbeit wurde eine Web-Applikation entwickelt, welche dieses Zuordnungsproblem durch Umsetzung eines evolutionären Algorithmus und unter Verwendung einer Wissensbasis automatisch löst. Die Menüplanung berücksichtigt Ernährungsweisen, Allergien, Intoleranzen, Zubereitungsdauern sowie individuelle tages- und mahlzeitenbezogene Nährstoffgrenzen. Es werden drei zu optimierende Zielfunktionen definiert, welche die Menüpläne hinsichtlich ihrer Vielfalt und ihrer Beachtung personenspezifischer Lebensmittelvorlieben und -abneigungen bewerten. Terminologisch strukturiertes Wissen hilft dabei, unzulässige Belegungen auf Mahlzeiten-Ebene zu vermeiden. Durch die Framework-gestützte Implementierung des Constrained Nondominated Sorting Genetic Algorithm II werden mehrere Kompromiss-Lösungen gefunden. Anhand eines Beispiels wurde gezeigt, dass auch für komplexe Planstrukturen mit vielen Restriktionen zulässige Lösungen hervorgebracht und fortlaufend verbessert werden können. Da die nachempfundene Evolution unzulässige Lösungen zügig verdrängt, nimmt die Diversität der Population ab.

Abstract

Designing a menu plan can be a challenging task if demands and tastes of multiple persons need to be considered. This thesis demonstrates the development of a web application that automatically solves this assignment problem by applying evolutionary computation and utilizing a knowledge base. The planning process considers different diets, allergies, intolerances, preparation times as well as individual day- and meal-related nutritional limits. A menu plan is evaluated by three objective functions that express a plan's variety and the extent to which a plan respects personal preferences and aversions for food. Terminological structured knowledge facilitates in avoiding infeasible assigments at the meal level. By implementing the Constrained Nondominated Sorting Genetic Algorithm II, multiple tradeoff-solutions are found. With an example it was shown that feasible solutions can be found and progressively improved for complex plan structures with several constraints. Because the imitated evolution quickly eliminates infeasible solutions, the population's diversity decreases.

Inhaltsverzeichnis

1 Einleitung

Die richtige Ernährungsweise ist eine wesentliche Voraussetzung für den Erhalt und für die Förderung körperlicher, geistiger und sozialer Fitness. Die Frage, welche Ernährung die richtige ist, kann nicht pauschal beantwortet werden, sondern muss stets unter Berücksichtigung individueller Umstände gestellt werden. In der Forschung wird dieses Problem in den Ernährungswissenschaften, in der Diätik und in der Ökotrophologie wissenschaftlich untersucht. Die in diesen Disziplinen gewonnenen Erkenntnisse haben Einfluss auf Prozesse in der Ernährungs- und Versorgungswirtschaft, auf die Entwicklung von Lebensmitteln, Geräten und Herstellungsverfahren und schlussendlich auf den alltäglichen Umgang mit dem Thema Ernährung.
Wie ein im Jahr 2016 veröffentlichtes Ergebnis einer Befragung[1] zeigte, ist der überwiegende Teil der Grundgesamtheit interessiert an den Themen Kochen und Kochrezepte. Dabei stieg die Anzahl der Personen, welche sich häufiger über diese Themen im Internet informieren, in den letzten zwei Jahren von 13 Millionen auf circa 17 Millionen.
In vielen Fällen erscheint die individuelle Gestaltung eines Ernährungsplans aufgrund spezieller Anforderungen erschwert, sodass es einer genauen Auseinandersetzung mit Gerichten, Lebensmitteln, Nährstoffen und weiteren Faktoren bedarf. Der Zusammenfluss mehrerer Informationsquellen ist erstrebenswert, wird aber häufig aufgrund heterogener Datenstrukturen nicht erreicht.

1.1 Problembeschreibung

Die Zusammenstellung geeigneter Speisen in Form eines Menüplans, ob im Alltag oder in speziellen Umgebungen wie beispielsweise in Krankenhäusern oder Rehabilitationseinrichtungen, ist keine simple Aufgabe. Neben der Einhaltung allgemein bekannter Empfehlungen (z.B. die 10 Regeln der Deutschen Gesellschaft für Ernährung e.V. [Deu]) sind auch Nahrungsmittelallergien, -intoleranzen, Ernährungsweisen (z.B. vegetarisch, vegan) sowie vorübergehende diätische Einschränkungen wie Vollkost, Schonkost oder Reduktionskost zu berücksichtigen.

[1] Eine grafische Visualisierung der Befragungsergebnisse befindet sich im Anhang A.1.

Die Planung wird zusätzlich durch organisatorische Faktoren wie beispielsweise Zubereitungszeiten oder eine schwankende Anzahl zu versorgender Personen erschwert.
Um die Akzeptanz eines Menüplans zu steigern und somit die tatsächliche Einhaltung zu forcieren, sollten individuelle Speisepräferenzen (z.B. auf der Basis von Lebensmitteln oder Geschmacksrichtungen) ein weiteres unterscheidendes Merkmal sein.
Die Erstellung eines Menüplans kann grundsätzlich als Suchproblem und genauer als Zuordnungsproblem mit den zuvor beschriebenen Nebenbedingungen sowie zu optimierenden Parametern verstanden werden. Das folgende Beispiel verdeutlicht dies:
Für eine Woche mit sieben Tagen sollen für jeden Tag drei Mahlzeiten geplant werden. Die Menüplanung berücksichtigt die Ansprüche und Geschmäcker von vier Personen. Person A hält eine strenge Diät und konsumiert täglich maximal 1800 Kalorien. Person B leidet unter einer Laktoseintoleranz und isst vegetarisch. Die Personen C und D ernähren sich vollwertig und bevorzugen Fisch und Kartoffelprodukte, während letztere von Person A nicht gerne gegessen werden. Person B nimmt an den ersten fünf Tagen der Woche nicht an der ersten Mahlzeit teil. Für alle weiteren Mahlzeiten bevorzugt diese Person Nudelgerichte sowie jegliches Gemüse - mit Ausnahme von Spargel und Rosenkohl. Aus einer endlichen Menge von Gerichten sollen nun 21 Gerichte so eingeplant werden, dass die jeweiligen Einschränkungen der Personen erfüllt werden. Ein Plan, welcher Gerichte mit möglichst vielen der präferierten Zutaten und möglichst wenigen der nicht präferierten Zutaten umfasst, ist dabei besser zu bewerten als ein Plan, welcher diese Faktoren nicht berücksichtigt.
Die Komplexität dieser Planung hängt zum einen von der Granularität der beschriebenen Kriterien und zum anderen von der Anzahl der Lösungskandidaten ab. Ein Lösungskandidat entspricht einer Belegung des Menüplans. Werden wenig einschränkende Restriktionen definiert und ist die Menge der möglichen einzelnen Komponenten eines Plans gering, kann eine Planung manuell vorgenommen werden. Ist der Suchraum hingegen sehr groß und sind die Restriktionen nicht trivial, ist das Finden einer Lösung mit einem hohen zeitlichen Aufwand verbunden.
Die Planung fußt weiterhin auf einer sinnvoll aufgebauten Wissensbasis, welche Zusammenhänge zwischen Gerichten, Lebensmitteln und ernährungsrelevanten Einschränkungen beschreibt. Es wird davon ausgegangen, dass keine Datenbasis verfügbar ist, welche das gesamte für die Planung benötigte Wissen in geeigneter Weise kapselt. Die Schwierigkeiten liegen daher bei der Beschaffung und Vernetzung mehrerer Datenquellen.

1.2 Eingrenzung und Zielstellung

Ausgangspunkt dieser Arbeit ist die zweiteilige Forschungsfrage, (I) wie das Menüplanungsproblem mit Hilfe von Optimierungsverfahren gelöst und (II) wie das in Form von Terminologien strukturierte Domänenwissen dafür genutzt werden kann.
Zur Beantwortung von (I) wird das Problem analysiert und konzeptionell unter dem Gesichtspunkt der Optimierung betrachtet. Es werden verschiedene Lösungsverfahren vorgestellt, von welchen eines für die Umsetzung ausgewählt wird. Das benötigte Wissen zu Gerichten, Lebensmitteln und gesundheitlichen Einschränkungen stammt aus unterschiedlichen Quellen und wird im Zuge der Arbeit sinnvoll miteinander vernetzt. Dieses terminologische Wissen wird durch einen Dienst verwaltet, welcher auf semantischen Technologien aufgebaut ist. Auf eine detaillierte Vorstellung dieses Dienstes wird im Rahmen dieser Arbeit verzichtet.
Das Ziel dieser Arbeit ist die Bereitstellung einer Software-Komponente, welche die automatische Zusammenstellung eines Menüplans vornimmt und dabei das terminologisch strukturierte Wissen nutzt. Dafür wird die Form einer Web-Applikation gewählt, welche neben der Planung die Erfassung von Nutzerprofilen und planungsspezifischen Daten ermöglicht.
Die Implementierung einer assistierenden Komponente, welche dem Nutzer die Bearbeitung eines generierten Menüplans erlaubt, wird als optionale Teilaufgabe festgelegt. Weiterhin wird das Erfüllen nicht-funktionaler Anforderungen (z.B. die Oberflächengestaltung betreffend) als marginal eingeordnet. Eine Ausnahme davon stellt das Laufzeitverhalten dar. Diesbezüglich wird gefordert, dass eine Abfrage eines Menüplans jederzeit möglich ist.
Die Leistung der umgesetzten Menüplanung ist in geeigneter Weise zu bewerten.

1.3 Aufbau der Arbeit

Der nachfolgende Abschnitt präsentiert und kategorisiert die aus der Problembeschreibung abgeleiteten Anforderungen an die Menüplanung.
Kapitel 2 gibt die für die Einordnung und Bearbeitung relevanten theoretischen Inhalte wieder und schildert Lösungsansätze. Das darauffolgende Kapitel widmet sich der Konzeption der optimierten Menüplanung. Nach einer Vorstellung ähnlicher Arbeiten wird das Menüplanungsproblem formal beschrieben und unter dem Gesichtspunkt der Optimierung klassifiziert. Daran schließen sich die Präsentation des Lösungsansatzes und die Beschreibung des Aufbaus der Wissensbasis an. Am Ende des Kapitels werden die Anforderungen hinsichtlich ihrer Umsetzbarkeit beurteilt.

Die Implementierung der Web-Applikation wird in Kapitel 4 erläutert. Die Umsetzung des konzipierten Lösungsansatzes mithilfe eines Java-Frameworks wird geschildert. Weitere behandelte Schwerpunkte sind die aufgebaute Software-Architektur und die Evaluation der Optimierung.
Am Ende der Arbeit werden die Ergebnisse zusammengefasst und kritisch diskutiert. Die Arbeit schließt mit einem Ausblick auf weitere vorstellbare Tätigkeiten.

1.4 Anforderungen an die Menüplanung

Im Folgenden werden Kriterien vorgestellt, welche die Menüplanung beeinflussen. Ob und in welchem Maße diese tatsächlich berücksichtigt werden können, ist abhängig von der Qualität der Datenbasis. Diese wird in Abschnitt 3.4 näher erläutert, sodass zunächst für jedes Kriterium die Anforderungen an die Datenbasis formuliert und fortlaufend beginnend mit A1 nummeriert werden. Ist bekannt, welche Daten für die Menüplanung herangezogen werden können, werden die umsetzbaren Kriterien formalisiert und konkrete Wertebereiche festgelegt.
Grundsätzlich wird gefordert, dass Rezepte durch Zutaten und Mengenangaben beschrieben werden (**A1**). Davon ausgehend könnte die Menüplanung die folgenden Restriktionen berücksichtigen:

1. **Schwankende Anzahl zu verpflegender Personen**
   Ein Wochenplan basiert häufig nicht auf einer gleichbleibenden Anzahl zu verpflegender Personen. Diese Schwankungen können sich tageweise oder auf einzelne Mahlzeiten auswirken. Um diese Restriktion zu berücksichtigen, muss für ein Rezept die Anzahl der Portionen festgelegt werden. (**A2**) Unabhängig davon gilt, dass alle an einer Mahlzeit teilnehmenden Personen dasselbe Gericht konsumieren.
2. **Variable Anzahl der Mahlzeiten**
   Die Menüplanung bietet dem Nutzer die Möglichkeit, die Anzahl der täglich zu planenden Mahlzeiten zu ändern und bestimmte Mahlzeiten von der Planung auszuschließen. Dies ist beispielsweise erforderlich, wenn auswärts gegessen wird. (**A3**)
3. **Ernährungsweisen**
   Folgt ein Nutzer einer bestimmten Ernährungsweise (z.B. ernährt dieser sich vegetarisch oder vegan), so sollte die Menüplanung diesen Umstand berücksichtigen. Dies setzt voraus, dass für ein Rezept eine Ernährungsweise festgelegt wird oder sich diese aus der Zutatenliste ableiten lässt. (**A4**)

4. **Lebensmittelallergien und -unverträglichkeiten**

   Leidet ein Nutzer bzw. eine Person, welche in die Menüplanung eingeschlossen ist, unter einer für die Ernährung relevanten Allergie oder Intoleranz, kann dies in Form von Ausschlussregelungen auf Rezept- und Lebensmittelebene formuliert werden. Dies setzt zum einen voraus, dass sich die zu berücksichtigenden Einschränkungen zuverlässig durch Regeln beschreiben lassen (**A5**) und zum anderen muss ein eindeutiges Lebensmittelverzeichnis als Basis für die Festlegung eben jener Regeln vorhanden sein (**A6**).

5. **Aufbrauchen vorrätiger oder gekaufter Lebensmittel**

   Um die Menge der Nahrungsmittelabfälle am Ende eines Menüplans zu verringern, sollen noch vorrätige oder für eine eingeplante Mahlzeit gekaufte Lebensmittel nach Möglichkeit vollständig aufgebraucht werden. Hierfür müssen herkömmliche Verpackungsvolumen bekannt sein (**A7**). Darüber hinaus muss unterschieden werden zwischen Lebensmitteln, welche aufgrund einer geringen Haltbarkeit zügig aufgebraucht werden müssen und Lebensmitteln, welche zur Lagerung geeignet und damit von diesem Kriterium ausgeschlossen sind (**A8**).

6. **Zubereitungsform**

   Soll die Menüplanung die Art der Zubereitung (z.B. Kochen, Backen, Frittieren) berücksichtigen, muss diese für ein Rezept angegeben oder aus den Zubereitungsanweisungen abgeleitet werden (**A9**).

7. **Zubereitungszeit**

   Soll die Menüplanung die Dauer der Zubereitung berücksichtigen, muss diese für ein Rezept angegeben werden (**A10**).

8. **Kostengrenzen**

   Eine Preisobergrenze für eine Mahlzeit kann nur dann bei der Planung berücksichtigt werden, wenn ein Preis für ein Rezept angegeben oder aus den Zutaten abgeleitet werden kann (**A11**). Alternativ ist eine Einteilung in günstig, mittelpreisig und teuer vorstellbar.

9. **Nährstoffunter und -obergrenzen**

   Soll die Menüplanung Nährstoffvorgaben (bspw. in Form von Mindest- und Maximalmengen) berücksichtigen, so müssen die entsprechenden Nährwerte für ein Rezept angegeben sein oder aus den Zutaten abgeleitet werden (**A12**).

10. **Vielfalt**

    Um bspw. sich wiederholende Gerichte oder mehrere Gerichte mit gleichen Zutaten zu

vermeiden, muss ein Maß zur Angabe der Vielfalt eines Menüplans gefunden werden. (**A13**).

11. **Bevorzugte und nicht bevorzugte Lebensmittel oder -gruppen**
    Soll die Planung bevorzugte und nicht bevorzugte Lebensmittel oder -gruppen berücksichtigen, müssen diese eindeutig mit den Zutaten eines Rezeptes in Verbindung gebracht werden (**A14**).

12. **Ernährungs- und Diättrends sowie weitere gesundheitlich bedingte Diäten**
    Wenn bei der Planung Ernährungs- und Diättrends (zum Beispiel Low Carb, Regionale Kost) sowie weitere aufgrund eines bestimmten Gesundheitszustands erforderliche Diäten (zum Beispiel natriumarme oder proteinreiche Diäten) berücksichtigt werden sollen, müssen Rezepte diesen eindeutig zugeordnet werden bzw. eindeutig durch diese ausgeschlossen werden (**A15**). Ein Ernährungstrend ist möglicherweise auch durch eine Menge der vorherig genannten Restriktionen beschreibbar.

Diese Kriterien können für eine Mahlzeit, für mehrere Mahlzeiten (an einem Tag oder über mehrere Tage verteilt), tageweise oder für eine ganze Wochenplanung gelten. Auf eine Kategorisierung der Gerichte in Frühstücke, Mittagessen, Abendbrote, Zwischenmahlzeiten o.ä. wird verzichtet.
Nicht alle der beschriebenen Kriterien müssen in einem validen Menüplan erfüllt werden. Vielmehr können die Kriterien hinsichtlich ihrer Auswirkung auf einen Menüplan in **harte, weiche** und **Optimierungskriterien** unterteilt werden. Tabelle 1.1 zeigt diese Einordnung.[2]

Tabelle 1.1: Unterteilung der Anforderungen in harte, weiche und zu optimierende Kriterien

Harte Kriterien	Weiche Kriterien	Optimierungskriterien
Variable Anzahl der Mahlzeiten (A3) Ernährungsweisen (A4) Lebensmittelallergien und -intoleranzen (A5) Nährstoffunter- und Obergrenzen (A12) Ernährungs- oder Diättrends (A15) Kosten (A11)	Zubereitungsform (A9) Zubereitungszeit (A10)	Aufbrauchen vorrätiger oder gekaufter Lebensmittel (A7, A8) Vielfalt (A13) Bevorzugte und nicht bevorzugte Lebensmittel oder -gruppen (A14)

[2] Die Anforderungen A1, A2 und A6 beziehen sich ausschließlich auf die Datenhaltung und sind nicht Teil des Optimierungsmodells. Daher erfolgt für diese auch keine Einordnung in Tabelle 1.1.

Das Verletzten eines harten Kriteriums führt dazu, dass der entsprechende Menüplan keine gültige Lösung darstellt. Wird ein **weiches Kriterium** verletzt, wird der Plan als Lösung akzeptiert. Allerdings ist dieser Plan schlechter zu bewerten als ein Plan, welcher die weichen Kriterien erfüllt. **Optimierungskriterien** beziehen sich auf Werte, welche entweder minimiert oder maximiert werden sollen. Zu diesen zählen die Kriterien A7, A8, A13 und A14. Ein optimaler Menüplan ist eine Zusammenstellung von Gerichten, für welche gilt, dass

- die Menge der nicht verarbeiteten Lebensmittel (A7, A8) minimiert wird,
- die Vielfalt maximiert wird (A13),
- die Menge der von allen Nutzern präferierten Lebensmittel (A14) maximiert wird und
- die Menge der von allen Nutzern nicht bevorzugten Lebensmitteln (A14) minimiert wird .

2 Theoretische Grundlagen der Optimierung

Der Bereich der Optimierung beschäftigt sich mit dem Finden einer optimalen Lösung für ein gegebenes Problem. Optimal bedeutet dabei entweder, dass der Wert einer Lösung maximiert oder minimiert werden soll. Der zu optimierende Wert einer Lösung ergibt sich aus der *Zielfunktion*, welche auch als Kosten- (bei einem Minimierungsproblem) oder Gewinnfunktion (bei einem Maximierungsproblem) bezeichnet wird.
Ein sehr einfaches Beispiel eines Optimierungsproblems ist das Auffinden von Extremwerten einer stetig differenzierbaren Funktion f. Dabei wird zunächst die erste Ableitung der Funktion f gleich 0 gesetzt. Die Lösungen dieser Gleichung erfüllen die notwendige Bedingung, da die Tangenten in Höchst- und Tiefpunkten parallel zur x-Achse verlaufen und deren Anstieg damit 0 sein muss (für eine Lösung x_0 muss gelten, dass $f'(x_0) = 0$). Jede Lösung wird dann zur Überprüfung der hinreichenden Bedingung in die zweite Ableitung eingesetzt. Gilt $f''(x_0) > 0$, so hat f an dieser Stelle ein relatives Minimum. Ein relatives Maximum wird nachgewiesen, wenn $f''(x_0) < 0$ gilt.
Nicht immer wird die optimale Lösung für ein Problem gesucht. Interessante Fragestellungen zielen oft auf das Finden einer Entscheidung oder eines Zahlenwertes ab. Diese unterschiedlichen Varianten werden im Folgenden anhand zweier theoretisch bedeutsamer Probleme erläutert.

Rucksackproblem (engl. *Knapsack Problem*)

Beim Rucksackproblem soll ein Rucksack mit Gegenständen aus einer Menge M bepackt werden. M umfasst n Gegenstände, wobei jeder Gegenstand ein Gewicht $g_i, i = 1, ..., n$ und einen Nutzwert $w_j, j = 1, ..., n$ hat. Diese sind durch einen Gewichtsvektor $g \in \mathbb{R}^n$ und einen Nutzwertvektor $w \in \mathbb{R}^n$ gegeben. Beim Packen des Rucksacks darf ein festgelegtes Maximalgewicht $G \in \mathbb{R}$ nicht überschritten werden. Gleichzeitig soll der Nutzwert der ausgewählten Gegenstände möglichst groß sein. Gesucht ist daher ein Vektor $x \in \{0, 1\}^n$, der für jeden Gegenstand angibt, ob er eingepackt wird (1) oder nicht (0). Eine Bepackung ist erlaubt, wenn $x \cdot g \leq G$ gilt, wobei der Operator $\cdot$ das Skalarprodukt bezeichnet. Der Wert der Bepackung ergibt sich aus $x \cdot w$.

Soll für das vorliegende Problem die **Entscheidungsvariante** gelöst werden, wird die Frage beantwortet, ob es zu einem gegebenen Nutzwert W eine erlaubte Bepackung mit mindestens diesem Nutzwert gibt. Die Antwort auf ein solches Problem ist entweder ja (es existiert eine erlaubte Bepackung) oder nein (es existiert keine erlaubte Bepackung). Bei der **Zahlvariante** wird nach dem größtmöglichen Nutzwert W_{max} gesucht. Ziel der **Optimierung** ist das Finden der optimalen Bepackung x, für welche gilt, dass $x \cdot w = W_{max}$. [Soc08, S.170]

Traveling Salesman Problem (TSP)

Ein Handlungsreisender plant eine Rundreise durch insgesamt n Städte. Die Reise soll am Ausgangspunkt enden und jede Stadt soll genau einmal besucht werden. Dabei ist zu berücksichtigen, dass die Städte unterschiedlich weit voneinander entfernt sind und sich variable Längen für die Gesamtstrecke ergeben. Abbildung 2.1 verdeutlicht dies anhand zweier unterschiedlich langer Rundreisen. Die Länge der Strecke stellt nur eine Zielfunktion

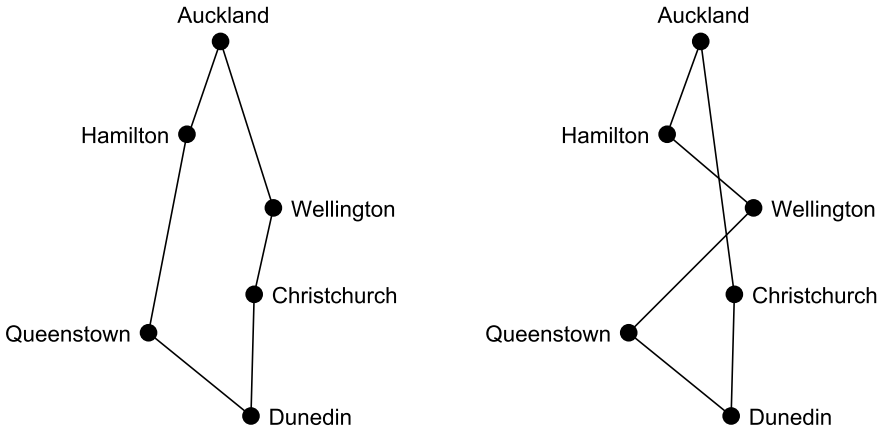


Abbildung 2.1: Zwei mögliche Rundreisen

dar. Grundsätzlich kommen auch andere Kriterien (z.B. Zeit, Fahrtkosten) in Frage.
Bei der Entscheidungsvariante des TSP ist ein Kostenwert C gegeben und es soll berechnet werden, ob es eine Rundreise gibt, deren Kosten C nicht übersteigen. Die Zahlvariante fragt nach den Kosten C_{min} der besten Reise und die Optimierung zielt auf die Berechnung eben jener Reise ab.
Viele Optimierungsprobleme gehören zur Klasse der nichtdeterministisch polynomiellen Probleme (*NP*, näher vorgestellt in Abschnitt 2.1). Für diese Probleme werden in der Praxis häufig *Approximationen*, welche einer optimalen Lösung sehr nahe kommen, akzeptiert. Dies kommt einem Herabsetzen von Qualitätsmaßstäben zugunsten einer zeitlich effizienteren

Berechnung gleich.

Die Untersuchung solcher Optimierungsprobleme gehört zu den Hauptaufgaben des *Operations Research (OR)*. Dieses Gebiet beschäftigt sich mit *„der Analyse von praxisnahen, komplexen Problemstellungen im Rahmen eines Planungsprozesses zum Zweck der Vorbereitung von möglichst optimalen Entscheidungen durch die Anwendung mathematischer Methoden.“* Diese aus [DD05] entnommene Definition betont die Bedeutung der Mathematik als tragende Wissenschaft. Sie ist nicht nur Ursprung relevanter Algorithmen, sondern wird innerhalb des OR-gestützten Planungsprozesses zur formalen Repräsentation eines realen Entscheidungs-, Optimierungs- oder Simulationsmodells genutzt. Dieser Planungsprozess gliedert sich gemäß [HL01, S.7ff] in die folgenden Phasen:

1. Definition des Problems und Sammlung relevanter Daten
2. Formulieren eines mathematischen Modells zur Repräsentation des Problems
3. Entwicklung einer rechnerbasierten Prozedur zur Lösung des Problems anhand des mathematischen Modells
4. Testen und Verfeinern des Modells
5. Vorbereitung des Modells für weitere vom Management bestimmte Anwendungen
6. Implementierung

Zu Beginn wird das vorliegende Problem genau erfasst und beschrieben. Dieser Schritt umfasst die Bestimmung realistischer Ziele, zeitlicher Limitierungen sowie alternativer Handlungsoptionen. Gleichzeitig werden Daten gesammelt und so aufbereitet, dass sie für die anschließende formale Modellierung genutzt werden können. Wird das Problem ungenau oder falsch definiert, besteht die Gefahr, dass am Ende eine „richtige“ Lösung für das „falsche“ Problem gefunden wird. Nachdem eine mathematische Repräsentation des Problems entwickelt wurde (Schritt 2), erfolgt die Implementierung eines Lösungsverfahrens. Analog zur herkömmlichen Entwicklung von Software muss auch das mathematische Modell getestet und ggf. korrigiert werden, bevor es im vorletzten Schritt in die vom Management vorgesehene Anwendung (z.B. als Komponente eines Entscheidungsunterstützungssystems) integriert wird. Die letzte Phase der Implementierung umfasst die Dokumentation und Inbetriebnahme der Anwendung, verbunden mit Schulungsmaßnahmen und finalen Tests.

Um sicherzustellen, dass die Ergebnisse jedes Teilschrittes inhaltlich übereinstimmen, überlappen sich die Phasen häufig oder es finden Rückkopplungen (insbesondere zur Schärfung des mathematischen Modells) statt.

2.1 Klassifikation von Optimierungsproblemen

Dieser Abschnitt beschreibt verschiedene Kriterien zur Unterteilung von Optimierungsproblemen. Zur Schaffung einer Klassifikationsgrundlage werden zunächst die Bestandteile eines allgemeinen Optimierungsmodells vorgestellt.
Das Lösen eines Optimierungsproblems kommt dem Treffen einer oder mehrerer Entscheidung(en) gleich, d.h. für jede **Entscheidungsvariable** $(x_1, x_2, \ldots, x_n)$ soll ein Wert bestimmt werden. Die **Ziel-** oder **Fitnessfunktion** ordnet einer Lösung, d.h. einer konkreten Belegung der Entscheidungsvariablen, einen Wert zu. **Restriktionen** (auch Nebenbedingungen, engl. *Constraints*) schränken die zulässigen Wertzuweisungen ein und werden in Form von Gleichungen und Ungleichungen definiert. Das eigentliche Problem besteht nun in der Belegung der Entscheidungsvariablen, sodass die Zielfunktion minimiert oder maximiert und das Restriktionensystem berücksichtigt wird. Dabei kann durch Umformung ein Maximierungsproblem in ein Minimierungsproblem (und umgekehrt) überführt werden.
Wird das Rucksackproblem an dieses allgemeine Modell angepasst, so ergibt sich die folgende Formulierung:

Maximiere

$$F(x) = \sum_{i=1}^{n} x_i w_i \tag{1.1}$$

mit

$$x = \begin{pmatrix} x_1 \\ \cdots \\ x_n \end{pmatrix} \tag{1.2}$$

unter den Nebenbedingungen

$$\sum_{i=1}^{n} x_i g_i \leq G \tag{1.3}$$

$$x_i \in \{0, 1\} \quad \text{für } i = 1, \ldots, n \tag{1.4}$$

In 1.1 wird die zu maximierende Zielfunktion $F(x)$ definiert. Die Nebenbedingungen fordern zum einen, dass die Kapazität des Rucksacks bezogen auf ein gegebenes Gewicht G nicht überschritten wird (1.3) und zum anderen, dass die durch einen Vektor repräsentierten Entscheidungsvariablen (1.2) binär belegt werden (1.4).

Die nachfolgende Klassifikation stützt sich auf mathematische Eigenschaften der Modellkomponenten sowie auf Eigenschaften der Lösungsfindung. Zu diesen zählen:

1. die Linearität der Zielfunktion und der Restriktionen
2. der Wertebereich der Entscheidungsvariablen
3. die Anzahl der Zielfunktionen
4. die Unterscheidung zwischen deterministischen und stochastischen Modellen
5. die Komplexität der Berechnung

zu 1.) Linearität der Zielfunktion und der Restriktionen

Ein *lineares Optimierungsproblem* ist dadurch gekennzeichnet, dass sowohl die Zielfunktion als auch die Restriktionen durch eine Menge linearer Gleichungen und Ungleichungen beschrieben werden. Dementsprechend werden bei einem *nichtlinearen Problem* die Zielfunktion oder ein Teil der Restriktionen durch nichtlineare Funktionen beschrieben. Ein Beispiel einer nichtlinearen Optimierung ist die aus [BZ12, S.237f.] entnommene Standortplanung[1]. Gesucht werden geeignete Standorte $(x_i, y_i),\ i = 1, \ldots, m$ für Lager, wobei die Lieferziele (z.B. Wohnorte oder Produktionsstätten) durch $(\alpha_j, \beta_j),\ j = i, \ldots, n$ gegeben sind. Ebenfalls bekannt sind die jeweiligen Bedarfe b_j der Abnehmer sowie die Kapazitäten a_i der Lager. Es ergibt sich das folgende zu lösende Minimierungsproblem:

Minimiere

$$\sum_{ij} d_{ij} z_{ij} \tag{2.1}$$

unter den Nebenbedingungen

$$\sum_{j} z_{ij} \leq a_i \quad \text{für } i = 1, \ldots, m, \tag{2.2}$$

$$\sum_{i} z_{ij} \leq b_j \quad \text{für } j = 1, \ldots, n, \tag{2.3}$$

$$z_{i,j} \geq 0 \quad \text{für } i = 1, \ldots, m;\ j = 1, \ldots, n. \tag{2.4}$$

Die Menge der Güter, welche vom i-ten Standort zum j-ten Abnehmer transportiert werden, wird mit $z_{i,j}$ bezeichnet. Die Entfernungen zwischen dem i-ten Standort und dem j-ten Abnehmer sind durch d_{ij} gegeben und können mit Hilfe der in Gleichung 3.1 dargestellten euklidischen Abstandsfunktion berechnet werden.

$$d_{ij} := \sqrt{(x_i - \alpha_j)^2 + (y_i - \beta_j)^2} \tag{3.1}$$

[1] Das in der Literaturquelle dargestellte Beispiel wurde hier leicht vereinfacht übernommen.

Ziel ist die Minimierung der Transportkosten, welche sich aus dem Produkt des zurückzulegenden Weges und der zu transportierenden Menge ergeben. Dabei dürfen weder die Kapazitäten der Lager a_i (2.2), noch die Bedarfe der Abnehmer b_j (2.3) überschritten werden. Diese Form der Standortbestimmung entspricht einem quadratischen Optimierungsproblem.

zu 2.) Wertebereiche der Entscheidungsvariablen

Bei der *ganzzahligen* linearen Optimierung (engl. *Integer Programming*) gilt - zusätzlich zur Linearität - die Restriktion, dass Entscheidungsvariablen ausschließlich mit ganzzahligen Werten belegt werden dürfen. Bezieht sich diese Forderung nur auf einen Teil der Variablen, handelt es sich um ein *gemischt-ganzzahliges* Problem (engl. *Mixed Integer Program*). Auch binäre Probleme, bei welchen die Entscheidungsvariablen entweder die Werte 0 oder 1 annehmen, zählen zu ganzzahligen Optimierungsproblemen. In der Praxis werden diese Bedingungen für Situationen formuliert, in welchen eine tatsächliche Entscheidung zu treffen ist (z.B. für oder gegen eine Investition) oder Ressourcen grundsätzlich nicht in Teilen (z.B. können nicht 4,5 Autos hergestellt werden) vorkommen. Die vorgestellten Probleme des optimalen Rucksack-Packens und Rundreisens werden als ganzzahlig linear klassifiziert. Darüber hinaus zählen sie zu den *diskreten* Optimierungsproblemen, welche durch endliche Wertebereiche der Variablen charakterisiert sind.

zu 3.) Anzahl der Zielfunktionen

Die bisherigen Beispiele zielen auf die Optimierung eines einzelnen Kriteriums ab. Bei einer Mehrzieloptimierung soll dagegen eine Lösung gefunden werden, welche die Werte mehrerer Zielfunktionen gleichzeitig optimiert. Formal wird dies durch die Umformung der ursprünglichen Zielfunktion $f : V \to \mathbb{R}$ in eine vektorwertige Zielfunktion $f : V \to \mathbb{R}^n$ beschrieben.
Ein anschauliches Beispiel eines Mehrzieloptimierungsproblems ist die Entscheidung für ein Transportmittel. Ein Auto bietet im Vergleich zur Reise mit einem Bus oder Zug ein hohes Maß an Privatsphäre und Komfort, ist aber häufig verbunden mit höheren Kosten. Ist die Umweltbelastung ein weiteres Kriterium, ist die Benutzung eines privaten Fahrzeugs schlechter zu bewerten als die Fahrt mit einem öffentlichen Verkehrsmittel. [Coh78]
In [ETV07] wird ein Optimierungsproblem mit einem erhöhten öffentlichen und wirtschaftlichen Interesse beschrieben. Dieses beschäftigt sich mit dem Finden einer Transportroute für Gefahrgut. Der ideale Transportweg soll u.a. möglichst kurz sein und gleichzeitig stark bevölkerte Gegenden sowie Straßen mit einem hohen Verkehrsaufkommen meiden. Diese sich oft widersprechenden Forderungen werden von Vertretern unterschiedlicher Interessengruppen formuliert und entspringen logistischen, wirtschaftlichen und sicherheitsbezogenen Überlegungen. Das Rucksackproblem wird zu einem Problem mit mehreren Zielstellungen,

wenn zusätzlich zur Wertmaximierung die Minimierung des Gewichts gefordert wird. Oft kann nicht eine optimale Lösung gefunden werden, welche die Werte aller Zielfunktionen gleichzeitig maximiert oder minimiert. Stattdessen wird eine Lösungsmenge (auch als *Pareto-Optimum* oder *-Menge* bezeichnet) gesucht, für deren Elemente gilt, dass eine Verbesserung eines Kriteriums zur Verschlechterung eines anderen Kriteriums führt. Bezogen auf das Rucksackproblem heißt das, dass die Pareto-Menge diejenigen Rucksäcke enthält, die so bepackt sind, dass es keine andere Bepackung gibt, welche leichter oder gleich schwer ist und trotzdem einen höheren Wert erreicht. Die Elemente der Lösungsmenge werden als nicht dominiert bezeichnet.
Alternativ zur Pareto-Optimierung können die einzelnen Zielfunktionen gewichtet und aggregiert werden. Das Finden geeigneter Wichtungen wird allerdings als schwierig eingeschätzt. [BHS07, S.73]

zu 4.) die Unterscheidung zwischen deterministischen und stochastischen Modellen

Sind die Parameter der Zielfunktionen und der Restriktionen (für das formulierte Rucksack-Problem bspw. die Gewichte g_i, die Werte w_i und das Maximalgewicht G) bekannt, wird das Optimierungsproblem als *deterministisch* klassifiziert. Im Gegensatz dazu umfasst ein *stochastisches* Modell zumindest einen Wahrscheinlichkeitswert. [DD05, S.6f.]

zu 5.) die Komplexität der Berechnung

Die wohl wichtigste Klassifikation erfolgt nicht anhand einer Modelleigenschaft, sondern anhand der Berechnungskomplexität. Die Untersuchung der Komplexität kann hinsichtlich der Zeit, des Speicherbedarfs oder anderer für die Berechnung benötigter Ressourcen erfolgen. An dieser Stelle soll die zur Lösung eines Problems benötigte Zeit die zu untersuchende Größe sein. Dabei wird unterschieden zwischen der Komplexität eines Algorithmus und eines Problems. Für einen Algorithmus kann durch das Zählen der Rechenschritte eine *obere Schranke* für die Lösung des zugrundeliegenden Problems ermittelt werden. Aufgrund dieser Schranke erfolgt eine Einteilung in *Komplexitätsklassen* unter Angabe der *O-Notation.* Ein Algorithmus wird demnach als *konstant, logarithmisch, linear, n-log-n, quadratisch, polynomiell* oder *exponentiell* klassifiziert. Die Komplexität eines Problems kann nur durch eine theoretische Analyse festgestellt werden. Ziel ist das Finden einer *unteren Schranke*, welche die Effizienz der bestmöglichen Berechnung beurteilt und damit aussagt, dass es keinen Algorithmus mit einem besseren Ressourcenverbrauch geben kann. Anhand dieser unteren Schranke wird ein Problem der Klasse **P** oder **NP** zugeordnet. Die Klasse P fasst diejenigen Probleme zusammen, welche in polynomieller Laufzeit gelöst werden können. Probleme,

welche von einer nichtdeterministischen Turingmaschine in polynomialer Zeit gelöst werden können, zählen zur Klasse NP. In praktischen Anwendungen ist diese Einstufung deshalb so bedeutend, da dadurch bestimmt wird, ob ein Problem *effizient lösbar* ist (P) oder nicht (NP). Derzeit ist noch unklar, ob es für NP-Probleme tatsächlich keine effizienten Algorithmen gibt oder ob diese noch nicht gefunden wurden. [Soc08, S.167ff.]

2.2 Kombinatorische Optimierung

Die Kombinatorische Optimierung beschäftigt sich mit dem Finden einer optimalen Lösung aus einer endlichen Menge von Kandidaten. Da diese Form der diskreten Optimierung eine hohe praktische Relevanz besitzt und auch das vorliegende Menüplanungsproblem dieser Kategorie zuzuordnen ist, präsentiert dieser Abschnitt weitere Anwendungsfälle.
In [DD05, S.120ff.] wird eine Unterteilung in **a.) Reihenfolgeprobleme, b.) Auswahlprobleme, c.) Gruppierungsprobleme** und **d.) Zuordnungsprobleme** vorgenommen.

zu a.) Reihenfolgeprobleme

Reihenfolgeprobleme sind dadurch charakterisiert, dass deren Lösungen stets Permutationen sind. So ist beim bereits vorgestellten Problem des Handlungsreisenden die Anordnung der Städte relevant. Bei *allgemeinen Tourenplanungssystemen* sollen mehrere Kunden beliefert werden, wobei für den Transport ebenfalls mehrere Fahrzeuge zur Verfügung stehen. Gesucht wird eine Route für jedes Fahrzeug, sodass alle Kunden beliefert werden und die Kosten (z.B. die Länge des zurückgelegten Weges) minimiert werden. [Dom95]
In [Ihr99, S.216] wird das Finden einer Auftragsreihenfolge für Maschinen als sogenanntes *Maschinenbelegungsproblem* formuliert. Ziel ist die rechtzeitige Anfertigung möglichst vieler Produkte bzw. die Minimierung der gesamten Terminabweichung von festgelegten Fertigstellungsterminen.
Ein weiteres Beispiel eines Reihenfolgeproblems ist das *Restoration Order Problem*. Nach schweren Naturkatastrophen oder Spannungszusammenbrüchen leiden Bewohner und ansässige Unternehmen in den betroffenen Gebieten unter dem Ausfall des Stromnetzes. Die Aktionen, welche notwendig sind, um die Versorgung wiederherzustellen, können nicht beliebig angeordnet werden, sondern müssen im Hinblick auf die Eigenschaften des Netzwerks geplant werden. Das Ergebnis der Optimierung ist eine störungsfreie Anordnung der Maßnahmen, sodass die Zeit des Ausfalls minimiert wird. Störungsfrei bedeutet, dass mit jedem Schritt die Stabilität des Netzwerks erhöht wird bzw. aufrecht erhalten bleibt. [MCVH$^+$14]

zu b.) Auswahlprobleme

Zu den Auswahlproblemen gehören beispielsweise die Mengenzerlegung (*Set Partitioning*), die Mengenüberdeckung (*Set Covering*) sowie das Rucksackproblem. Allen gemein ist das Finden einer optimalen Selektion von Elementen aus einer sehr großen Menge.

zu c.) Gruppierungsprobleme

Soll eine konkrete Instanz dieser Problemklasse gelöst werden, so wird stets nach einer Aufteilung bzw. Gruppierung von Elementen gefragt. Beispielsweise wird beim *Behälterproblem* (engl. *Bin Packing*) nach einer Aufteilung von Objekten auf eine bestimmte Anzahl von Behältern gesucht. Zu beachten sind die jeweiligen verschiedenen Größen der Objekte und die für alle Behälter gleiche Kapazität, sodass es bei einer Aufteilung nicht zu einem Überlaufen kommt. Bei der Optimierung wird nach der Aufteilung mit der geringsten Anzahl an benötigten Behältern gesucht. [Soc08, S.173]
Diese Probleme finden sich u.a. in der Industrie und der Medizin wieder. Beispielsweise wird in der Verschnittoptimierung ein zusammenhängendes Materialstück so zugeschnitten, dass so wenig Material wie möglich als Verschnitt übrig bleibt. [SSV11] Ein medizinisches Gruppierungsproblem lässt sich bei der Lebendspende von Nieren identifizieren. Benötigt eine Person eine Niere, sind Familienmitglieder oder nahe Bekannte oft bereit, eine ihrer Nieren zu spenden. Dieses Paar aus einem Spender und einem Empfänger ist möglicherweise nicht kompatibel und somit ist eine Spende in dieser Form ausgeschlossen. Gibt es allerdings ein weiteres solches Spender-Empfänger-Paar, so kann es möglich sein, dass diese zwei Paare *kreuzweise* spenden bzw. empfangen können. Der Spender des ersten Paares gibt dann eine seiner Nieren an den Empfänger des zweiten Paares und umgekehrt. In der Realität existieren Tausende solcher Paare, sodass ein *verketteter* Austausch, welcher mehrere Paare beinhaltet, vorstellbar wäre. Das Ziel ist die Maximierung der dadurch erreichten Lebendspenden. Eine theoretische Untersuchung dieser *Transplantationsketten* sowie deren Anwendbarkeit in der klinischen Praxis wird in [DPS12] präsentiert.

zu d.) Zuordnungsprobleme

Zu den wohl prominentesten Zuordnungsproblemen gehören die **Scheduling**-Verfahren. Scheduling bezeichnet die Erstellung von Plänen, zum Beispiel von Stundenplänen, Spielplänen oder Fahrplänen. Besondere wissenschaftliche Beachtung wurde in den letzten Jahren dem *Nurse Scheduling Problem (NSP)*[2] geschenkt. Gegenstand des NSP ist das Generieren eines Dienstplans für Krankenschwestern. Die Restriktionen werden in harte und weiche

[2] auch *Nurse Rostering Problem* genannt

Kriterien unterteilt, wobei das Verstoßen gegen ein hartes Kriterium dazu führt, dass ein Dienstplan als Lösung unzulässig wird. Weiche Restriktionen hingegen dürfen missachtet werden - mit dem Ergebnis, dass der Dienstplan schlechter bewertet wird. Beispielsweise wird ein Dienstplan nur akzeptiert, wenn der für eine Schicht festgelegte Personalbedarf (hartes Kriterium) erfüllt wird. Werden überdies die individuellen Präferenzen der Mitarbeiter (weiches Kriterium) berücksichtigt, kann der Dienstplan als optimal angesehen werden. Eine umfassende Betrachtung des NSP sowie verschiedener Lösungsansätze ist in [BDCBVL04] gegeben.
In der Praxis kommt die Behandlung solcher Scheduling-Probleme einem kombinatorischen Albtraum gleich. Dies zeigt eine im Jahr 1982 durchgeführte Studie der Fluggesellschaft United Airlines. [HB86] Ziel war die Minimierung der Kosten durch eine verbesserte Personalplanung für über 5000 Mitarbeiter, welche in Reservierungsbüros und an Flughäfen tätig sind. Teilzeitangestellte arbeiten in Schichten zwischen zwei und acht Stunden, während Mitarbeiter in Vollzeit Schichten zwischen acht und zehn Stunden absolvieren. Schichten starten zu unterschiedlichen Zeiten und den Angestellten stehen eine 30-minütige Mittagspause sowie kurze Erholungsphasen (etwa alle zwei Stunden) zu. Die Reservierungsbüros sind - ebenso wie jeder größere Flughafen - durchgängig besetzt. Es wurde festgestellt, dass die Anzahl der benötigten Mitarbeiter stark fluktuiert und häufig halbstündlich angepasst werden muss. Das vollständige Optimierungsmodell berücksichtigte noch weitere Restriktionen und definierte ca. 20.000 Entscheidungsvariablen. Das *Station Manpower Planning System* führte zu jährlichen Ersparnissen von sechs Millionen Dollar und die Optimalität der erzeugten Dienstpläne wurde deutlich im Betrieb wahrgenommen.

2.3 Lösungsverfahren

Optimierungsprobleme lassen sich auf vielfältige Art und Weise lösen. Dieser Abschnitt stellt einige Ansätze und Algorithmen vor und betrachtet, wann diese anwendbar sind und wie effizient (bezogen auf das Laufzeitverhalten) die Berechnung stattfindet. Da diese Faktoren abhängig von der Problemklasse und der gewählten Modellierung sind, ist das Treffen einer allgemeinen Aussage oft erschwert bzw. nicht möglich.
Optimierungsverfahren sind äußert vielfältig motiviert und entspringen den Forschungsaktivitäten unterschiedlicher Disziplinen. Die Methodiken reichen von rein mathematischen Verfahren (z.B. Lineare Programmierung (LP) oder Goal Programming (GP)) über wissensbasierte Ansätze wie Expertensysteme oder fallbasiertes Schließen bis hin zu von der Natur inspirierten Lösungsstrategien. Es sei daher darauf hingewiesen, dass die nachfolgende

Darstellung keinesfalls lückenlos ist.

2.3.1 Exakte Verfahren

Existiert eine optimale Lösung für ein gegebenes Problem, so wird diese beim Einsatz eines exakten Verfahrens gefunden. Das Auffinden einer Lösung ist garantiert, da diese Verfahren auf dem gesamten Lösungsraum operieren.

Erschöpfende Suche

Bei diesem auch als *Brute Force* bekannten Verfahren wird der gesamte Suchraum durchlaufen. Aufgrund seiner Simplizität ist dieser Algorithmus sehr einfach zu implementieren. Der Rechenaufwand wächst allerdings proportional zur Anzahl der zu probierenden möglichen Lösungen, wobei diese zumeist exponentiell wächst.

Branch and Bound

Im Gegensatz zum Brute-Force Vorgehen wird eine optimale Lösung gefunden, indem nur ein kleiner Teil des möglichen Lösungsraums durchsucht wird. *Branching* bezeichnet das Aufspalten des Problems in Teilprobleme; übertragen auf einen Entscheidungsbaum entspricht ein Teilproblem einem Knoten des Baums. Mittels *Bounding* wird eine Schranke berechnet, anhand derer entschieden wird, ob ein Teilproblem weiter verzweigt wird oder nicht. Steht fest, dass durch eine weitere Enumeration keine bessere als die bisherig gefundene beste Lösung produziert werden kann, wird das Teilproblem nicht weiter betrachtet. Zur Berechnung der Schranke kommen häufig problemspezifische Heuristiken oder Relaxationsverfahren (s.u.) zum Einsatz. [DD05, S.126ff.] Durch dieses Vorgehen kann - abgesehen vom ungünstigsten Fall - eine verbesserte Laufzeit gegenüber der Erschöpfenden Suche erreicht werden. Entscheidende Faktoren, welche die Effizienz beeinflussen, sind die Wahl des aktuell zu untersuchenden Teilproblems und der Heuristik, welche zur Initialisierung der ersten oberen Schranke genutzt wird. [KV12, S.624]
Die Erschöpfende Suche und das Branch and Bound Verfahren sind bei ganzzahligen, kombinatorischen Optimierungsproblemen anwendbar.

Relaxationsverfahren

Die Relaxation von Problemen beschreibt das Auflösen derjenigen Nebenbedingungen, welche ein Problem schwer lösbar machen. Beispielsweise führt eine LP-Relaxation (LP für Lineare Programmierung) zum Aufgeben der Ganzzahligkeitsbedingungen von Variablen. Gilt für

das Grundproblem die binäre Bedingung $x_i \in \{0, 1\}$, so wird diese nach der Relaxation zu $0 \leq x_i \leq 1$. Durch Algorithmen der linearen Optimierung kann nun eine Lösung für dieses veränderte Problem gefunden werden. Diese Lösung gilt zwar nicht für das ursprüngliche Problem, lässt aber Rückschlüsse zu oder kann als Näherungslösung herangezogen werden. [DD05, S.132ff]

Dynamische Programmierung

Die Dynamische Programmierung bezeichnet einen generellen Ansatz der Problemlösung, bei welchem zunächst eine optimale Lösung für einen kleinen Ausschnitt des Gesamtproblems bestimmt wird. Das Teilproblem wird anschließend schrittweise vergrößert, wobei die optimale Lösung des aktuellen Schrittes auf Basis der Lösung des vorherigen Schrittes bestimmt wird. Dies wird so oft wiederholt, bis das Gesamtproblem gelöst ist.
Dieser Ansatz kann nur dann angewendet werden, wenn das Problem gewisse Eigenschaften erfüllt. Eine diese Eigenschaften ist das *Prinzip der Optimalität*, welches fordert, dass eine optimale Lösung für ein Problem effizient aus den optimalen Lösungen der Teilprobleme konstruiert werden kann. [HL01, S.563ff.] Ein solches Problem besitzt dann eine *optimale Substruktur* (engl. *optimal substructure*). [CLRS09, S.379]

2.3.2 Heuristische Verfahren

Die Anwendung exakter Verfahren führt bei vielen praktischen Problemen zu inakzeptablen Laufzeiten. Eine Alternative stellen dann die heuristischen Verfahren dar. Diese garantieren nicht das Auffinden des Optimums, führen aber dennoch zu guten zulässigen Lösungen. Es besteht auch die Möglichkeit, ein exaktes Verfahren mit einer Heuristik zu kombinieren. In diesem Fall wird eine optimale Lösung gefunden, sofern diese existiert.
Heuristiken beschreiben *Vorgehensregeln*, die die Exploration des Suchraums für ein bestimmtes Problem steuern. Bezogen auf eine Baumstruktur bedeutet das, dass ein Knoten mittels einer heuristischen Funktion bewertet wird. Diese Funktion liefert eine Einschätzung darüber, wie Erfolg versprechend die weitere Bearbeitung des Knotens scheint. [BHS07, S.30ff.] Heuristische Verfahren können grundsätzlich eingeteilt werden in

1. Konstruktionsverfahren und
2. Lokale Suche bzw. Verbesserungsverfahren

Konstruktionsverfahren dienen der schrittweisen Entwicklung zulässiger Lösungen. Dabei kann eine *gierige* (engl. *greedy*) oder eine *vorausschauende* Strategie verfolgt werden. Bei ersterer wird stets derjenige Knoten expandiert, welcher zur größtmöglichen Verbesserung

des Zielfunktionswertes der bisherigen Teillösung führt.
Bezogen auf das binäre Rucksackproblem können verschiedene Heuristiken zur Konstruktion betrachtet werden. Beispielsweise kann es sinnvoll sein, möglichst viele Gegenstände einzupacken, sodass ein gieriger Algorithmus in jedem Schritt immer den Gegenstand mit dem geringsten Gewicht auswählen wird. Eine andere Heuristik kann das Verhältnis von Wert und Gewicht betrachten und "packt" in jedem Schritt den Gegenstand mit der höchsten Wertdichte in den Rucksack.
Im Gegensatz zu diesem gierigen Vorgehen schätzt ein vorausschauendes Verfahren ab, wie sich eine Variablenbelegung auf die in den zukünftigen Schritten erzielbare Lösungsgüte auswirkt. Ein Beispiel für ein vergleichsweise aufwendiges Verfahren mit sehr hoher Lösungsgüte ist die Vogel'sche Approximations-Methode. [DD05, S.84ff., S.127ff.] Ein weiterer Vertreter der Konstruktionsverfahren ist die im nachfolgenden Abschnitt präsentierte Constraint Programmierung.
Ausgangspunkt für die Anwendung eines **Verbesserungsverfahrens** ist das Vorhandensein einer zulässigen Lösung. Diese kann zufällig gefunden werden oder das Ergebnis eines Konstruktionsverfahrens sein. In jedem Schritt wird aus der Nachbarschaft dieser Lösung eine Lösung als Eingabe für die nächste Iteration ausgewählt. Benachbarte Lösungen entstehen durch *Transformationen*, bei welchen u.a. eine Stelle der Lösung verändert, ein Element vertauscht oder ein Element verschoben wird. Eine Veränderung einer Stelle der Lösung lässt sich gut auf Optimierungsprobleme mit binären Wertebereichen anwenden. Beim Rucksack-Problem ergeben sich benachbarte Lösungen beispielsweise durch das Ein- bzw. Auspacken eines Gegenstands. Elementvertauschungen und -verschiebungen eignen sich zum Beispiel zur Definition von Nachbarschaften bei Reihenfolgeproblemen wie dem TSP. Grundsätzlich sind beliebige Nachbarschaftsbeziehungen vorstellbar. Jedoch sollte beachtet werden, dass die Anzahl der in jeder Iteration entstehenden Nachbarn nicht zu hoch ist. [BHS07, S.42]
Neben der Wahl einer Nachbarschaftsdefinition muss bei Verbesserungsverfahren ferner entschieden werden, in welcher Reihenfolge die Nachbarlösungen untersucht werden und welche Lösung als Eingabe für die nächste Iteration gewählt wird.[3] [DD05, S.129] *Reine Verbesserungsverfahren* (z.B. das Bergsteigerverfahren) terminieren, sobald keine verbessernde Lösung in der Nachbarschaft gefunden wird. Die Gefahr solcher Verfahren besteht darin, dass lediglich lokale Optima gefunden werden und zwischenzeitliche Verschlechterungen - die möglicherweise eine Nachbarschaft mit dem globalen Optimum eröffnen - nicht erlaubt sind. [DD05, S.128ff.] Weiterentwicklungen der reinen Verbesserungsverfahren heben diese Einschränkung auf und ermöglichen auch verschlechternde Züge. Diese als *Metaheuristiken*

[3] Beispielsweise bezeichnet *First fit* die Wahl der als erstes gefundenen verbessernden Lösung, während beim *Best fit* alle Nachbarn untersucht und mit der insgesamt besten Lösung fortgefahren wird.

bezeichneten Verfahren sind nicht an ein spezielles Problem gebunden und können z.B. auch auf ganzzahlige, nichtlineare Probleme angewendet werden. [HL01, S.604] Eine bedeutende Gruppe der Metaheuristiken stellen die *naturanalogen* Verfahren dar. Die Art und Weise, mit welcher der Suchraum erforscht wird, ist bei diesen Algorithmen motiviert von natürlichen a.) evolutionären, b.) physikalischen oder c.) neuronalen Vorgängen. Wichtige Vertreter von a.) sind die *Evolutionären Algorithmen*, welche im nachfolgenden Abschnitt näher beschrieben werden. Zu den physikalisch inspirierten Verfahren zählen das *Simulierte Abkühlen*, das *Toleranzschwellenverfahren* sowie der aus letzterem abgeleitete *Sintflut-Algorithmus*. Der Arbeitsweise des menschlichen Gehirns nachempfunden sind die in c.) eingeordneten *Neuronalen Netze* und die *Tabu-Suche*. [Fel99, S.71]

2.3.3 Evolutionäre Algorithmen

Evolutionäre Algorithmen bilden eine Klasse von Verfahren zur Lösung von Optimierungsproblemen nach dem Vorbild der biologischen Evolution. Die ersten Ideen dieser Art gehen zurück auf die späten 50er Jahre, beispielsweise auf [Box57] oder [Fra57]. Im Laufe der vergangenen Jahrzehnte entwickelten sich daraus vier Hauptrichtungen, namentlich sind dies *Evolutionäre Programmierung (EP), Evolutionsstrategien (ES), Genetische Algorithmen (GA)* sowie *Genetisches Programmieren (GP)*. Allen gemein ist, dass sie sich der Terminologie der Evolution bedienen und dabei die Begrifflichkeiten im Kontext der rechnergestützten Lösung von Optimierungsproblemen verändert interpretieren. Ein **Individuum** entspricht einer Problemlösung und kann in Form eines **Chromosoms** unterschiedlich repräsentiert werden. Zu den gängigen *Kodierungen* zählen Automaten (bei EP), Vektoren reeller Zahlen (bei ES), Bitvektoren (bei GA) und Bäume (bei GP). Diese formalen Strukturen entsprechen dem **Genotyp**. Durch eine **Dekodierung** wird der **Phänotyp** erzeugt, welcher der Darstellung im zu durchsuchenden Raum entspricht. Eine Menge von Individuen wird als **Population** bezeichnet. Die **Fitness** entspricht der Güte (oder Qualität) eines Individuums bezogen auf ein formuliertes Optimierungsziel. Im Gegensatz zum biologischen Pendant wird die Fitness in den meisten Fällen durch eine Ziel- oder Fitnessfunktion quantifiziert. Sie kann allerdings auch das Ergebnis einer subjektiven Bewertung sein.
Jeder evolutionäre Algorithmus durchläuft den in Abbildung 2.2 dargestellten Zyklus. Zu Beginn wird eine initiale Population gebildet. Diese kann zufällig erzeugt werden oder das Ergebnis anderer Optimierungsverfahren sein. Im nächsten Schritt werden die Individuen der Population anhand ihrer Phänotypen durch Anwendung der Zielfunktion bewertet. Die Werte dieser qualitativen Einschätzung bestimmen, welche Individuen als Eltern ausgewählt werden (*Paarselektion*). Die darauf folgende Rekombination erzeugt aus den selektierten

Eltern ein oder mehrere Nachkommen. Getreu dem biologischen Prinzip der Vererbung soll dadurch die Weitergabe und Vermischung von vielversprechendem Genmaterial erreicht werden. Die Kindindividuen werden anschließend einer Mutation unterzogen. Durch diese zufällige Veränderung kann theoretisch jeder Teil des Suchraums erreicht werden. Von Relevanz ist dabei das Ausmaß der Veränderung, welches von einer Feinabstimmung bis hin zu einer Makromutation (bei welcher der Geno- und Phänotyp stark verändert werden) reicht. Die Nachkommen, welche durch die zwei genetischen Operatoren Rekombination und Mutation entstehen, werden ebenfalls durch die Zielfunktion bewertet.

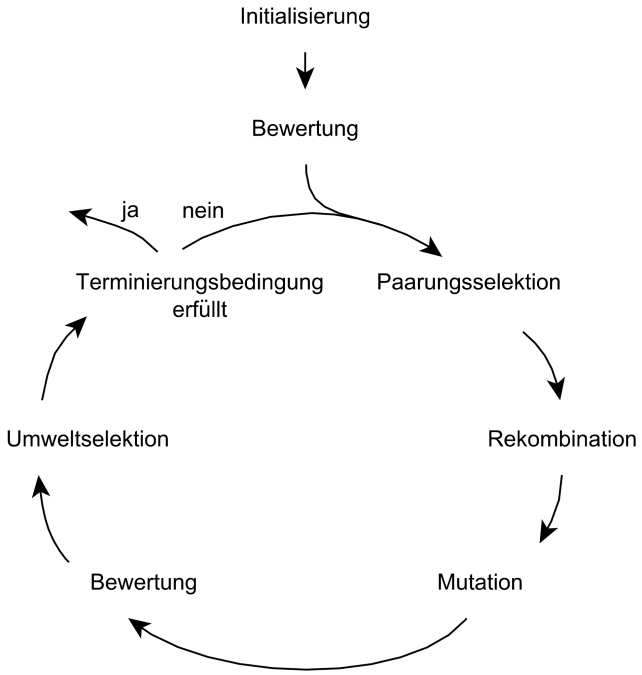


Abbildung 2.2: Schematische Darstellung des Ablaufs evolutionärer Algorithmen aus [Wei02, S.43]

Die zweite Selektion des Prozesses ersetzt nun einen Teil der Eltern oder die gesamte Elternpopulation durch ausgewählte Nachkommen. Die simulierte Evolution ist beendet, wenn eine vorherig definierte Terminierungsbedingung erfüllt wird. Diese kann beispielsweise durch einen zu erreichenden Gütewert des besten Individuums oder durch eine maximale Anzahl an Durchläufen gegeben sein.
Ein evolutionärer Algorithmus verspricht nur dann eine gute Lösung für ein Optimierungsproblem, wenn die Annahme gilt, dass ähnliche Individuen ähnliche Fitnesswerte besitzen. Bezogen auf den nachempfundenen Evolutionsprozess bedeutet das, dass die Kinder von *fitten Eltern* ebenfalls eine gute Bewertung durch die Zielfunktion erhalten. Diese Voraussetzung muss deshalb erfüllt sein, weil alle evolutionären Algorithmen in der Nachbarschaft guter Lösungen operieren bzw. aus dieser Individuen für die nächste Generation selektieren.

Für kleine Dimensionen bietet sich die Visualisierung des Lösungsraums in Form einer **Fitnesslandschaft** an.

Von den vier erwähnten Richtungen eignen sich insbesondere die *Genetischen Algorithmen* zur Lösung kombinatorischer Optimierungsprobleme. Diese gehen zurück auf John Holland, der adaptive Vorgänge in komplexen natürlichen und künstlichen Systemen untersuchte. [Hol75] Zur Anwendung eines genetischen Algorithmus müssen die folgenden Parameter festgelegt werden:

Kodierung eines Individuums

Mit der Kodierung eines Individuums soll die Frage nach der bestmöglichen Repräsentation einer Lösung für ein vorliegendes Problem beantwortet werden. Dabei gelten *Bitvektoren* als klassisches Repräsentationsschema; jedoch sind auch problemspezifische Kodierungen möglich. Das Finden einer *optimalen* Kodierung wird gemäß [OVGB10] als besonders schwierig - wenn nicht sogar unmöglich - eingeschätzt.

Fitnessfunktion

Die Fitnessfunktion weist den Elementen des Suchraums eine üblicherweise positive reelle Zahl zu. Diese dient als Gütekriterium und beeinflusst maßgeblich das "Überleben" eines Individuums getreu dem *Survival of the Fittest*-Prinzip. Für ein Chromosom wird unter Berücksichtigung der Gesamtfitness der Population die Überlebenswahrscheinlichkeit (auch *Auswahlwahrscheinlichkeit*) berechnet.

Selektionsstrategie

Es existieren verschiedene Ansätze zur Auswahl von vielversprechenden Chromosomen im Rahmen der Paar- und Umweltselektion. An dieser Stelle seien die *Roulette-*, *Rangbasierte* und *Turnierselektion* genannt. Diese Selektionsarten werden in [BHS07, S.74f.] näher beschrieben. Empfohlen wird die zusätzliche Anwendung der *Elite-Strategie*, bei welcher die besten n Individuen direkt in die nächste Population übernommen werden. Ebenfalls sinnvoll kann die Festlegung eines konstanten oder mit jeder Generation wachsenden Selektionsdrucks sein.

Genetische Operatoren

Durch die genetischen Operatoren *Mutation* und *Crossover* werden die selektierten Chromosomen verändert. Mutationen arbeiten auf einem Chromosom und können auf Bitebene (durch Kippen eines Bits) oder auf Parameterebene (durch Vertauschungen oder Verschiebungen) angewandt werden. Durch die entstehenden Abwandlungen vorhandener Individuen

wird das Genmaterial reichhaltiger und bisher unbekannte Bereiche des Suchraums können erschlossen werden. Dabei wird der Anteil der für die Mutation vorgesehenen Gene über die *Mutationsrate* bestimmt.
Als Crossover wird die Kombination von zwei oder mehreren Eltern-Chromosomen zu zwei oder mehreren Nachkommen bezeichnet. Es werden verschiedene Ausprägungen des Crossovers unterschieden, zum Beispiel das *1-, 2-* und *n-Punkt-Crossover*, das *uniforme* und *arithmetische* Crossover sowie das *Partially Matched Crossover (PMX)*, das *Ordered Crossover (OX)* und das *Cycle Crossover (CX)*. Wie bei der Mutation sind einige Crossover-Verfahren auf Bit- und/oder Parameterebene anwendbar.
Verschiedene Forschungsergebnisse geben Aufschluss darüber, wann die jeweiligen Operatoren den evolutionären Prozess erfolgreich vorantreiben oder diesen behindern. Grundsätzlich wird der Rekombination bei genetischen Algorithmen eine mächtigere Bedeutung zugesprochen als der Mutation. In [Spe95] und [Spe92] wurden positive Auswirkungen von Mutationen in kleinen Populationen und in sich ändernden Fitnesslandschaften beobachtet. [Wei02, S.42ff.] [BHS07, S.70ff.] [GKK04, S.12]

2.3.4 Constraint Programmierung

> "Constraint programming represents one of the closest approaches computer science has yet made to the Holy Grail of programming: the user states the problem, the computer solves it."
>
> Eugene C. Freuder in [Fre97]

Die Constraint Programmierung gilt als *Computational Paradigm* und entstand als solches in der Mitte der 80er Jahre durch die Einbettung des Constraint-Lösens in die Logikprogrammierung. Durch die Vereinigung dieser beiden deklarativen Verfahren kann insbesondere für kombinatorische Probleme, welche aufgrund ihrer exponentiellen Laufzeit praktisch unlösbar sind, eine effiziente Lösung gefunden werden.
In diesem Zusammenhang wird die Definition von Constraintproblemen (engl.: constraint satisfaction problem (CSP)) relevant. Diese auch als Zuordnungs- oder Beschränkungsprobleme bezeichneten speziellen Suchprobleme besitzen die folgenden Komponenten:

- eine endliche Menge von Variablen V
- Wertebereiche (auch Domäne genannt) für jede Variable $v \in V$
- eine endliche Menge von Beschränkungen (engl. *Constraints*), welche für eine Untermenge der Variablen die erlaubten Wert-Kombinationen durch Formeln beschreiben

Eine Menge von Wertzuweisungen zu Variablen (kurz *Zuordnung*) wird als *Zustand* bezeichnet. Eine *konsistente* Zuordnung verletzt keine der spezifizierten Beschränkungen und eine *vollständige* Zuordnung beinhaltet Wertzuweisungen für alle Variablen. Die Lösung eines Constraintproblems ist eine vollständige, konsistente Zuordnung. Jede Lösung eines solchen CSP befindet sich im Suchbaum in der Tiefe $|V|$. Dies entspricht auch der maximalen Tiefe des Baums. Aus diesem Grund hat sich der Einsatz der Tiefensuche zur Exploration des Suchbaums für Constraintprobleme bewährt. Ein Zustandsübergang entspricht einer Variablenbelegung und es gilt, dass die Reihenfolge der Belegungen irrelevant ist; d.h. Vertauschungen verändern die Lösung nicht. Allerdings kann sich eine ungünstige Reihenfolge negativ auf die Rechenzeit auswirken.
Bei der Behandlung von Zuordnungsproblemen ist es sinnvoll, die Abhängigkeiten der Constraints von den Variablen zu untersuchen. Eine einstellige Beschränkung bezieht sich nur auf eine Variable, währenddessen ein zweistelliges Constraint Aussagen über erlaubte Relationen zwischen zwei Variablen formuliert. Dementsprechend beziehen sich höhergradige Beschränkungen auf drei oder mehr Variablen. Dabei ist zu beachten, dass jede höhergradige Beschränkung eines endlichen CSP durch eine Menge binärer Beschränkungen ausgedrückt werden kann, wenn genügend Hilfsvariablen eingeführt werden. Besteht ein CSP ausschließlich aus zweistelligen Constraints, wird es auch als *binäres CSP* bezeichnet. Um die Variablenabhängigkeiten solcher Zuordnungsprobleme zu veranschaulichen, eignet sich ein Constraintnetz. [BHS07, S.49ff.]
Im Gegensatz zu Algorithmen, welche einen Zustandsraum strategisch durchsuchen, können Algorithmen zur Lösung eines CSP in jedem Schritt aus zwei Aktionen wählen: entweder eine Variable wird mit einem Wert belegt oder die *Constraint Propagierung* wird angewendet. Diese besondere Form der Inferenz nutzt die Restriktionen, um die zulässigen Wertebereiche der Variablen weiter einzuschränken. In günstigen Fällen werden die Wertebereiche durch eine Propagierung bereits so eingeschränkt, dass das Problem vollständig gelöst wird.
Ein Constraintproblem kann neben absoluten Constraints - deren Verletzung zu einem inkonsistenten Zustand führt - auch präferierte Constraints umfassen. Diese beschreiben einen bevorzugten Lösungszustand und können oft durch Kosten für gewisse Wertzuweisungen repräsentiert werden. Ein Problem mit dieser Ausprägung wird als *Constraint Optimization Problem* bezeichnet.
[RN09, S.137ff.]

3 Konzeption der optimierten Menüplanung

3.1 Ähnliche Arbeiten

In diesem Abschnitt werden Arbeiten vorgestellt, dessen Betrachtung bei der Lösungsfindung des Menüplanungsproblems sinnvoll erscheint.
Zu den wohl ältesten Optimierungsproblemen im Ernährungsbereich zählt das von *Stigler* im Jahre 1945 formulierte Ernährungsmodell. Gesucht wurde eine Auswahl von Lebensmitteln, welche den mittleren täglichen Bedarf an Kalorien, Eiweiß, Kalzium, Eisen, den Vitaminen $A,\ B_1,\ B_2,\ C$ und Niacin eines normal gewichtigen, mäßig aktiven Mannes deckt und dabei hinsichtlich der Kosten minimal ist. Stigler formulierte ein lineares Optimierungsproblem, welches im Jahre 1947 von J. Laderman mit Hilfe des Simplex-Algorithmus innerhalb von 120 Personentagen gelöst werden konnte. [Dan66] Das Ergebnis war eine Zusammenstellung von 10 Lebensmitteln, welche jährlich 36,64$ kostete. Modelle dieser Art finden sich gegenwärtig in der Viehhaltung wieder. [BZ12, S.11]
Ein weiterer mathematischer Lösungsansatz für ein Menüplanungsproblem wird in [Bal64] vorgestellt. Für eine festgelegte Anzahl von Tagen werden Gerichte so geplant, dass Nährstoffvorgaben eingehalten und ein gewisser Grad an Variabilität erreicht wird. Die Lösung soll möglichst kostengünstig sein.
In [CA15] wird ein System zur automatischen Erstellung von Menüplänen präsentiert. Die Anwendung ist sowohl für den Einsatz in Krankenhäusern als auch für entlassene Patienten vorgesehen. Das Planungsproblem wird in Form eines *Constraint Satisfaction Problems* beschrieben und mit Hilfe des *Choco Solvers* [PFL16] gelöst. Das Wissen zu Rezepten, Menüs, bestimmten Krankheiten und jeweilig nicht verzehrbaren Lebensmitteln wird durch eine Datenbank bereitgestellt. Die Autoren formulieren Bedingungen zur Einhaltung von Nährstoffgrenzen bei bestimmten Krankheitsbildern. Entlassene Patienten können die Planung durch weitere Restriktionen zu nicht verzehrbaren Lebensmitteln personalisieren. Das vollständige CSP zur Erstellung eines Menüplans für vier Wochen wird nicht vorgestellt. Ebenfalls unklar sind die genaue Anzahl der infrage kommenden Rezepte sowie die berücksichtigten Krankheiten. Für letztere werden u.a. Diabetes, Nierenerkrankungen und Gicht

genannt. Es wird ausgesagt, dass das System hinsichtlich der Variabilität der erstellten Pläne und der Planungsdauer eine Verbesserung gegenüber der manuellen Planerstellung darstellt. Der Einsatz von evolutionären Algorithmen zur Lösung eines Menüplanungsproblems mit mehreren Zielen und Restriktionen wird in [Sel09] präsentiert. Das Problem wird zunächst als mehrdimensionales Rucksackproblem formuliert und anschließend mit Hilfe des *Nondominated Sorting Genetic Algorithm (NSGA-II)* [DPAM02] gelöst. Lösungen, welche nicht alle Restriktionen erfüllen, werden durch Verfahren der linearen Programmierung und Ersetzungen "repariert".

Ein wissensbasiertes System, welches sowohl Fall- als auch Regelwissen verarbeitet, ist der in [MPS99] vorgestellte *CAse-based Menu Planner Enhanced by Rules (CAMPER)*. Das Ergebnis des in CAMPER verfolgten Planungsprozesses ist ein tägliches Menü für eine einzelne Person, wobei diätische Anforderungen und persönliche Präferenzen berücksichtigt werden. Erstere werden durch numerische Restriktionen repräsentiert (z.B. das Menü soll einen Snack, nicht weniger als 1800 und nicht mehr als 2000 Kalorien und mindestens 800 mg Kalzium beinhalten). Die fallbasierte Komponente des Systems wählt aus einer Falldatenbank eine Lösung aus und adaptiert diese, bis alle Restriktionen erfüllt sind. Anschließend kann der Nutzer interaktiv Veränderungen am Plan vornehmen. Möchte ein Nutzer eine Ersetzung vornehmen, werden ihm Vorschläge präsentiert, welche regelbasiert abgeleitet wurden. Das Erschließen geeigneter Ersetzungen fußt auf dem Wissen, welche *Rollen* ein Lebensmittel einnehmen kann und wie bestimmte Lebensmittel zu Gerichten zusammengefasst werden können. Diese Vorschläge können auch zu einer Verletzung der Restriktionen führen. Diese Form der Planung unterscheidet sich von den vorherigen Ansätzen, da es nicht darum geht, den Wert einer Zielfunktion zu minimieren oder maximieren. Vielmehr arbeitet das System entscheidungsunterstützend und überlässt dem Nutzer die eigentliche Optimierung.

3.2 Formalisierung des Menüplanungsproblems

Dieser Abschnitt präsentiert eine formale Beschreibung des Menüplanungsproblems. Das zur Lösung benötigte Wissen kann in

A Plan-unabhängiges Wissen und

B Plan-abhängiges Wissen

unterteilt werden.

zu A.) Plan-unabhängiges Wissen

Das Plan-unabhängige Wissen umfasst diejenigen Informationsobjekte, welche unspezifisch

für alle Menüplanungsprobleme sind. Dieses Wissen wird wie folgt formalisiert:

Indexe

g Index für Gerichte

l Index für Lebensmittel

n Index für Nährstoffe

e Index für Einschränkungen

z Index für Zubereitungsformen

Mengen

G die Menge der Gerichte

L die Menge der Lebensmittel

N die Menge der Nährstoffe (z.B. Kalorien, Kohlenhydrate, Fette, Laktose)

E die Menge der Einschränkungen (z.B. Lebensmittelallergien und -intoleranzen (A5), diätische Trends (A15) oder Ernährungsweisen (A4)), welche einen Verzehr bestimmter Lebensmittel ausschließen

Z die Menge der Zubereitungsformen (z.B. gekocht, gegrillt)

A_g die Menge der im Gericht g enthaltenen Lebensmittel

Parameter

$a_{g,l}$ die in einem Gericht g enthaltene Menge des Lebensmittels l in Gramm; $a_{g,l} \geq 0$

$c_{l,n}$ die in 100g eines Lebensmittels l enthaltene Menge des Nährstoffes n in Gramm; $c_{l,n} \geq 0$

$d_{g,n}$ die in einem Gericht g enthaltene Menge des Nährstoffs n in Gramm; $d_{g,n} = \sum_{l \in L} a_{g,l} \cdot \frac{c_{l,n}}{100}$

$f_{g,e}$ gibt an, ob ein Gericht g bei einer Einschränkung e verzehrt werden darf; $f_{g,e} \in \{0$ (darf nicht verzehrt werden), 1 (darf verzehrt werden)$\}$

$h_{g,z}$ gibt an, ob ein Gericht g entsprechend der Zubereitungsform z zubereitet wird; $h_{g,z} \in \{0$ (g wird entsprechend z zubereitet), 1 (g wird nicht entsprechend z zubereitet)$\}$

i_g die bei der Zubereitung einer Portion des Gerichts g anfallenden Kosten in Euro; $i_g \geq 0$

j_g die zur Zubereitung des Gerichts g benötigte Zeit in Minuten; $j_g \geq 0$

ps_l die Menge einer herkömmlichen Packungsgröße des Lebensmittelproduktes l in Gramm; $ps_l > 0$

db_l gibt an, ob das Lebensmittel l haltbar ist oder nicht; $db_l \in \{1 \text{ (haltbar)}, 0 \text{ (nicht haltbar)}\}$

zu B.) Plan-abhängiges Wissen

Diese Form des Wissens ist spezifisch für einen zu erstellenden Menüplan. Dies bedeutet, dass neben Parametern zur Individualisierung eines Plans[1] auch die von den Nutzern getätigten Angaben zu ernährungsbedingten Einschränkungen sowie präferierten und nicht präferierten Lebensmitteln zu dieser Kategorie zählen.

Indexe

p Index der Nutzer

b Index der Bewertungsmöglichkeiten

t Index der Tage

m Index der Mahlzeiten

Mengen[2]

P die Menge der Nutzer eines Plans; ein Nutzer ist eine Person, welche mindestens eine Mahlzeit des zu erstellenden Plans zu sich nimmt

B die Menge von Bewertungsmöglichkeiten zur Angabe der Präferenz eines Lebensmittels; $B = \{1 \text{ (überhaupt nicht schmackhaft)}, 2 \text{ (wenig schmackhaft)}, 3 \text{ (neutral)}, 4 \text{ (schmackhaft)}, 5 \text{ (sehr schmackhaft)}\}$

T die Menge der zu planenden Tage

M_t die Menge der für einen Tag t zu planenden Mahlzeiten

$Slots$ die Menge der zu belegenden Mahlzeiten; $Slots = \bigcup_{t \in T} \{T \times M_t\}$, wobei ein Element (t, m) die Mahlzeit m am Tag t bezeichnet

[1] Individualisierung eines Plans meint zum Beispiel die Angabe der Tage und der jeweiligen Mahlzeiten, für welche eine Belegung mit Gerichten erfolgen soll.

[2] Beschriebene Mengenbildungen nehmen Bezug auf die im Anschluss formulierten Parameter.

$Q^{min}_{n,t,m}$ die Menge der Mindestwerte des Nährstoffes n, welche von den an der Mahlzeit (t, m) teilnehmenden Nutzern gefordert werden; $Q^{min}_{n,t,m} = \{q^{min}_{p,n,t,m} \mid \exists p \in P : q^{min}_{p,n,t,m} \cdot u_{p,t,m} > 0\}$

$Q^{max}_{n,t,m}$ die Menge der Höchstwerte des Nährstoffes n, welche von den an der Mahlzeit (t, m) teilnehmenden Nutzern gefordert werden; $Q^{max}_{n,t,m} = \{q^{max}_{p,n,t,m} \mid \exists p \in P : q^{max}_{p,n,t,m} \cdot u_{p,t,m} > 0\}$

$V_{t,m}$ die Menge der maximalen Zubereitungszeiten, welche von den an der Mahlzeit (t, m) teilnehmenden Nutzern gefordert werden; $V_{t,m} = \{v_{p,t,m} \mid \exists p \in P : v_{p,t,m} \cdot u_{p,t,m} > 0\}$

$S_{t,m}$ die Menge der Kostengrenzen, welche von den an der Mahlzeit (t, m) teilnehmenden Nutzern gefordert werden; $S_{t,m} = \{s_{p,t,m} \mid \exists p \in P : s_{p,t,m} \cdot u_{p,t,m} > 0\}$

A die Menge aller Lebensmittel, welche als Zutaten in einem Menüplan vorkommen

Parameter

$k_{p,e}$ gibt an, ob ein Nutzer p eine Einschränkung e hat; $k_{p,e} \in \{0$ (Nutzer p hat keine Einschränkung $e)$, 1 (Nutzer p hat Einschränkung $e)\}$

$o_{p,l}$ die von einem Nutzer p vorgenommene Bewertung der Schmackhaftigkeit eines Lebensmittels l; $o_{p,l} \in B$

$q^{min}_{p,n,t,m}$ die von einem Nutzer p in Gramm spezifizierte Menge des Nährstoffes n, welche mindestens in der Mahlzeit (t, m) enthalten sein soll; $q^{min}_{p,n,t,m} \geq 0$

$q^{max}_{p,n,t,m}$ die von einem Nutzer p in Gramm spezifizierte Menge des Nährstoffes n, welche maximal in der Mahlzeit (t, m) enthalten sein soll; $q^{min}_{p,n,t,m} \leq q^{max}_{p,n,t,m}$

$r^{min}_{p,n,t}$ die von einem Nutzer p in Gramm spezifizierte Menge des Nährstoffes n, welche mindestens in den Mahlzeiten am Tag t enthalten sein soll; $r^{min}_{p,n,t} \geq 0$

$r^{max}_{p,n,t}$ die von einem Nutzer p in Gramm spezifizierte Menge des Nährstoffes n, welche maximal in den Mahlzeiten am Tag t enthalten sein soll; $r^{min}_{p,n,t} \leq r^{max}_{p,n,t}$

$s_{p,t,m}$ die von einem Nutzer p in Euro spezifizierte Kostenobergrenze, welche bei der Zubereitung einer Portion der Mahlzeit (t, m) nicht überschritten werden darf; $s_{p,t,m} \geq 0$

$u_{p,t,m}$ gibt an, ob ein Nutzer p an der Mahlzeit (t, m) teilnimmt; $u_{p,t,m} \in \{0$ (Nutzer p nimmt nicht teil), 1 (Nutzer p nimmt teil)$\}$

$v_{p,t,m}$ die von einem Nutzer p in Minuten spezifizierte Zubereitungszeit, welche bei der Zubereitung der Mahlzeit (t, m) nicht überschritten werden soll; $v_{p,t,m} \geq 0$

$w_{p,z,t,m}$ gibt an, ob ein Nutzer p die Zubereitung der Mahlzeit (t, m) gemäß der Zubereitungsform z wünscht; $w_{p,z,t,m} \in \{$ 0 (Nutzer p wünscht keine Zubereitung gemäß Form z), 1 (Nutzer p wünscht eine Zubereitung gemäß Form z)$\}$

$Z^1_{g,t,m}$ Anzahl der von den teilnehmenden Nutzern als schmackhaft bewerteten Zutaten des Gerichts g, bezogen auf die Mahlzeit (t, m)

$Z^2_{g,t,m}$ Anzahl der von den teilnehmenden Nutzern als nicht schmackhaft bewerteten Zutaten des Gerichte g, bezogen auf die Mahlzeit (t, m)

$score^*_b$ der Summand einer Bewertungsmöglichkeit b, $*$ ist entweder *reward* oder *penalty*

$sv_{t,m}$ die Anzahl der benötigten Portionen der Mahlzeit (t, m)

req_l die Menge des zur Zubereitung aller zugeordneten Gerichte benötigten Lebensmittels l in Gramm

av_l die Menge des vorhandenen Lebensmittels l in Gramm

$waste_l$ die Menge des Lebensmittels l, welche unter Berücksichtigung der benötigten und vorhandenen Lebensmittel nicht verarbeitet wird; Angabe in Gramm

Entscheidungsvariablen

$x_{g,t,m}$ gibt an, ob ein Gericht g der Mahlzeit (t, m) zugeordnet wird; $x_{g,t,m} \in \{$ 0 (Gericht g wird nicht zugeordnet), 1 (Gericht g wird zugeordnet)$\}$

Die Anzahl der Entscheidungsvariablen berechnet sich wie folgt:

$$|G| \cdot \sum_{t \in T} |M_t|$$

Beispiel: Für eine Kalenderwoche ergeben sich $|T| = 7$ zu planende Tage. Es sollen für alle Wochentage jeweils drei Mahlzeiten eingeplant werden. Ausnahmen stellen der Samstag und der Sonntag dar, an welchen jeweils eine Mahlzeit auswärts eingenommen wird.

$t \in T$	1	2	3	4	5	6	7
$\|M_t\|$	3	3	3	3	3	2	2

Der Menüplan umfasst in dieser Form 19 Slots und $|G| \cdot 19$ Entscheidungsvariablen, welche binär belegt werden müssen.

Das Menüplanungsproblem wird nachfolgend durch Zielfunktionen, harte Nebenbedingungen und weiche Nebenbedingungen formalisiert. Eine Erläuterung der jeweiligen Komponenten ist im Anschluss gegeben.

Maximiere

$$f_1 = \sum_{t \in T} \sum_{m \in M_t} \sum_{g \in G} Z^1_{g,t,m} \cdot x_{g,t,m} \tag{2.1}$$

$$f_2 = |A| \tag{2.2}$$

Minimiere

$$f_3 = \sum_{t \in T} \sum_{m \in M_t} \sum_{g \in G} Z^2_{g,t,m} \cdot x_{g,t,m} \tag{2.3}$$

$$f_4 = \sum_{l \in A} waste_l \tag{2.4}$$

mit

$$Z^1_{g,t,m} = \sum_{p \in P} u_{p,t,m} \cdot \sum_{y \in A_g} score^{reward}_{o_{p,y}} \tag{2.5}$$

$$score^{reward}_b = \begin{cases} 1 & \text{für } b = 4 \\ 2 & \text{für } b = 5 \\ 0 & \text{sonst} \end{cases} \tag{2.6}$$

$$Z^2_{g,t,m} = \sum_{p \in P} u_{p,t,m} \cdot \sum_{y \in A_g} score^{penalty}_{o_{p,y}} \tag{2.7}$$

$$score^{penalty}_b = \begin{cases} 2 & \text{für } b = 1 \\ 1 & \text{für } b = 2 \\ 0 & \text{sonst} \end{cases} \tag{2.8}$$

$$A = \bigcup_{t \in T, m \in M} \{A_g | g \in G,\ x_{g,t,m} = 1\} \tag{2.9}$$

$$A_g = \{l \in L | a_{g,l} > 0\} \tag{2.10}$$

$$waste_l = \begin{cases} av_l - req_l & \text{für } av_l \geq req_l \text{ und } db_l = 0 \\ \lceil \frac{req_l - av_l}{ps_l} \rceil \cdot ps_l - req_l + av_l & \text{für } av_l < req_l \text{ und } db_l = 0 \\ 0 & \text{sonst} \end{cases} \tag{2.11}$$

$$req_l = \sum_{g \in G} \sum_{t \in T} \sum_{m \in M_t} x_{g,t,m} \cdot a_{g,l} \cdot sv_{t,m} \tag{2.12}$$

$$sv_{t,m} = \sum_{p \in P} u_{p,t,m} \tag{2.13}$$

unter den harten Nebenbedingungen

$$\sum_{g \in G} x_{g,t,m} = 1 \quad , \forall t \in T,\ m \in M_t \tag{2.14}$$

$$x_{g,t,m} \cdot u_{p,t,m} \cdot k_{p,e} \leq f_{g,e} \quad , \forall g \in G,\ t \in T,\ m \in M_t,\ p \in P,\ e \in E \tag{2.15}$$

$$x_{g,t,m} \cdot max(Q^{min}_{n,t,m}) \leq x_{g,t,m} \cdot d_{g,n} \leq min(Q^{max}_{n,t,m}), \tag{2.16}$$
$$\forall g \in G,\ t \in T,\ m \in M_t,\ n \in N$$

$$r^{min}_{p,n,t} \leq \sum_{m \in M_t} x_{g,t,m} \cdot u_{p,t,m} \cdot d_{g,n} \leq r^{max}_{p,n,t}, \tag{2.17}$$
$$\forall\ p \in P,\ t \in T : \sum_{m \in M_t} u_{p,t,m} > 0,\ g \in G,\ n \in N$$

$$x_{g,t,m} \ \cdot \ u_{p,t,m} \ \cdot \ i_g \leq s_{p,t,m}, \forall g \in G,\ t \in T,\ m \in M_t,\ p \in P \tag{2.18}$$

unter den weichen Nebenbedingungen

$$x_{g,t,m} \ \cdot \ j_g \leq min(V_{t,m}), \forall g \in G,\ t \in T,\ m \in M_t \tag{2.19}$$

$$\sum_{p \in P} \sum_{z \in Z} w_{p,z,t,m} \cdot h_{g,z} \cdot x_{g,t,m} \cdot u_{p,t,m} \geq 1, \forall g \in G,\ t \in T,\ m \in M_t \tag{2.20}$$

Erläuterung der Zielfunktionen

Die Zielfunktionen f_1 und f_3 bewerten einen Menüplan hinsichtlich der erfüllten **Nutzerpräferenzen**. Für jedes zugewiesene Gericht wird die Anzahl der von den teilnehmenden

Nutzern bevorzugten Zutaten (2.5) bzw. nicht bevorzugten Zutaten (2.7) ermittelt. Dabei wird ein überaus "gut" (siehe $b = 5$ in 2.6) oder "schlecht" (siehe $b = 1$ in 2.8) bewertetes Lebensmittel doppelt gezählt. Dementsprechend gilt es, die Anzahl der als schmackhaft bewerteten Lebensmittel zu maximieren (2.1) und die Anzahl der als nicht schmackhaft bewerteten Lebensmittel zu minimieren (2.3).

Die **Vielfalt** eines Menüplans wird abgeleitet aus der Menge der Lebensmittel, welche in allen zugewiesenen Gerichten als Zutaten vorkommen. Bei der Optimierung gilt es, die Mächtigkeit dieser als A definierten Vereinigungsmenge zu maximieren (siehe Zielfunktion f_2). Ein Plan mit einer hohen Vielfalt besteht demnach aus Gerichten, welche stets "neue" Zutaten hervorbringen und nur wenige bereits in anderen Gerichten enthaltene Lebensmittel als Zutaten aufführen.

Zielfunktion f_4 summiert die jeweiligen **Abfallmengen** eines im Plan vorgesehenen Lebensmittels auf. Für ein einzelnes Lebensmittel wird die Abfallmenge gemäß 2.11 unter Einbezug vorhandener Mengen (av_l), benötigter Mengen (req_l) und herkömmlicher Verpackungsgrößen (ps_l) für den gesamten Menüplan berechnet. Grundsätzlich wird Abfall als Rest eines verderblichen Lebensmittels verstanden und demnach bezieht sich die Berechnung nur auf die Lebensmittel, für welche gilt, dass $db_l = 0$.

Erläuterung der Nebenbedingungen

Nebenbedingung 2.14 drückt aus, dass jedem Slot genau ein Gericht zugeordnet werden muss. Dies ist ebenfalls an der in Tabelle 3.1 gezeigten Belegung der Entscheidungsvariablen zu erkennen: Jede Zeile enthält genau einen eins-Eintrag und alle anderen Einträge sind null. Der umgekehrte Fall gilt nicht, da nicht jedes Gericht einem Slot zugeordnet wird. Tabelle 3.1 zeigt dies beispielsweise für das Gericht 5: für alle $t \in T, m \in M_t$ gilt $x_{5,t,m} = 0$.

Tabelle 3.1: Tabellarische Belegung der Entscheidungsvariablen für das beschriebene Beispiel und $|G| = 100$ (Auszug)

t	m	$g \in G$							
		1	2	3	4	5	6	...	100
...						0	...		
5	1	0	0	0	0	0	**1**	0	0
	2	0	**1**	0	0	0	0	0	0
	3	**1**	0	0	0	0	0	0	0
6	1	0	0	0	**1**	0	0	0	0
	2	0	0	**1**	0	0	0	0	0
...						0	...		

Nebenbedingung 2.15 entspricht der folgenden Regel: Wenn ein Gericht g der Mahlzeit m am

Tag t zugewiesen wird ($x_{g,t,m} = 1$) und ein Nutzer p an diesem Slot teilnimmt ($u_{p,t,m} = 1$) und dieser Nutzer eine Einschränkung e hat ($k_{p,e} = 1$), muss das Gericht bei dieser Einschränkung verzehrbar sein ($f_{g,e} = 1$). Nebenbedingung 2.16 fordert, dass die Nährstoffwerte eines zugewiesenen Gerichts innerhalb der von den teilnehmenden Nutzern formulierten Nährwertschranken liegen. Dies soll auch für die von den Nutzern geforderten Tagesgrenzwerte gelten (siehe Nebenbedingung 2.17). Ein Tagesgrenzwert gilt für alle Mahlzeiten, welche von einem Nutzer an einem Tag eingenommen werden. Dies bedeutet, dass sich ein solcher Wert auch nur auf einen Teil der Mahlzeiten beziehen kann. Diese Nebenbedingung muss dann erfüllt werden, wenn ein Nutzer an mindestens einer Mahlzeit des Tages teilnimmt.
Die in 2.18 formulierte Ungleichung drückt aus, dass die von einem Nutzer angegebene Kostengrenze zur Zubereitung einer Portion eines Gerichts eingehalten werden muss (sofern der Nutzer an der entsprechenden Mahlzeit teilnimmt).
Die weiche Nebenbedingung 2.19 fordert, dass die Zubereitungsdauer eines zugewiesenen Gerichts keine der von den teilnehmenden Nutzern geforderten Zeiten überschreitet.
Da sich die bevorzugten Zubereitungsformen der teilnehmenden Nutzer (bezogen auf einen Slot) nicht immer überschneiden, macht es wenig Sinn, ein Gericht nur dann zu akzeptieren, wenn es gemäß aller in der Schnittmenge enthaltenen Formen zubereitet wird. Stattdessen gilt eine Zuweisung als valide, wenn es gemäß einer Form zubereitet wird, welche von einem teilnehmenden Nutzer gewünscht wird. Dies wird in Nebenbedingung 2.20 ausgedrückt.

Klassifikation des Menüplanungsproblems entsprechend Abschnitt 2.1

Das Menüplanungsproblem wird unter Berücksichtigung der Formalisierung als *linear* und binär - und damit *ganzzahlig, diskret* - klassifiziert. Das Modell umfasst vier Zielfunktionen und ist aufgrund der Kenntnis aller Parameterwerte als deterministisch einzuordnen. Die Komplexität der Berechnung kann an dieser Stelle noch nicht abgeschätzt werden. Aufschlussreich ist allerdings die Betrachtung der Anzahl aller Lösungskandidaten. Aus dem binären Wertebereich und der Anzahl der Entscheidungsvariablen ergeben sich ohne Berücksichtigung der Restriktionen $2^{|G| \cdot \sum_{t \in T} |M_t|}$ mögliche Menüpläne. Im Falle von 21 zu planenden Mahlzeiten und 500 zur Auswahl stehenden Gerichten sind dann 2^{10500} Menüpläne vorstellbar. Diese Erkenntnisse wirken sich auf die **Wahl eines Lösungsverfahrens** aus.
Die in Abschnitt 3.1 beschriebenen mathematischen Ansätze von Dantzig und Balintfy lösen ein Problem mit nur einer Zielfunktion und sind somit nicht direkt auf das vorliegende Menüplanungsproblem anwendbar. Erforderlich wäre die Kombination der Zielfunktionen, z.B. unter Verwendung von Zielgewichten. Eine solche Modellierung hat zur Folge, dass eine

Verschlechterung des Wertes einer Zielfunktion durch die Verbesserung des Wertes einer anderen Zielfunktion ausgeglichen wird.
In Anbetracht der Komplexität des Problems ist es offensichtlich, dass Verfahren, welche den gesamten Suchraum absuchen, gänzlich ungeeignet sind. Gleiches gilt aufgrund ihrer voraussichtlich inakzeptablen Laufzeiten für alle exakten Verfahren. Dementsprechend wird der Einsatz eines heuristischen Verfahrens favorisiert.
In 1.2 wurde als funktionale Anforderung festgehalten, dass die Berechnung jederzeit vom Anwender beendet werden kann und der bis dahin beste Menüplan präsentiert wird. Aus diesem Grund scheiden konstruierende Verfahren aus, da nicht garantiert werden kann, dass zum Zeitpunkt des Abbruchs eine Lösung bereit steht. Ein Verbesserungsverfahren wie es in [Sel09] beschrieben wird, verbleibt als geeignetes Lösungsverfahren.
Auch das in [MPS99] gewählte Vorgehen verspricht eine gute Lösung und adressiert das Problem der nachträglichen Verfeinerung eines Menüplans durch einen Nutzer. Die Implementierung einer solchen assistierenden Komponente ist aus Sicht des Nutzers wünschenswert und wurde im Rahmen dieser Arbeit als optional eingeordnet. Das Aufgreifen des regelbasierten Ansatzes ist auch im Hinblick auf die vorhandene wissensbasierte Dienstplattform (gegeben mit dem Terminologieserver) sinnvoll.
Ein nicht gänzlich zu vernachlässigender Aspekt bei der Umsetzung eines Lösungsverfahrens ist die Integration benötigter Programmbibliotheken. Idealerweise sind diese in Java verfasst und erlauben eine objektorientierte Behandlung der Informationsobjekte ohne weitere Transformationsschritte. Die Informationsobjekte bezeichnen die vom Terminologieserver abfragbaren Konzepte zu Allergien, Lebensmitteln, Rezepten, etc. Angestrebt ist eine elegante Umformung dieses Problemwissens in eine vom Lösungsverfahren vorausgesetzte Repräsentation.

3.3 Lösungsansatz

Vor dem Hintergrund der bisherig beschriebenen Lösungsverfahren und der Einordnung wird in diesem Abschnitt ein auf das Menüplanungsproblem anwendbares Vorgehen präsentiert. Ausgewählt und adaptiert wird die in [Sel09] beschriebene Umsetzung eines evolutionären Verfahrens mithilfe des **Nondominated Sorting Genetic Algorithm II (NSGA-II)**. Die verbesserte Variante des in [SD94] vorgestellten Algorithmus gilt als Standard-Verfahren zur Lösung multikriterieller Probleme. Bei einer Populationsgröße von N und M Zielfunktionen erreicht der NSGA-II eine Laufzeitkomplexität von $O(MN^2)$. Dabei wird die Pareto-Optimierung umgesetzt, durch welche die besten Kompromiss-Lösungen gefunden

werden. Diese als *Pareto-Menge* bezeichnete Lösungsmenge enthält *nicht-dominierte* bzw. *Pareto-optimale* Elemente. Wird die Optimierung unter Ausschluss der Pareto-Menge erneut durchgeführt, werden wiederum nicht-dominierte Lösungen gefunden. Dieser Vorgang kann wiederholt werden, bis alle Lösungen Elemente von Pareto-Mengen sind. Die in jeder Iteration gefundenen Pareto-Mengen werden als *Fronten* bezeichnet und entsprechend des Durchlaufs, in dem sie gebildet wurden, nummeriert. Die initial gefundene Pareto-Menge entspricht demnach der 1.Front, die in der darauffolgenden Wiederholung gefundene Pareto-Menge der 2.Front usw.

Mithilfe des NSGA-II werden mehrere Pareto-optimale Lösungen innerhalb eines Durchlaufs gefunden. Dabei wird nicht garantiert, dass die gefundenen Lösungen in der wahren Pareto-Front enthalten sind. Im Vergleich mit anderen Algorithmen zur Mehrzieloptimierung zeigt der NSGA-II für viele Testprobleme eine verbesserte Konvergenz und damit einen geringeren Abstand gefundener Lösungen zu Lösungen der wahren Pareto-Menge. Darüber hinaus erlaubt er die Behandlung von Restriktionen (*Constrained NSGA-II)* und ist Bestandteil vieler Programmbibliotheken.

Mit Bezug auf den in Abschnitt 2.3.3 dargestellten schematischen Ablauf evolutionärer Algorithmen (Abbildung 2.2) wird nachfolgend der Prozess des Constrained NSGA-II erläutert.

1. *Initialisierung*

   Es wird zufällig eine Population P_t mit N Individuen erstellt.

2. *Bewertung*

   Die Individuen der Population P_t werden entsprechend der *Nondominated Sorting* Prozedur[3] sortiert. Die Dominanz-Relation zwischen zwei Lösungen wird dabei wie folgt definiert: Eine Lösung i dominiert eine Lösung j, wenn eine der folgenden Bedingungen gilt:

   a) Lösung i ist zulässig[4] und Lösung j nicht

   b) sowohl Lösung i als auch Lösung j sind nicht zulässig, aber Lösung i hat insgesamt eine geringere Constraint-Verletzung als Lösung j

   c) sowohl Lösung i als auch Lösung j sind zulässig und Lösung i dominiert Lösung j hinsichtlich der Zielfunktionswerte

   Die Individuen werden gemäß dieser Definition in Pareto-Fronten unterteilt. Durch diese Strategie wird sichergestellt, dass zulässige Lösungen besser bewertet werden als unzulässige. Gleichermaßen dominiert eine unzulässige Lösung eine andere, wenn sie -

[3] siehe [DPAM02, S.184]
[4] zulässig bedeutet, dass alle Restriktionen erfüllt werden

gemessen an der Summe aller Constraint-Verletzungen - dem zulässigen Bereich näher ist. Jedem Individuum wird als Fitness ein Rang zugeordnet. Dieser Rang ist gleich der Front, zu welcher das Individuum gehört; das heißt: Individuen der ersten Front erhalten den Rang 1, Individuen der zweiten Front Rang 2 usw.

3. Paarungsselektion

Aus der Population werden zufällig zwei Individuen einer binären Turnier-Selektion unterzogen. In der initialen Generation ($t = 0$) gewinnt dabei stets das Individuum mit dem besseren Rang. In allen darauf folgenden Generationen steuert der *Crowded-Comparison Operator* ($\prec_n$) die Selektion. Neben dem Rang eines Individuums i (i_{rank}) wird zusätzlich die *Crowding-Distance* ($i_{distance}$) als zu vergleichende Größe herangezogen. Die Crowding Distance gibt an, wie nah ein Individuum an seinen Nachbarn liegt.[5] Ziel ist die Bildung einer möglichst diversen Nachfolgepopulation, sodass Individuen, welche in weniger dichten Nachbarschaften liegen, bevorzugt werden. Diese Forderung wird dann erfüllt, wenn Individuen mit hohen Distanzen als Eltern selektiert werden. Zwei an der Turnier-Selektion teilnehmende Individuen i und j werden mittels des Crowed-Comparison Operators wie folgt verglichen:

$$i \prec_n j \text{ if } (i_{rank} < j_{rank}) \text{ or } ((i_{rank} = j_{rank}) \text{ and } (i_{distance} > j_{distance})) \tag{3.1}$$

Demnach gewinnt das Individuum der besseren (niedrigeren) Front oder - gehören beide Individuen zur selben Front - das Individuum mit der höheren Crowding-Distanz. Aus der Beschreibung des Algorithmus geht nicht hervor, ob ein Individuum nur ein einziges Mal an der Selektion teilnimmt.

4., 5. Rekombination und Mutation

Durch Anwendung der genetischen Operatoren wird eine Kind-Population Q_t der Größe N erzeugt.

6. Bewertung

Zunächst wird eine kombinierte Population $R_t = P_t \cup Q_t$ erstellt und die enthaltenen $2N$ Individuen werden anhand der Dominanz-Definition sortiert (siehe Abbildung 3.1). Da diese Population die Eltern und Nachkommen eint, wird garantiert, dass die besten Individuen unverändert übernommen werden (*Elite-Strategie).*

[5] die Berechnung der Crowding Distance wird ausführlich in [DPAM02, S.185] beschrieben

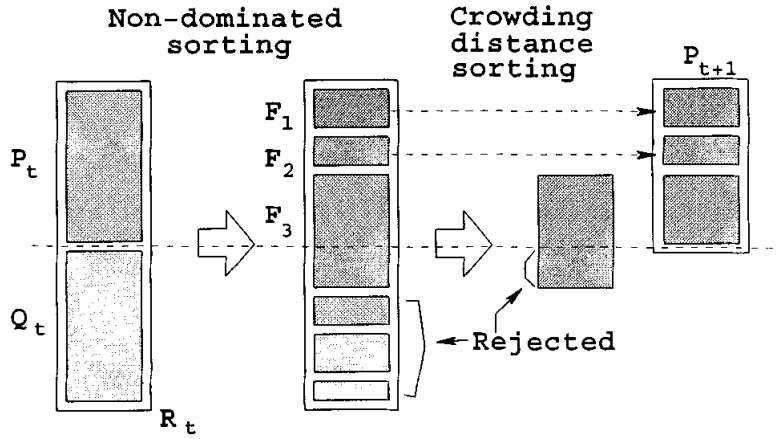


Abbildung 3.1: Bewertung und Umweltselektion des NSGA-II, entnommen aus [DPAM02, S.186]

7. *Umweltselektion*

   Aus R_t müssen nun N Individuen zur Bildung der nächsten Population gewählt werden. Individuen der vordersten Fronten sollen bei diesem Auswahlprozess bevorzugt werden, sodass - beginnend mit der besten Front F_1 - nacheinander die Mitglieder der Fronten in die nächste Population P_{t+1} übernommen werden. Dies erfolgt so lange, bis die Individuen einer Front nicht mehr vollständig untergebracht werden können, da andernfalls die Populationsgröße überschritten werden würde. Abbildung 3.1 veranschaulicht diesen Umstand: Die Individuen der Fronten F_1 und F_2 werden vollständig selektiert, F_3 kann nicht vollständig übernommen werden, da ansonsten zu viele Individuen in P_{t+1} enthalten wären. In diesem Fall werden die Lösungen in F_3 absteigend nach ihrer *Crowding Distance* sortiert und es werden die besten Individuen zum Auffüllen der Folge-Population gewählt.

   Die Crowding Distance wird grundsätzlich für alle in die Population P_{t+1} übernommenen Individuen berechnet, da sie Einfluss auf die nächste Paarungsselektion nimmt.

8. *Terminierungsbedingung*

   Wird eine festgelegte Abbruchbedingung nicht erfüllt, beginnt die nächste Iteration mit der Paarungsselektion (Schritt 3). Andernfalls terminiert der Algorithmus und liefert die sortierte Nachfolgepopulation P_{t+1} als Ergebnis.

Mit dem Constrained NSGA-II wird ein vielversprechendes Lösungsverfahren gewählt. Um dieses auf das vorliegende Menüplanungsproblem anwenden zu können, müssen die folgenden EA-Komponenten spezifiziert werden: **die Kodierung eines Individuums, die Fitnessfunktionen** und **die genetischen Operatoren**. Als weiterer Punkt wird die Behandlung von **Constraints** beschrieben. Der NSGA-II setzt standardmäßig die beschriebene binäre

Turnier-Selektion um, sodass die Festlegung der Selektionsstrategie entfällt.

a.) Kodierung eines Individuums

Ein Individuum entspricht einem mit Gerichten belegten Menüplan. Diese Menge an Zuordnungen kann vielfältig repräsentiert werden. Für genetische Algorithmen wird häufig die Kodierung der Bitvektoren gewählt. Eine alternative Repräsentation wird in [Sel09] vorgestellt. Dabei wird das zu lösende Menüplanungsproblem in mehrere Sub-Probleme (n-Tages-Ebene, Tages-Ebene und Mahlzeiten-Ebene) unterteilt und durch Chromosomen aus ganzzahligen Werten kodiert. Das Ergebnis einer Zuordnung entspricht einem Paar, welches aus einer Gramm-Angabe und einer Referenz auf einen Rezept-Schlüssel besteht.
Eine solche Kodierung ist im Vergleich zu einer Repräsentation mit Bitvektoren besser verständlich und in Anbetracht der objektorientierten Ziel-Umgebung weniger aufwendig umzusetzen. Zudem spiegelt sie die zugrundeliegende Zuordnungsvorschrift unmittelbar wider und wird daher der klassischen Kodierung vorgezogen. Die im vorherigen Abschnitt präsentierte Formalisierung kann aufgrund des binären Wertebereichs der Entscheidungsvariablen nicht direkt auf eine Individuen-Kodierung übertragen werden. Wenn möglich, werden die definierten Mengen und Parameter zur Beschreibung eines Chromosoms herangezogen.
Ein Menüplanungsproblem wird im Kern durch die Menge $Slots$ beschrieben. Ein Element (t, m) dieser Menge wird als Parameter $p_{t,m}$ verstanden, dessen Wertebereich der Menge der Gerichte G entspricht. Eine Lösung ist demnach eine Belegung aller Parameter mit einer in G enthaltenen ganzen Zahl. Es bietet sich an, alle Parameter in einem Array zu bündeln. Um den Zugriff zu erleichtern, wird als Index das Paar (t, m) mit $(t, m) \in Slots$ angenommen.

Beispiel: Ein Menüplan umfasst insgesamt acht zu planende Mahlzeiten. Diese verteilen sich auf drei Tage, wobei an den ersten zwei Tagen jeweils drei und am letzten Tag zwei Mahlzeiten vorgesehen sind. Dies entspricht der folgenden Menge $Slots = \{(0,0), (0,1), (0,2), (1,0), (1,1), (1,2), (2,0), (2,1)\}$. Daraus ergeben sich die Parameter $p_{0,0}, p_{0,1}, \ldots, p_{2,1}$, welchen beispielhaft die Gerichte $50, 3, 7, 29, 4, 24, 13$ und 74 zugeordnet werden. Die Belegung wird in Abbildung 3.2 in Form eines Arrays dargestellt. Die Klammerausdrücke beinhalten den Index zur Adressierung eines Elements.

[(0,0)]	[(0,1)]	[(0,2)]	[(1,0)]	[(1,1)]	[(1,2)]	[(2,0)]	[(2,1)]
50	3	7	29	4	24	13	74

Abbildung 3.2: Beispiel eines Individuums zur Repräsentation eines Menüplans

Die gewählte direkte Repräsentation eines Individuums als Parameter-Array macht eine Dekodierung überflüssig.

b.) Fitnessfunktionen

Die vier Fitnessfunktionen wurden im Rahmen der Formalisierung definiert. Während der Pareto-Optimierung werden die Werte dieser Funktionen stets individuell betrachtet und nicht aggregiert. Aufgrund dessen muss keine Normalisierung erfolgen.

c.) Constraints

Die Population, welche durch den genetischen Algorithmus fortlaufend verändert wird, kann in jeder Iteration unzulässige Lösungen beherbergen.
Für den überwiegenden Teil der Restriktionen kann eine Verletzung ausgeschlossen werden, indem die Menge der zuordenbaren Gerichte eingeschränkt wird. Dies ist der Fall für die durch die Ungleichungen 2.15, 2.16, 2.18, 2.19 und 2.20 formulierten Nebenbedingungen.[6] Diese nehmen jeweils Bezug auf gesundheitliche Einschränkungen eines Nutzer, für eine Mahlzeit spezifizierte Nährstoffgrenzen, maximale Zubereitungskosten und -zeiten sowie geforderte Zubereitungsformen. Leidet ein Nutzer beispielsweise unter einer Erdnuss-Allergie, werden für alle Mahlzeiten, an welchen der Nutzer teilnimmt, diejenigen Gerichte aus G exkludiert, welche bei dieser Einschränkung nicht verzehrt werden dürfen. Das Wissen über solche Zusammenhänge wird durch den Terminologiedienst bereitgestellt, sodass es dessen Aufgabe ist, den Wertebereich $G_{p_{t,m}}$ mit $G_{p_{t,m}} \subset G$ eines Parameters $p_{t,m}$ festzulegen.
Diese Strategie kann nicht verfolgt werden, wenn die Zulässigkeit eines Individuums anhand einer Parameter-Kombination überprüft wird. Dies betrifft die Nebenbedingung 2.17, welche die Erfüllung der tagesbezogenen Nährstoffgrenzen aller teilnehmenden Nutzer fordert. Der NSGA-II quantifiziert solche Restriktionen in Form von *Constraint-Verletzungen*. Dieser numerische Wert gibt für eine Lösung das Ausmaß der Unzulässigkeit an. Für ein zulässiges Individuum ergibt sich demnach eine Constraint-Verletzung von null.

Beispiel: Für einen Tag t werden von zwei Nutzern i, j jeweils zu erreichende Mindestmengen des Nährstoffes n spezifiziert. Diese werden mit $r_{i,n,t}$ und $r_{j,n,t}$ bezeichnet. Durch den Algorithmus wird ein Individuum bewertet, welches alle Mahlzeiten des Tages t so mit Gerichten belegt, dass die Nutzer den Nährstoff jeweils in den Mengen x_i bzw. x_j zu sich nehmen. Gilt nun, dass $r_{i,n,t} \leq x_i$ und $r_{j,n,t} \leq x_j$, dann ist das Individuum zulässig und die Constraint-Verletzung ist gleich null. Für jede nicht erfüllte Restriktion wird die Differenz

[6] Der NSGA-II berücksichtigt nicht die ursprünglich vorgenommene Kategorisierung in *weiche* und *harte* Nebenbedingungen, sodass diese fortan nicht mehr unterschieden werden.

zwischen dem zu erreichenden Wert und dem tatsächlichen Wert gebildet und aufsummiert. Werden beispielsweise die zwei Restriktionen nicht erfüllt und es gilt, dass $r_{i,n,t} > x_i$ und $r_{j,n,t} > x_j$, dann berechnet sich die Constraint-Verletzung wie folgt: $r_{i,n,t} - x_i + r_{j,n,t} - x_j$.

Auf diese Art und Weise werden alle tagesbezogenen Nährstoffgrenzen durch den Constrained NSGA-II verarbeitet. Ein Individuum, welches eine Nebenbedingung nicht erfüllt, wird durch den Algorithmus mit einer schlechteren Bewertung (einem höheren Rang) bestraft. Ein solches Individuum wird an zwei Stellen des Algorithmus benachteiligt. Zum einen unterliegt es bei der binären Turnierselektion einem Individuum, welches alle Constraints erfüllt oder insgesamt eine geringere Constraint-Verletzung aufweist. Damit verringern sich die Chancen des Individuums, am Reproduktionsprozess teilzunehmnen. Zum anderen werden unzulässige Nachkommen in der kombinierten Population stets von Eltern oder anderen Kind-Individuen dominiert, wenn diese die Constraints in einem geringeren Maße oder überhaupt nicht missachten. Dies führt dazu, dass solchen Nachkommen die Übernahme in die nächste Population verwehrt bleibt.

d.) Genetische Operatoren

Da keine Kodierung eines Individuums als Bitvektor vorgenommen wird, verbleiben diejenigen genetischen Operatoren, welche auf Parameterebene angewandt werden können. Von diesen werden das *1-Punkt-Crossover* und die *Integer-Mutation* gewählt.
Beim 1-Punkt-Crossover werden zwei Individuen an einem zufällig bestimmten Index in zwei Teile getrennt. Zwei Nachfolgeindividuen entstehen, indem die Startstücke mit dem Endstück des jeweils anderen Chromosoms kombiniert werden. Abbildung 3.3 zeigt diesen Vorgang für zwei Beispiel-Individuen.

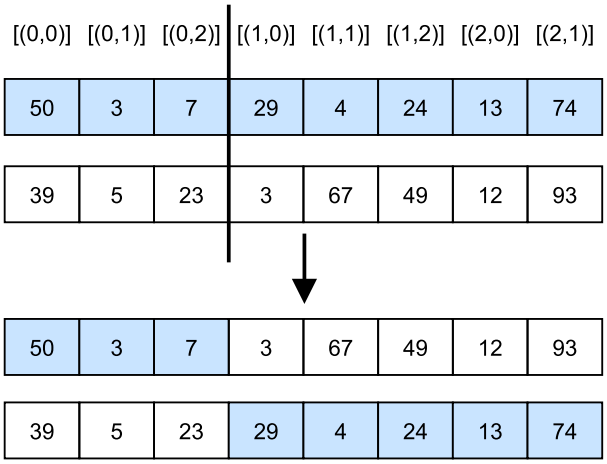


Abbildung 3.3: 1-Punkt-Crossover zwischen zwei Beispiel-Individuen, Crossover nach (0,2)

Diese Form der Rekombination resultiert in neuen Zuordnungen für die durch die Endstücke repräsentierten Slots. Dies hat Auswirkungen auf die Zulässigkeit der erzeugten Nachkommen und kann dazu führen, dass zuvor erfüllte tagesbezogene Constraints durch die Rekombination verletzt werden. Umgekehrt kann eine solche Aufteilung auch dazu führen, dass Constraint-Verletzungen der Eltern reduziert oder gänzlich aufgehoben werden.
Die gewählte Integer-Mutation zählt zu den Ein-Elter-Operatoren und erzeugt aus einem Individuum durch die Veränderung der Parameterwerte ein neues Individuum. Jede Belegung eines Parameters wird dabei mit einer als *Mutationsrate* bezeichneten Wahrscheinlichkeit geändert. Eine Änderung entspricht einer Zuordnung eines neuen Gerichts. Abbildung 3.4 veranschaulicht die Mutation an einem Beispiel. Den Parametern (0,2) und (2,1) werden unter Berücksichtigung der jeweiligen Wertebereiche die Gerichte 7 bzw. 8 zugeordnet.

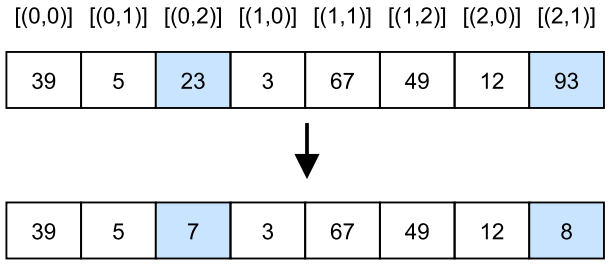


Abbildung 3.4: Integer-Mutation eines Beispiel-Individuums, Mutation bei (0,2) und (2,1)

Auch diese genetische Operation kann die Zulässigkeit einer Lösung beeinflussen.

Der beschriebene Lösungsansatz zeichnet sich im Wesentlichen durch die folgenden Elemente aus:

- Mehrzieloptimierung gemäß NSGA-II
- direkte Kodierung eines Menüplans als Parameter-Array
- separate Behandlung von Zielfunktionen und Constraints
- 1-Punkt-Crossover und Integer-Mutation als genetische Operatoren

3.4 Aufbau der terminologischen Wissensbasis

Dieser Abschnitt dokumentiert die Arbeitsschritte, welche zur Beantwortung des zweiten Teils der Forschungsfrage beitragen. In diesem soll untersucht werden, wie terminologisch strukturiertes Domänenwissen im Kontext der optimierten Menüplanung genutzt werden kann.

Terminologisch strukturiert bedeutet hier, dass die Daten aus *kontrollierten Vokabularien* (auch *Ordnungs-* oder *Begriffssysteme* genannt) stammen. Diese repräsentieren das von Domänenexperten formulierte Wissen in standardisierter Form, wodurch eine eindeutige Kommunikation zwischen Akteuren und Systemen ermöglicht wird. Diese Begriffssysteme unterscheiden sich hinsichtlich ihrer Komplexität und variieren von einfachen Codelisten über hierarchisch strukturierte Taxonomien bis hin zu Wissensnetzen. Ein System, welches Axiome, Integritätsbedingungen und Regeln zur Ableitung neuen Wissens umfasst, wird als *Ontologie* bezeichnet. [Gru93] Zur Beschreibung der Strukturen eines Begriffssystems eigen sich formale Sprachen wie beispielsweise das *Resource Description Framework Schema (RDFS)*[7], die objektorientierte und Frame-basierte Sprache F-Logic [KLW95] oder die *Web Ontology Language (OWL)*[8].

Um diese Wissensstrukturen effizient zu verwalten und darüber hinaus nützliche Funktionalität anzubieten, werden spezielle Dienste benötigt. Vor diesem Hintergrund entwickelte das Fraunhofer FOKUS den Terminologiedienst CTS2-LE, welcher nachfolgend erläutert wird.

CTS2-LE: ein wissensbasierter Terminologiedienst

CTS2-LE (LE für Linked Data Edition) folgt allgemeingültigen und international etablierten Standards zur Repräsentation und Strukturierung von Terminologien. Zum einen sind dies die Semantic Web Standards *Resource Description Framework (RDF)*[9], *RDFS* sowie *SPARQL*[10]. Zum anderen wird das plattformunabhängige Modell des HL7/OMG Standards *Common Terminology Services Release 2* (CTS2)[11] aufgegriffen und als sogenanntes *Informationsmodell* verankert. Dieses entspricht der Konzeptionalisierung (im Bereich der Beschreibungslogiken als TBox bezeichnet), welche neben *Klassen* und *Relationen*[12] auch *Restriktionen* umfasst. Zur Formulierung des Informationsmodells wird ein am Fraunhofer

[7] https://www.w3.org/TR/rdf-schema/, RDF Schema 1.1, letzter Zugriff am 29.09.16
[8] https://www.w3.org/TR/owl-primer/, OWL 2 Web Ontology Language Primer (Second Edition), letzter Zugriff am 03.10.16
[9] https://www.w3.org/TR/rdf11-concepts/, RDF 1.1 Concepts and Abstract Syntax, letzter Zugriff am 29.09.16
[10] https://www.w3.org/TR/rdf-sparql-query/, SPARQL Query Language for RDF, letzter Zugriff am 27.12.16
[11] http://www.omg.org/spec/CTS2/, letzter Zugriff am 27.12.16
[12] Die Begriffe *Klasse* und *Konzept* sowie *Relation* und *Property* werden nachfolgend synonym gebraucht.

FOKUS entwickeltes Signaturen-Vokabular verwendet. Das eigentliche Faktenwissen (auch als *assertionale Komponente* oder ABox bezeichnet) wird in Form von RDF-Graphen abgelegt.
CTS2-LE bietet neben der reinen Verwaltung und Abfrage von beliebig strukturierten Terminologien die folgenden Funktionalitäten:

- Validierung des integrierten Wissens anhand des Informationsmodells
- Verwaltung von anwendungsspezifischen Begriffsmengen (*Value Sets*) und Semantikerhaltenden Abbildungen von Begriffen unterschiedlicher Terminologien (*Mappings*)
- Suche über eine Volltextindizierungs-Komponente
- komfortables Erforschen der Informationsobjekte innerhalb einer Web-Applikation (auch als *Navigator* bezeichnet)
- Zuordnung von Begriffen einer oder mehrerer Terminologien zu einem gegebenen Freitext (*Konzept-Matching*) gemäß [BBL07]

Weiterführende Informationen sind [BK14] oder dem CTS2-LE Wiki[13] zu entnehmen.
Im Rahmen der Studienarbeit mit dem Titel *Aufbau eines RDFS-Wissensnetzes für ein Speiseempfehlungssystem im Krankenhaus*[14] wurde der Terminologiedienst genutzt, um eine Datenbasis zu schaffen, welche Zusammenhänge zwischen gesundheitlichen Einschränkungen, ethischen und organisatorischen Faktoren sowie Lebensmitteln und Rezepten beschreibt. Zu diesem Zweck wurde ein zusätzliches Informationsmodell zur Strukturierung der Domäne entwickelt. Aus diesem kann beispielsweise abgelesen werden, dass Instanzen der Klasse *DietRestriction* über die Relation *excludesFood* mit Instanzen der Klasse *Food* verbunden werden können. Eine Instanz entspricht einem konkreten Objekt einer Klasse, z.B. einer Laktoseintoleranz, einer Glutenunverträglichkeit oder - auf das Konzept *Food* bezogen - einer Kartoffel oder Zucchini. Dieses Faktenwissen stammt überwiegend aus standardisierten Vokabularien, welche auf vielfältige Art und Weise im elektronischen Gesundheitswesen eingesetzt werden. Verwendet wurden u.a. Einträge des Verschlagwortungskatalogs *Medical Subject Headings (MeSH)*[15], der Systematisierten Nomenklatur der Medizin (engl. *Systematized Nomenclature of Human and Veterinary Medicince - Clinical Terms (SNOMED-CT)*)[16], des Bundeslebensmittelschlüssels (BLS)[17] und des Einheitensystems *Unified Code for Units of Measure (UCUM)*[18]. Der Einsatz des Wissensnetzes innerhalb eines Speiseempfehlungssys-

[13] erreichbar unter https://publicwiki-01.fraunhofer.de/CTS2-LE, letzter Zugriff am 27.12.16
[14] Die Studienarbeit ist dem DVD-Anhang zu entnehmen.
[15] https://www.nlm.nih.gov/mesh/, letzter Zugriff am 29.09.16
[16] http://www.ihtsdo.org/snomed-ct, letzter Zugriff am 29.09.16
[17] https://www.mri.bund.de/de/service/datenbanken/bundeslebensmittelschluessel/, letzter Zugriff am 29.09.16
[18] http://unitsofmeasure.org/trac, letzter Zugriff am 29.09.16

tems zeigte, dass die in Form von Kompetenzfragen formulierten Anforderungen erfüllt und die benötigte Ausdrucksstärke erreicht wurde. Die Vollständigkeit und Richtigkeit des (insbesondere abgeleiteten) Wissens konnten nicht sichergestellt werden, da keine fachliche Prüfung durch einen Domänenexperten (z.B. einen Ernährungswissenschaftler oder Diätassistenten) erfolgte. Weitere Mängel betreffen

1. die Herkunft, den Umfang und die Qualität verwendeter Daten,
2. die semantische Vernetzung unterschiedlicher Quellen und
3. die Auswirkungen manuell festgelegter Relationen auf die Vernetzung.

Ersteres bezieht sich auf die Datenquellen, welche zur Integration von Rezepten und Lebensmittelpreisen genutzt wurden. Da für diese Konzeptmengen keine standardisierten Vokabularien o.ä. existieren, erfolgte eine händische Sammlung und Aufbereitung. Infolgedessen ist die Anzahl eben jener Informationsobjekte eher gering.[19] Der zweite Punkt bezieht sich insbesondere auf das *Matching* von Zutatenbezeichnern auf eindeutige Einträge des BLS. Ein automatisches Matching konnte u.a. aufgrund von Textverarbeitungsschwierigkeiten (z.B. Synonyme, Rechtschreibfehler, Numeri) und der hohen Anzahl infrage kommender BLS Einträge nicht umgesetzt werden. Stattdessen wurde für die in Rezepten vorkommenden Zutaten das Matching auf ein Konzept des BLS manuell vorgenommen. Dies führt dazu, dass sich weitere manuell festgelegte Relationen nicht auf das gesamte BLS-Vokabular, sondern auf die Menge der zugeordneten BLS-Konzepte beziehen (siehe Punkt 3). [Kre16]
Die Ergebnisse der Studienarbeit bilden die Grundlage der für die Menüplanung benötigten Wissensbasis. Um möglichst viele der in Abschnitt 1.4 formulierten Kriterien zu erfüllen und einen Teil der zuvor genannten Probleme zu adressieren, wird die Auswahl der Datenquellen reflektiert und bestehende Programmeinheiten zur Integration und Vernetzung werden angepasst. Der Terminologiedienst wird ggf. um zusätzlich benötigte Funktionalität erweitert.

3.4.1 Wiederverwendung existierender Terminologien

Der Aufbau des Wissensnetzes orientiert sich stark an dem von Noy und McGuinness vorgestellten Leitfaden zur Entwicklung von Ontologien. [NM01] Der zweite Schritt des Leitfadens sieht die Wiederverwendung existierender Begriffssysteme vor, sodass im Rahmen der Studienarbeit zahlreiche Begriffsmengen aus standardisierten Terminologien entnommen oder nach Bedarf ganze Vokabularien zur Abbildung der Domäne genutzt wurden.

[19] Im Rahmen der Studienarbeit wurden 51 Rezepte und Preisangaben für ca. 200 Lebensmittel integriert.

Im Falle der Menüplanung erschließen sich die Konzepte und Relationen aus den Anforderungen und sind nahezu deckungsgleich mit den in der Vorarbeit definierten Konstrukten. Daraus übernommen werden die um das relationale Wissen angereicherten Instanzen für die folgenden Klassen und Properties:

A **Lebensmittel** mit Nährwertdaten

B **Diätische Einschränkungen** wie **Allergien, Unverträglichkeiten** und **Ernährungsweisen** mit jeweiligen auszuschließenden Lebensmitteln

C **Einheiten** zur Spezifizierung von Mengenangaben

Das Faktenwissen zu A wird vollständig durch den BLS bereitgestellt, während für B einzelne Konzepte der standardisierten Begriffssysteme MeSH und SNOMED CT wiederverwendet werden. Ebenso werden für C nur Teile des Einheitenvokabulars UCUM benötigt. Da UCUM keine haushaltsüblichen Einheiten umfasst, werden diese in einem zusätzlichen Begriffssystem gebündelt.
Das Faktenwissen zu

D Rezepten, Zubereitungszeiten und -formen sowie zu

E Verpackungsgrößen, Preisangaben und Hinweisen zur Haltbarkeit bzw. zur Aufbewahrungsdauer von Lebensmitteln

kann nicht aus der Vorarbeit übernommen werden, da dieses entweder nicht umfänglich genug, unvollständig oder überhaupt nicht spezifiziert wurde.
Nachfolgend werden mögliche Datenquellen zur Integration von D und E erörtert. Anschließend wird das Informationsmodell der Vorarbeit aufgegriffen und angepasst.

Auswahl einer Rezeptdatenbank

Im Rahmen dieser Arbeit wurden zur Beschaffung von D zwei Datenquellen untersucht: 1.) die *Rezeptsammlung der Unix-AG* und 2.) die Abfrageschnittstelle der *Yummly*-Plattform.

Rezeptsammlung der Unix-AG

Diese Sammlung umfasst über 19000 Rezepte[20], welche im **Mealmaster**-Format bereitgestellt werden. Anhang A.3 zeigt die zugrundeliegende Grammatik in der Erweiterten Backus-Naur-Form (EBNF). Eine nähere Untersuchung der Sammlung offenbarte die folgenden Mängel:

[20] Die Sammlung ist unter der folgenden URL unter dem Menüpunkt Kochbuch abrufbar: http://www.data norm-software.de/ (letzter Zugriff am 10.10.16).

- Es gibt einen Katalog von möglichen Rezeptkategorien[21], welcher jedoch nicht ausschließlich zur Einordnung der Rezepte benutzt wird. Nahezu jedes der Rezepte wird durch weitere Stichwörter beschrieben. Diese deuten beispielsweise auf dominierende Zutaten, Zubereitungsformen ("Mikrowelle") oder regionale Küchen ("Spanien") hin.
- Neben Rezepten sind auch Hinweistexte u.ä. informierende Einträge enthalten. Für diese Rezepte werden entweder keine Zutaten oder genau eine Zutat zur inhaltlichen Ausrichtung (z.B. "Kartoffeln" als Zutat im vermeintlichen Rezept mit dem Titel "Alles über Kartoffeln") definiert.
- Es gibt keine einheitliche Notation zur Angabe alternativer oder optionaler Zutaten. Schlüsselwörter wie zum Beispiel "oder", "nach Wunsch", "evtl." finden sich in ähnlichen Schreibweisen in Zutatenangaben wieder.
- Beginnt eine gemäß *ingredone* ersetzte Zeile mit einem Bindestrich, deutet dies eine Fortführung der Zutatenangabe an. Dies zeigt der folgende Ausschnitt:

```
1     150 g  schwarze Johannisbeeren
2            - (TK)
```

- Einheiten sowie Mengenangaben sind nicht das Ergebnis einer Ersetzung der Symbole *amount* und *unit*, sondern sind im Zutatenbezeichner eingeschlossen. An dieser Stelle werden zusätzliche - nicht im spezifizierten Einheitenkatalog[22] enthaltene - Größen verwendet. Die folgenden Beispiele verdeutlichen dies:

```
1       1    Gläschen Mangosaft
2       1    Dose Champignons, 200g
3       1    Filet, falsch vom Rind
4            - (ca. 1,5 kg)
5       1    Pckg Mccain 1,2,3
6            - Kartoffelpuffer (=1500 g)
```

- Für Zubereitungszeiten und -formen sind in der EBNF-Grammatik keine gesonderten Symbole vermerkt, sodass diese Angaben entweder als Kategorie oder innerhalb der Zubereitungsanweisungen aufgeführt werden.

Die beschriebenen Unsauberkeiten der Rezeptsammlung erschweren die Entwicklung eines Parsers und darauf aufbauender Programmeinheiten.

Schnittstelle der Yummly-Plattform

Bei Yummly handelt es sich um ein Unternehmen, welches über eine mobile Anwendung und

[21] erreichbar unter http://kochbuch.unix-ag.uni-kl.de/kategorie.php, letzter Zugriff am 10.01.16
[22] erreichbar unter http://www.wedesoft.de/anymeal-api/mealmaster.html unter dem Abschnitt *Units*, letzter Zugriff am 12.10.16

über die Webseite www.yummly.com Rezept-Empfehlungen unter Berücksichtigung individueller Geschmäcker anbietet. Die Rezepte werden aus unterschiedlichen Quellen aggregiert. Die semantische Rezeptsuche kann von einem Nutzer durch Angaben zu Ernährungsweisen, Allergien, bevorzugten Zutaten, Geschmacksrichtungen (salzig, süß, bitter, pikant, etc.), Nährwertobergrenzen, gewünschten Zubereitungsarten, maximalen Zubereitungszeiten und speziellen Küchen (z.B. italienisch, indisch, spanisch, etc.) personalisiert werden. Über ein REST[23]-konformes *Application Programming Interface* (API) können Entwickler auf die umfangreiche Rezeptdatenbank Zugriff nehmen und die oben beschriebenen Suchfunktionen nutzen. Die Verwendung ist kostenpflichtig; es besteht jedoch die Möglichkeit, eine akademische Lizenz zu erwerben, mit welcher insgesamt 30.000 Anfragen kostenfrei getätigt werden dürfen. Eine solche Lizenz konnte akquiriert werden, sodass semantisch angereicherte Rezepte entsprechend der API-Dokumentation[24] im JSON[25]-Format angefordert werden können. Ein Beispiel für diese Rückgabestruktur zeigt die folgende JSON-Datei, welche bei der Ausführung des Requests **http://api.yummly.com/v1/api/recipe/Champignon—Zucchini—Pfanne-mit-Feta-911527** geliefert wird.

Listing 3.1: Ergebnis der Anforderung des Rezeptes mit der ID Champignon—Zucchini—Pfanne-mit-Feta-911527


```
{
   "yield": "2",
   "nutritionEstimates": [
      {
         "attribute": "ENERC_KCAL",
         "description": "Energy",
         "value": 348.16,
         "unit": {
            "id": "fea252f8-9888-4365-b005-e2c63ed3a776",
            "name": "calorie",
            "abbreviation": "kcal",
            "plural": "calories",
            "pluralAbbreviation": "kcal",
            "decimal": true
         }
      }
   ],
   "prepTimeInSeconds": 600,
   "totalTime": "10 min",
   "name": "Champignon - Zucchini - Pfanne mit Feta",
   "source": {
      "sourceDisplayName": "Chefkoch",
      "sourceSiteUrl": "chefkoch.de",
      "sourceRecipeUrl": "http://www.chefkoch.de/rezepte
         /1394251244387022/Champignon-Zucchini-Pfanne-mit-Feta.html"
```


[23] Representational State Transfer
[24] siehe https://developer.yummly.com/documentation, letzter Zugriff am 17.11.16
[25] JavaScript Object Notation

```
    },
    "prepTime": "10 Min",
    "id": "Champignon---Zucchini---Pfanne-mit-Feta-911527",
    "ingredientLines": [
        "1 Zucchini",
        "150 g Champignons",
        "150 g Feta-Käse",
        "Pfeffer, grob"
    ],
    "numberOfServings": 2,
    "totalTimeInSeconds": 600,
    "attributes": {},
    "flavors": {
        "Sweet": 0.1667,
        "Piquant": 0,
        "Sour": 0.5,
        "Salty": 0.3333,
        "Meaty": 0.5,
        "Bitter": 0.8333
    },
    "rating": 3
}
```


Aus Gründen der Übersichtlichkeit wurden das *nutritionEstimates*-Array auf ein Objekt reduziert und das *images*-Array sowie das *attribution*-Objekt entfernt. Die Struktur umfasst wesentliche, für den Aufbau des Wissensnetzes relevante, Datenfelder. Zu diesen zählen geschätzte Nährwertangaben (*nutritionEstimates*), geordnete Zutaten- und Mengenangaben (*ingredientLines*), Zubereitungszeiten (*totalTimeInSeconds*, *prepTimeInSeconds* und *cookTimeInSeconds*) und Portionsangaben (*numberOfServings*). Darüber hinaus wird ein Rezept bzw. ein Gericht hinsichtlich unterschiedlicher Geschmacksrichtungen bewertet (*flavors*). Da Yummly als eine Art soziales Netzwerk im Bereich Kochen auftritt, ist auch die durchschnittliche Nutzerbewertung (*rating*) Inhalt der Rezeptbeschreibung. Angaben zur Herkunft der Rezepte können dem *source*-Array entnommen werden. Anweisungen zur Zubereitung eines Gerichts sind nicht Teil der Response-Struktur.

Aus der Dokumentation geht hervor, dass der Rezeptname, die Rezept-ID sowie die Zutatenangaben stets definiert sind. Es wird vermutet, dass alle anderen Felder optional sind und die jeweiligen Schlüssel auf Nullwerte oder gar keine Werte verweisen. Hinsichtlich der bereitgestellten Nährwertinformationen wird angenommen, dass diese durch Yummly berechnet werden, da die ursprünglichen Rezeptbeschreibungen, welche durch das Feld *source* referenziert werden, entweder keine, weniger oder abweichende Nährwertangaben enthalten. Da nicht bekannt ist, wie diese Werte berechnet werden, kann keine Aussage zur Qualität dieser Information getroffen werden.

Ein Vorteil der API-Nutzung ergibt sich aus der Möglichkeit, parametrisierte Suchanfragen zu spezifizieren. Zum Beispiel lassen sich nur diejenigen Rezepte abfragen, welche bei einer

bestimmten Allergie oder Ernährungsweise verzehrt werden dürfen oder die einer bestimmten Menüart (Hauptgericht, Dessert, Frühstück, etc.) entsprechen. Für diese Parameter definiert Yummly sogenannte *Dictionaries*, welche die Wertebereiche in Form von Listen zusammenfassen. Die Dictionaries werden über die API abgefragt und im *JSON with Padding*-Format repräsentiert. Eine Aufschlüsselung der Werte der Parameter *diet, allergy* und *course* ist Tabelle A.2 in Anhang A.4 zu entnehmen.

Im Vergleich zur Rezeptsammlung der Unix-AG ist die Angabe der Zutaten weniger spezifisch, da keine Unterteilung in Mengen und Einheiten vorgenommen wird. Es ist unklar, ob die Einheiten bei der Integration in die Yummly Datenbank auf ein internes Vokabular abgebildet werden. Eine solche Zuordnung willkürlicher Einheitenbezeichner auf unmissverständliche Codes wird im Falle der im MealMaster-Format repräsentierten Rezepte angestrebt. Dass dieses Vorgehen nicht durchgängig eingehalten wird, wurde bei der Untersuchung der Sammlung gezeigt.
Aufgrund der erweiterten Suchfunktionalität und der genaueren Attribuierung eines Rezeptes durch Zubereitungszeiten und Nährwertangaben wird die Yummly API zum Aufbau der Wissensbasis genutzt. Da die Anzahl der Requests auf insgesamt 30.000 begrenzt ist und dennoch möglichst viele Rezepte beschafft werden sollen, muss eine sinnvolle Abfragestrategie entwickelt werden. Sinnvoll bedeutet, dass die Parametrisierung genutzt wird, um nur nach Rezepten mit vorhandenen Nährwertinformationen zu suchen und gleichzeitig untaugliche Rezepte auszuschließen. Als untauglich werden diejenigen Rezepte empfunden, welche aufgrund ihrer Zuordnung zu einem bestimmten Gang nicht als Menükomponenten infrage kommen. Im Hinblick auf die in Tabelle A.2 gelisteten Einträge werden Rezepte der Kategorien *Beverages, Condiments and Sauces* und *Cocktails* durch die Übergabe des Parameters *excludedCourse[]* ausgeschlossen. Es bestünde weiterhin die Möglichkeit, die Suche auf zu einer Ernährungsart passenden oder bei einer Allergie erlaubten Gerichte einzuschränken. Demgegenüber steht die Tatsache, dass aus den Resultaten nicht abgeleitet werden kann, welche weiteren Zusammenhänge existieren. Das heißt, die in der Antwort enthaltenen Informationen geben keinen Aufschluss darüber, ob ein Gericht auch bei anderen Allergien oder Ernährungsweisen verzehrbar ist. Im Sinne eines möglichst vollständig verwobenen Wissensnetzes muss dieses Faktenwissen nachträglich erschlossen werden. Diese Parametrisierung ergibt demnach keinen Benefit. Um sicherzustellen, dass bei der Suche nur Rezepte mit vorhandenen Nährwertangaben aufgefunden werden, wird in der Anfrage ein minimaler Energiegehalt (*nutrition.ENERC_KCAL.min*) von einer Kilokalorie gefordert.
Es wird die folgende **Abfragestrategie** gewählt: Unter Ausnutzung der Seitennummerierung

und zur Vermeidung von Zeitüberschreitungen werden insgesamt 20500 Suchresultate gemäß der folgenden Request-Struktur abgefragt.[26]

Methode	GET
URL	http://api.yummly.com/v1/api/recipes
Query-Parameter	maxResult=500
	excludedCourse[]=courseˆcourse-Beverages
	excludedCourse[]=courseˆcourse-Cocktails
	excludedCourse[]=courseˆcourse-Condiments and Sauces
	nutrition.ENERC_KCAL.min=1
	start=[0,500,1000,...,20000]

Jede Antwort wird zur weiteren Verarbeitung im JSON-Format gespeichert. Die ID eines im Suchresultat referenzierten Rezeptes wird erfasst, sodass eine wiederholte Abfrage desselben Rezeptes vermieden wird. Für jedes Rezept wird die vollständige Beschreibung angefordert und als Datei gespeichert.

Auswahl einer Produktdatenbank

Die Beschaffung der in E aufgeführten Daten setzt das Vorhandensein einer mit dem BLS verknüpften Produktdatenbank voraus. Kandidaten für solche Datenbanken sind die von der *saymo* GmbH entwickelte *foodbase*[27] oder die Ernährungsdatenbank *NutriBase*®[28]. Da beide Datenbanken über keine frei zugängliche Abfrageschnittstelle verfügen, scheidet eine Verwendung aus.
Beim Aufbau des Wissensnetzes liegt der Schwerpunkt auf der Integration der Lebensmittel und Rezepte. Die nachfolgenden Abschnitte konzentrieren sich daher auf die Behandlung und Aufbereitung der zugrundeliegenden Datenquellen.

3.4.2 Informationsmodellierung

In diesem Abschnitt wird die Modellierung mit der Durchführung der Schritte 4, 5 und 6 des Ontologie-Entwicklungsleitfadens abgeschlossen. Ausgehend von dem in der Vorarbeit spezifizierten Informationsmodell wird die Definition der Klassen und Klassenhierarchien, der Eigenschaften und der Beschränkungen reflektiert und an die veränderte Domäne angepasst.

[26] Die im Zuge der Lizenzierung erhaltenen Zugangsdaten (Applikations-ID und -Schlüssel) werden durch Request-Header übermittelt.
[27] saymo Lebensmittel Online Shop: foodbase - die Lebensmittel-Produktdatenbank https://www.saymo.de/mehrwertdienste/foodbase/, letzter Zugriff am 12.10.16
[28] Nutri-Science GmbH : NutriBase® http://www.nutri-science.de/software/nutribase.php, letzter Zugriff am 12.10.16

Das Ergebnis dieser Aufarbeitung wird in Form von Signaturen (siehe Abschnitt A.5 im Anhang) spezifiziert und grafisch in Abbildung 3.5 visualisiert. Zur Angabe der Kardinalitäten wird die Erweiterte Backus-Naur-Form (EBNF) mit der folgenden Notation verwendet:

- ? entspricht 0..1
- * entspricht 0..*
- + entspricht 1..*

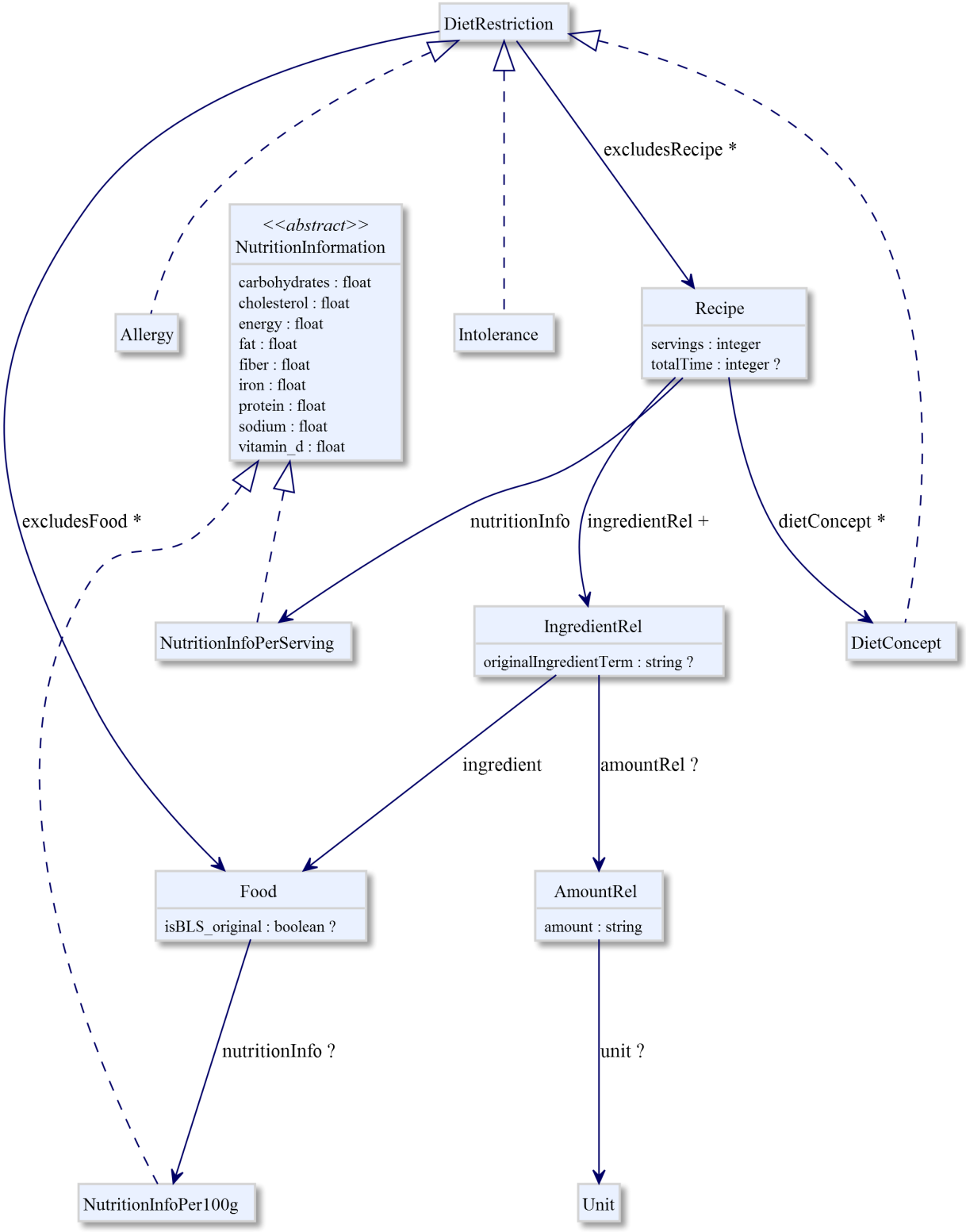


Abbildung 3.5: Informationsmodell

Es ist zu beachten, dass domänenunabhängige Konstrukte (beispielsweise Codesysteme, -versionen, Konzepte, Bezeichner und Relationen zwischen diesen Entitäten) durch das CTS2-

Informationsmodell spezifiziert werden. Das vorliegende Modell ist demnach als Erweiterung zu verstehen.

Das Modell umfasst Klassen (im Sinne einer *rdfs:Class*, dargestellt als Kästen) und Relationen (gemäß einer *rdf:Property*, dargestellt als Attribute einer Klasse oder als durchgezogene Linien mit Pfeilenden). Gestrichelte Linien mit nicht gefüllten Pfeilen dienen der Repräsentation von *rdfs:subClassOf*-Properties.

Zu den wichtigsten Änderungen gegenüber dem ursprünglichen Modell zählen:

Entfernen der Klassen *DietTherapy* und *Religion*

Auf die Integration diätischer Therapien (z.B. Reduktionsdiäten, natriumarme oder fettarme Diäten) wird verzichtet, da zum einen für viele Therapien die genaue Auswirkung auf die Ernährungsweise nicht klar definiert ist und zum anderen davon ausgegangen wird, dass sich die Therapien durch nutzerspezifische Grenzwerte für Nährstoffe repräsentieren lassen. Gleichermaßen schwierig gestaltet sich die allgemeingültige Definition von nicht zu verzehrenden Lebensmitteln für Religionen, sodass dieses Konzept ebenfalls aus dem Modell entfernt wurde.

Anpassung der Properties der Klasse *NutritionInformation*

Die Properties der Klasse *NutritionInformation* werden unter Berücksichtigung des vorliegenden Faktenwissens zu Rezepten und Lebensmitteln angepasst. Für Instanzen der Konzepte *Recipe* und *Food* wird nun gefordert, dass Mengenangaben zu Kohlenhydraten, Cholesterin, Kilokalorien, Fetten, Ballaststoffen, Eisen, Proteinen, Natrium und Vitamin D vorliegen. Diese beziehen sich im Falle eines Rezeptes auf eine Portion (gekapselt durch eine Instanz der Klasse *NutritionInfoPerServing*) oder im Falle eines Lebensmittels auf 100g (gekapselt durch eine Instanz der Klasse *NutritionInfoPer100g*).

Anpassung der Properties der Klassen *Recipe, IngredientRel, AmountRel* und *Food*

Auch diese Modifikationen sind auf fehlende Datensätze zurückzuführen. Beispielsweise sind keine Zubereitungsangaben für Rezepte (zuvor *instructions*) oder Preise (ehemals *pricePerServing* bzw. *pricePer100g*) gegeben. Zur Spezifizierung der Zubereitungszeiten wird die Property *totalTime* definiert. Aufgrund der manuellen Rezeptbeschaffung und -aufbereitung war es im Vorgängerprojekt möglich, Angaben zur Verarbeitungsform einer Zutat und zur Quantifizierung von Einheiten (z.B. eine kleine oder große Dose, ein gestrichener oder gehäufter Teelöffel) zu erfassen und auf Modellebene widerzuspiegeln. Eine Ableitung dieser Attribute ist in Anbetracht der verfügbaren Daten nicht umsetzbar und führt zu keinem Mehrwert.

Die Klassen *Food* und *IngredientRel* wurden um die Relationen *isBLS_original*, respektive *originalIngredientTerm* erweitert. Erstere wird eingeführt, um ein Lebensmittel als originalen

BLS Eintrag oder als eine im Rahmen der Hierarchie-Erweiterung (siehe Abschnitt 3.4.3) hinzugefügte Instanz zu kennzeichnen. Die Property *originalIngredientTerm* verweist auf den in einem Rezept angegebenen Zutatenbezeichner. Dieser stellt den Ausgangspunkt für das Matching auf BLS Einträge dar und wird aus Gründen der Nachvollziehbarkeit gespeichert. Um fehlende Nährwertangaben für Lebensmittel zu tolerieren, wird die Kardinalität der Property *nutritionInfo* zwischen den Klassen *Food* und *NutritionInfoPer100g* zu 0..1 geändert. Weiterhin wurden alle Relationen der Klasse *DietRestriction* entfernt und die Kardinalität der Property *dietConcept* dahin gehend angepasst, dass ein Rezept keiner, einer oder mehreren Ernährungsweisen zugeordnet werden kann.

3.4.3 Integration des Wissens

Dieser Abschnitt beschreibt, wie die zuvor festgelegten Konzepte auf das Informationsmodell abgebildet werden und wie das freistehende Faktenwissen zu einem Wissensnetz aggregiert wird.

Integration des Bundeslebensmittelschlüssels

Der BLS dokumentiert die durchschnittlichen Nährstoffgehälter von über 14.000 Lebensmitteln. Motiviert wurde die Entwicklung dieser Datenbank durch den Wunsch nach einem Instrument zur Auswertung von ernährungsepidemiologischen Studien. Der BLS definiert in der aktuellen Version für jedes Lebensmittel 138 Nährstoffwerte, welche das Ergebnis von Analysen und Berechnungen sind. Für jeden Eintrag sind eine deutsche und englische Bezeichnung gegeben. [HSS16]
Ein Lebensmittel im BLS wird durch einen sieben-stelligen, sprechenden Schlüssel eindeutig identifiziert. Die ersten vier Stellen bilden den *Grundschlüssel*, wobei die erste Stelle die Art, die zweite Stelle die Gruppe und die dritte und vierte Stelle das Einzellebensmittel beschreiben. Die fünfte Stelle dient der näheren Spezifizierung und/oder der Codierung der industriellen Verarbeitung. Das Garverfahren und die Zubereitung werden in der sechsten Stelle verschlüsselt. Die letzte Stelle definiert das Bezugsgewicht. Das Schlüsselsystem ist hierarchisch aufgebaut, sodass durch das Nullsetzen einer Stelle eine Generalisierung erreicht wird.
Die Art (oder auch Hauptgruppe) wird durch einen Buchstaben von B bis Y angegeben, alle weiteren Stellen entsprechen Zahlen. Die mit X und Y kodierten Hauptgruppen umfassen Rezepte bzw. Menükomponenten. Für diese sind jedoch keine Zutaten- und Mengenangaben enthalten. Anhand der BLS Einträge

K110022 Kartoffeln geschält gegart

K110200 Kartoffeln geschält tiefgefroren

K110902 Kartoffeln geschält Konserve abgetropft

soll das Schlüsselsystem exemplarisch erläutert werden. Die ersten zwei Stellen klassifizieren die Lebensmittel als Kartoffeln in der Hauptgruppe *Kartoffeln und Kartoffelerzeugnisse, stärkereiche Pflanzenteile, Pilze*. Die darauffolgenden zwei Stellen dienen der weiteren Spezialisierung der Einzellebensmittel, sodass alle Einträge, welche geschälte Kartoffeln näher beschreiben, ebenfalls durch diese Ziffern gekennzeichnet sind. Die fünfte Stelle klassifiziert die Lebensmittel als handelsüblich (0), tiefgefroren (2) und Konserve (9). Eine 0 an der sechsten Stelle deutet auf ein handelsübliches, nicht zubereitetes Lebensmittel hin, während eine 2 zur Deklaration gegarter, erwärmter, gargezogener oder panierter Lebensmittel genutzt wird. Die letzte Stelle wird zur Kodierung des Bezugsgewichts genutzt. Die jeweiligen Nährstoffwerte beziehen sich auf ein handelsübliches Lebensmittel ohne Küchenabfall (0) und auf ein zubereitetes, abgegossenes Lebensmittel ohne Küchenabfall (2).
Diese Beispiele betonen, mit welcher Sorgfalt das Schlüsselsystem entwickelt wurde. Gleichwohl wurden bei der Verwendung des BLS im Rahmen der Studienarbeit einige Auffälligkeiten beobachtet. Zum einen wird die Strukturvorgabe, nur an der ersten Stelle des Schlüssels einen Buchstaben zu verwenden, durch einige BLS Einträge verletzt.[29] Zum anderen wurde festgestellt, dass der Grundschlüssel mitunter doppelt zur Kodierung ähnlicher Lebensmittel verwendet wurde.[30]

Erweiterung der Hierarchie des BLS

Grundsätzlich definiert die vom Max-Rubner-Institut bereitgestellte Quelldatei zur maschinellen Verarbeitung des BLS kein hierarchisches Geflecht. Die ein- und zweistelligen Gruppenbegriffe (z.B. Brot und Kleingebäck (B) und Broterzeugnisse (B8)) werden lediglich im Handbuch aufgeführt. In diesem wird ausgesagt, dass sich durch Streichung (Nullsetzen) einer Stelle der Gruppenbegriff ergeben soll. Dies ist nicht gänzlich nachvollziehbar, da zur Erreichung des Gruppenbegriffs alle auf die zweite Stelle folgenden Ziffern gestrichen werden müssten. Grundsätzlich kann daher von einer dreistufigen Hierarchie ausgegangen werden, auf deren unterster Ebene die eigentlichen Lebensmittel mit Nährwertangaben angesiedelt sind. Soll der BLS im Zuge von ihn kapselnden Anwendungen einem Benutzer präsentiert

[29] Beispiele hierfür sind die Getränke Diätbier (P1A4000) und Bier mit Cola (P1A3000).
[30] Für das Lebensmittel Blattspinat werden mehrere Verarbeitungsformen mit den Grundschlüsseln *G211* und *G210* definiert. Letzterer wird auch zur Klassifizierung von Spinat verwendet (z.B. Spinat Trunk (G210700), Spinat Konserve (G210902).

werden, ist eine weitere Strukturierung aufgrund der Genauigkeit des Schlüsselsystems dringend erforderlich. Es werden zwei Strategien zur Erweiterung der Taxonomie verfolgt:

1. Subsumtion bestehender BLS Einträge
2. Subsumtion durch Hinzufügen neuer BLS Einträge mit dreistelligen Schlüsseln

Eine genaue Beschreibung dieser Vorgehensweisen ist in Abschnitt A.6 im Anhang gegeben. Durch das Hinzufügen von 442 Konzepten konnte die Navigation durch den Hierarchiebaum erleichtert werden. Unter Berücksichtigung der Besonderheiten des Schlüsselsystems ist dieser Ansatz dennoch als unausgereift zu werten. Idealerweise wird eine Beschreibung eines Einzellebensmittels durch verschiedene Attribute vorgenommen. Die Attribute könnten dann den durch die fünfte, sechste und siebte Stelle spezifizierten Eigenschaften entsprechen. Je nach Gruppenbegriff sind dies die Verarbeitung oder Spezifizierung, die Zubereitung, das Garverfahren, die Fettstufe oder das Bezugsgewicht. Dies entspräche einer *Achsenbildung*, wie sie beispielsweise im LOINC[31]-System implementiert ist.
Der BLS wird in dieser erweiterten Form unter Ausschluss der Gruppen X (Menükomponenten überwiegend pflanzlich) und Y (Menükomponenten überwiegend tierisch) auf die Klassen *Food* und *NutritionInfoPer100g* abgebildet, wobei die jeweiligen Instanzen über die Relation *nutritionInfo* vernetzt sind. Für Einträge mit ein- und zweistelligen Schlüsseln sind keine Nährwertangaben gegeben. Ebenfalls wurde auf die Ableitung von Nährwertangaben für Einträge, welche durch die Hierarchie-Erweiterung hinzugefügt wurden, verzichtet.

Integration der über Yummly abgefragten Rezepte

Die Integration der Rezepte beginnt mit einem zweiteiligen Filterungsprozess, durch welchen a.) Rezepte mit unzureichenden Nährwertangaben und b.) Rezepte, für deren Zutatenbezeichner keine adäquate Zuordnung eines BLS Eintrages erfolgen kann, von der weiteren Verarbeitung ausgeschlossen werden.
In a.) wird gefordert, dass ein Rezept Mengenangaben zu Kohlenhydraten, Cholesterin, Kilokalorien, Fetten, Ballaststoffen, Eisen, Proteinen, Natrium und Vitamin D umfasst. Dadurch wird gewährleistet, dass die Modellrestriktionen bezüglich der Relationen der Klasse *NutritionInformation* eingehalten werden.
Die in b.) benannte Zuordnung von BLS Einträgen führt dazu, dass die Zutatenbezeichner durch die Verwendung eines standardisierten Vokabulars disambiguiert werden. Dieser Schritt ist wichtig, da auch die Nutzer der Applikation ihre individuellen Lebensmittelpräferenzen

[31] Logical Observation Identifiers Names and Codes; ein Begriffssystem zur Verschlüsselung von medizinischen Untersuchungen, http://loinc.org/, letzter Zugriff am 2.12.16

durch Konzeptmengen aus eben diesem Vokabular ausdrücken.

Das Matching wird durch eine Vorverarbeitung der im JSON Array *ingredientLines* gegebenen Zutaten eingeleitet. Zunächst werden unter Zuhilfenahme regulärer Ausdrücke Klammerausdrücke und Stoppwörter entfernt. Zu Stoppwörtern zählen neben herkömmlichen Termen wie “und“, “in“, “ca.“ auch Phrasen zur näheren Spezifizierung eines Lebensmittels (“tiefgekühlt“, “unbehandelt“, “zerteilt“, “Bio“), Herstellernamen (“KNORR“, “PFANNI“) oder Quantifizierungen (“klein“, “mittelgroß“). Die Stoppwörter werden in Flexionen für die Merkmale Kasus, Numerus und Genus in einer Liste zusammengefasst. Im Zuge der Vorverarbeitung werden Mengenangaben und Einheiten entfernt und zwischengespeichert für den Fall, dass das Rezept die Filterung erfolgreich durchläuft. Um die Erkennung von Einheiten zu erleichtern, werden jeweils alternative Schreibweisen und Abkürzungen (z.B. Teelöffel, Teel., TL, Tl, tl für die Einheit *teaspoon*) definiert. Anschließend werden häufig vorkommende Lebensmittelbezeichner in der verbleibenden Zeichenkette durch Terme, welche lexikographisch den BLS-Bezeichnern näher sind, ersetzt. Während der Erprobung des Matching-Verfahrens wurde festgestellt, dass dadurch mehr und bessere Zuordnungen erzielt werden. Zum Beispiel werden Singulare durch Plurale oder spezifische durch allgemeine Lebensmittel ausgetauscht.[32]

Für das Matching werden die folgenden Funktionalitäten des Terminologiedienstes kombiniert:

- Konzept-Matching
- Index-basierte Suche
- Vorschläge bei der Suche

Das Konzept-Matching liefert für eine Zeichenkette und ein Vokabular eine Menge von Kandidaten aus eben jenem Vokabular. Für jedes Konzept wird ein Score zwischen 0 und 1 geliefert, wobei 1 der bestmöglichen Übereinstimmung zwischen Zeichenkette und Konzept entspricht. Das Matching ist als streng zu bewerten und führt häufig selbst dann zu keinem Ergebnis, wenn ein gegebener Term vollständig mit einem Konzeptbezeichner übereinstimmt. Die Index-basierte Suche sowie die Vorschlagsfunktionalität werden in der Web-Applikation des Terminologieservers intensiv zum Wiederauffinden von Konzepten genutzt und eignen sich auch als Matching-Werkzeug. Die Suche bewertet ebenfalls jedes Resultat mit einem Wert zwischen 0 und 1, während die Vorschläge in einer geordneten Liste mit maximal 20 Konzepten ohne Scores zusammengefasst werden.

[32] Die Stoppwörter, alternative Schreibweisen und Abkürzungen für Einheiten sowie die Ersetzungen können den Dateien “stopwords.csv“, “unitMapping.csv“ und “ingredientReplacements.csv“ auf der beigelegten DVD entnommen werden.

Abbildung 3.6 veranschaulicht den Ablauf des Matchings einer vorbereiteten Zeichenkette auf einen BLS Eintrag.

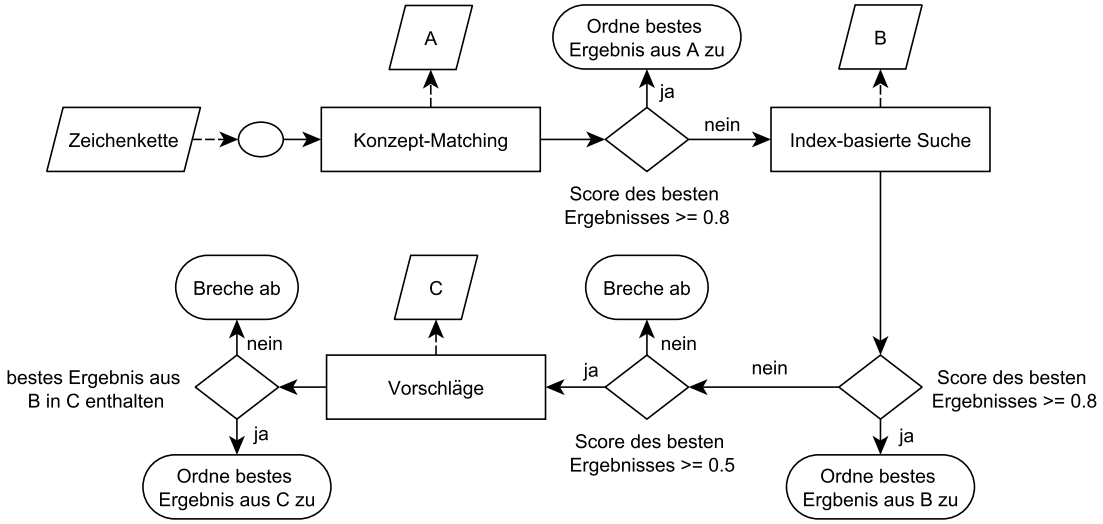


Abbildung 3.6: Ablauf des BLS-Matchings

Zu Beginn wird das Konzept-Matching angewandt. Die daraus resultierende Liste (in der Abbildung als *A* bezeichnet) wird absteigend nach Score und nach Länge des Konzeptschlüssels sortiert, sodass die Zuordnung spezifischer Konzepte begünstigt wird. Wird das beste Konzept aus *A* mit einem Score größer gleich 0.8 bewertet, wird dieses zugeordnet und das Matching ist beendet. Fällt die Bewertung schlechter aus, wird die Suche genutzt und die beschriebene Sortierung und Auswahl wird bei der Verarbeitung der durch *B* bereit gestellten Konzepte wiederholt. Ist der Score des besten Konzeptes aus B kleiner als 0.8, jedoch größer oder gleich 0.5, schließt sich eine Abfrage der Vorschläge an. Enthält die als *C* bezeichnete Liste das beste Ergebnis der Suche, erfolgt eine Zuordnung dieses Konzeptes. Andernfalls endet der Prozess ohne eine Zuordnung. Anhang A.7 listet Beispiele für erfolgreiche (in dem Sinne, dass ein BLS Konzept gemäß des beschriebenen Verfahrens gefunden wurde) und nicht erfolgreiche Zuordnungen.

Die Filterung beginnt mit 21918 verschiedenen Rezepten. Von diesen sind für 16499 Rezepte die geforderten Nährwertangaben vorhanden. Der überwiegende Teil der Rezepte scheidet im zweiten Schritt aus, sodass letztlich 2423 Rezepte integriert werden können.

Dafür wird jedes Rezept durch eine Menge von Instanzen und Relationen entsprechend des Informationsmodells repräsentiert. Namentlich sind dies Instanzen der Klassen *Recipe*, *NutritionInfoPerServing*, *IngredientRel* und *AmountRel*, welche durch eine Vielzahl an Properties näher beschrieben bzw. miteinander in Beziehung gesetzt werden. Die Relationen *unit* und *ingredient* referenzieren die Instanzen der Klassen *Unit* und *Food*, welche im Rahmen der

Filterung zugeordnet wurden.

Die zum Aufbau des Faktenwissens benötigten Begriffssysteme sind Tabelle 3.2 zu entnehmen. Für jedes Vokabular sind die verwendete Version und die Anzahl der aufgegriffenen Begriffe angegeben.

Tabelle 3.2: Anzahl und Ursprung des verwendeten Faktenwissens

Konzept	Vokabular	Anzahl
Lebensmittel und Nährwertdaten	Bundeslebensmittelschlüssel, Version 3.02	11386 (davon 442 abgeleitete Einträge)
Allergien	MeSH, Version 2014	5
	SNOMED-CT, Version 2015	1
Unverträglichkeiten	MeSH, Version 2014	2
Ernährungsweisen	MeSH, Version 2014	2
Einheiten	UCUM, Version 2013	9
	zusätzliches Einheitenvokabular	23
Rezepte	Yummly	2423

Eine genaue Auflistung der ausgewählten Begriffe für die Konzepte Allergien, Unverträglichkeiten, Ernährungsweisen und Einheiten findet sich in Tabelle A.1 im Anhang.

Vernetzung

Die Wissensbasis wird durch die Vernetzung der Instanzen komplettiert. Dabei werden die in der Studienarbeit beschriebenen Techniken[33] teilweise wiederverwendet, sodass das relationale Wissen **manuell** oder **regelbasiert** integriert wird.
Manuell bedeutet, dass Aussagen der Form *Subjekt Prädikat Objekt* aus einer händisch angefertigten Datei eingelesen, durch einen Interpreter verarbeitet und modellkonform zur Wissensbasis hinzugefügt werden. Gemäß dieses Vorgehens erfolgte die Vernetzung von Instanzen der Klassen *DietRestriction* und *Food* über die Relation *excludesFood*. Dabei wird die Hierarchie des BLS gezielt ausgenutzt, um eine Assoziation bei Bedarf auch auf die Kinder einer Ziel-Instanz auszuweiten. Beispielsweise kann so ausgedrückt werden, dass bei einer Nuß-Hypersensitivität alle Lebensmittel der Gruppe *Nüsse* (Schlüssel *H1*) - und damit alle Instanzen, welche dieser Ressource direkt oder indirekt untergeordnet sind - nicht

[33] siehe Abschnitt 4 in der beigefügten Studienarbeit

verzehrt werden dürfen.
Beim regelbasierten Ansatz wird das vorhandene Wissen genutzt, um algorithmisch auf Quellcode-Ebene neue Zusammenhänge zu erschließen. Dies bietet sich zur Ableitung der Relationen *exludesRecipe* (zum Ausschluss von Gerichten bei einer bestimmten Einschränkung) und *dietConcept* (zur Zuordnung einer Ernährungsweise) an. Eine der umgesetzten Regeln zieht beispielsweise den Schluss, dass ein Gericht dann nicht bei einer Einschränkung verzehrt werden darf, wenn es ein Lebensmittel enthält, welches über die Relation *exludesFood* referenziert wird. Für eine genauere Erläuterung der angewandten Regeln sei auf Abschnitt 4.4 der Studienarbeit verwiesen.

Erfüllbare Anforderungen an die Menüplanung

Die eingeschränkte inhaltliche Ausdrucksstärke des vorliegenden Wissensnetzes macht eine Anpassung der ursprünglich definierten Anforderungen an die Menüplanung erforderlich.
Da für Lebensmittel keine Produktinformationen in Form von herkömmlichen Packungsgrößen, Preisen oder Haltbarkeitsvermerken beschafft werden konnten, können die Anforderungen A7, A8 und A11 nicht umgesetzt werden. Gleiches gilt aufgrund der fehlenden Angabe der Zubereitungsformen für Anforderung A9. Bezogen auf die Formalisierung bedeutet dass, dass die Ausdrücke 2.4, 2.11, 2.12, 2.18, 2.20 sowie alle Mengen und Parameter, welche als Terme ausschließlich in diesen Ausdrücken verwendet werden, für die Umsetzung des Lösungsverfahrens nicht relevant sind.

4 Implementierung der Web-Applikation

Dieses Kapitel thematisiert die Implementierung der Web-Applikation und beschreibt wesentliche Arbeitsschritte und Entwurfsentscheidungen. Zunächst werden die Auswahl einer Java-Bibliothek und die darauf aufbauende Umsetzung des konzipierten Lösungsansatzes geschildert. Anschließend werden die Server- und Client-seitigen Programmkomponenten erläutert. Das Kapitel schließt mit einer Evaluation der implementierten Optimierung.

4.1 Auswahl einer GA-Bibliothek

Es existieren zahlreiche Programmbibliotheken, welche die Entwicklung eines GA durch die Bereitstellung benötigter Funktionalitäten erleichtern. Nachfolgend werden Java-Bibliotheken, welche den umzusetzenden NSGA-II Algorithmus implementieren, miteinander verglichen. Anhand der Kriterien

A Zugänglichkeit des Quellcodes

B Bereitstellung von Analysen zur Auswertung des GA

C Aktualität

D Qualität der Dokumentation

E Aktivität der Community

wird eine geeignete Bibliothek ausgewählt und zur Erstellung und Ausführung eines GA zur Lösung des Menüplanungsproblems benutzt. Die zu vergleichenden Bibliotheken sind:

- ECJ Evolutionary Computation and Genetic Programming System[1]
- jMetal Metaheuristic Algorithms in Java [DN11]
- OPT4J [LGRT11]
- MOEA - A Free and Open Source Java Framework for Multiobjective Optimization[2]

[1] https://cs.gmu.edu/~eclab/projects/ecj/ ECJ 23 A Java-based Evolutionary Computation Research System, letzter Zugriff am 06.02.17

[2] http://moeaframework.org/index.html MOEA Website, letzter Zugriff am 06.01.17

Tabelle 4.1: Vergleich von vier Java-GA-Bibliotheken

Kriterium	ECJ	jMetal	OPT4J	MOEA
A	Open-Source-Projekt[3]	Open-Source-Projekt[4]	Open-Source-Projekt[5]	Open-Source-Projekt[6]
B	Ausdruck statistischer Werte in Dateien, keine Visualisierungen	Bereitstellung von Qualitätsindikatoren (z.B. *Generational Distance, Inverted Generational Distance*), keine Visualisierungen	Überwachung des Optimierungsprozesses vermutlich nur im interaktiven Modus möglich	(visualisierte) Bereitstellung von Qualitätsindikatoren sowie Laufzeitparametern
C	Version 23 veröffentlicht im Juni 2015	Version 5.1 veröffentlicht im Juli 2015	Versionierung unklar, letztes Update in Code-Repository im Juli 2016	Version 2.12 veröffentlicht im Januar 2017
D	umfassendes Handbuch [Luk15]	verständliche, jedoch lückenhafte Dokumentation[7]	Technisches Papier, kurzes Tutorial und Javadoc	zwei Handbücher (ein frei verfügbares sowie ein kostenpflichtiges umfängliches Handbuch), API Spezifizierung
E	Kommunikation über Mailing-Listen, hohe Reaktionsbereitschaft	aktive Community bestehend aus Hauptentwicklern und Mitwirkenden	Diskussionsforum mit hoher Reaktionsbereitschaft	aktives Diskussionsforum mit Möglichkeit zur Fehlermeldung, hohe Antwortbereitschaft des Hauptentwicklers

Aus Tabelle 4.1 geht hervor, dass alle untersuchten Bibliotheken quelloffen sind. Dies ist von großer Bedeutung, da dadurch die Definition und Behandlung spezifischer Individuenrepräsentationen und damit einhergehender angepasster genetischer Operatoren ermöglicht wird. Dabei ist der Implementierungsaufwand abhängig von der jeweilig umgesetzten Architektur

[3] ECJ Github-Repository unter https://github.com/eclab/ecj, letzter Zugriff am 08.01.17
[4] jMetal Github-Repository unter https://github.com/jMetal, letzter Zugriff am 08.01.17
[5] OPT4J Repository unter https://sourceforge.net/projects/opt4j/, letzter Zugriff am 08.01.17
[6] MOEA Github-Repository unter https://github.com/MOEAFramework/MOEAFramework, letzter Zugriff am 08.01.17
[7] jMetal User Manual unter https://github.com/jMetal/jMetalDocumentation, letzter Zugriff am 08.01.17

sowie von Umfang und Qualität der Dokumentation. Letztere ist im Falle von ECJ und MOEA als zufriedenstellend zu bewerten. Die Zeitpunkte der letzten Veröffentlichungen und die Diskussionen der beteiligten Entwickler und Nutzer lassen vermuten, dass alle Bibliotheken aktiv gepflegt und weiterentwickelt werden. Letztendlich fällt die Wahl auf das MOEA-Framework, da dieses erweiterte Möglichkeiten der Auswertung bietet und besonders umfänglich dokumentiert ist.

4.2 Lösung des Menüplanungsproblems mit Hilfe des MOEA-Frameworks

Das MOEA-Framework begrüßt die Entwicklung problemspezifischer Komponenten durch die Bereitstellung von Interfaces und abstrakten Klassen. Zu diesen zählen **Problem, AbstractProblem, Initialization, Variable** und **Variation**. An dieser Stelle seien noch die Schnittstellen **Selection** und **TerminationCondition** genannt. Eine individuelle Einrichtung dieser Komponenten ist im Kontext der Menüplanung nicht notwendig, da die vom Framework bereitgestellten Standardwerkzeuge problemlos genutzt werden können.
Das in Abbildung 4.1 dargestellte Klassendiagramm zeigt, wie die Schnittstellen implementiert und wie von abstrakten Klassen geerbt wird. Die Implementierung von Schnittstellen wird im Diagramm durch gestrichelte Kanten mit nicht gefüllten Pfeilenden dargestellt. Durchgängige Kanten mit ebenfalls nicht ausgefüllten Pfeilenden visualisieren eine Vererbungsrelation. Die gerichteten Assoziationen zwischen zwei Klassen kennzeichnen, dass Objekte der Zielklasse als Variablen in der Quellklasse genutzt werden. Der Kantenname gleicht dabei dem Variablennamen. Die Abbildung umfasst die Kernentitäten und vernachlässigt diejenigen Klassen, Variablen und Methoden, welche nicht zum Verständnis beitragen. Für eine vollständige Begutachtung sei auf das auf dem DVD-Anhang hinterlegte Projekt *PersonalSmartMenu* verwiesen.
Die Klasse **MenuPlan** kapselt das für einen Menüplan spezifische Wissen. Dazu zählen die Anzahl der zu planenden Tage (*numberOfDays*), die Mahlzeiten eines Tages (*slotsPerDay*) sowie die von den Nutzern angegebenen tagesbezogenen Nährstoffgrenzwerte (*dayToUserToNutritiveRestrictions*). Die Klasse **MenuPlanningProblem** erbt von der Klasse **AbstractProblem**, welche wiederum das Interface **Problem** implementiert. Aufgrund dieser Struktur muss das MenuPlanningProblem die Methoden *evaluate* und *newSolution* sowie einen Konstruktor ausformulieren.

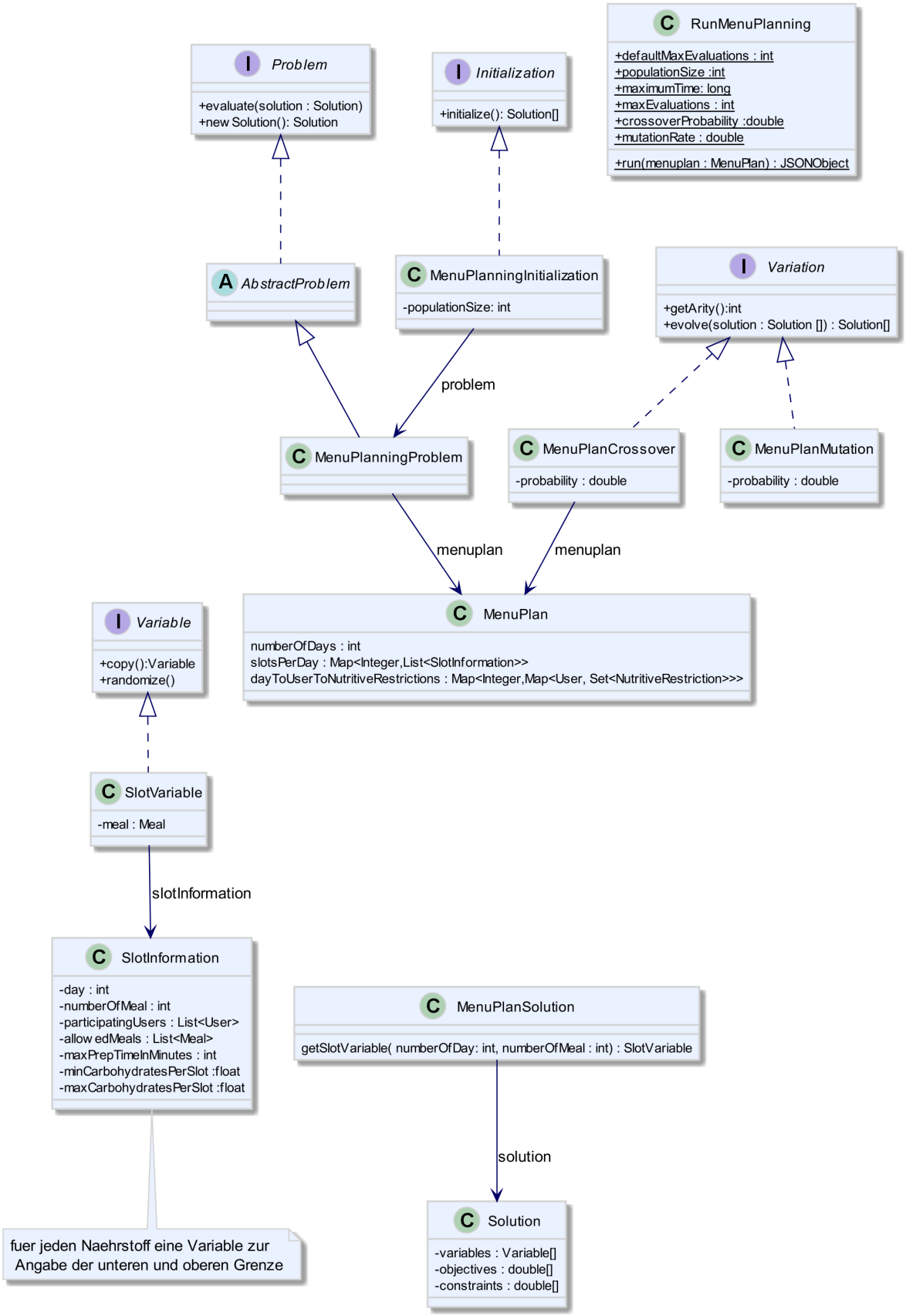


Abbildung 4.1: Klassendiagramm zur Veranschaulichung der implementierten EA-Komponenten

Die Methode *evaluate* erwartet als Argument eine Lösung, für welche die Werte der Zielfunktionen sowie die Constraint-Verletzungen berechnet werden. Die Methode *newSolution* bereitet eine neue Lösung vor und gibt diese zurück. Dies bedeutet, dass eine Instanz der Klasse Solution erstellt wird und Variablen vom Typ **SlotVariable** initialisiert werden. Eine SlotVariable implementiert das Interface **Variable** und repräsentiert einen Parameter einer Lösung. An dieser Stelle weicht die Implementierung geringfügig vom Lösungsansatz ab, da jeder Parameter mit einem Objekt der Klasse **Meal** und nicht mit einer Ganzzahl belegt wird. Diese Zuordnung wird willkürlich in der zu implementierenden Methode *initialize* in der Klasse **MenuPlanningInitialization** vorgenommen. Die Menge der zulässigen Gerichte wird durch die Variable *slotInformation* der Klasse SlotVariable bereitgestellt. Dieses Objekt trägt auch die Information über teilnehmende Nutzer (*participatingUsers*). Ferner bildet es die Slot-bezogenen Restriktionen zu Zubereitungszeiten (*maxPrepTimeInMinutes*) und Nährstoffgrenzen (beispielsweise durch die Variablen *minCarbohydratesPerSlot* und *maxCarbohydratesPerSlot*) ab.
Eine Instanz der Klasse **Solution** beschreibt eine Lösung durch die drei Felder *variables*, *objectives* und *constraints*. Diese enthalten die Werte der Parameter, die Werte der Zielfunktionen und die Werte der Constraint-Verletzungen. Die durch *variables* referenzierten Variablen können nur über einen einstelligen Elementindex abgefragt werden. Um den Zugriff wie in Abschnitt 3.3 dargestellt über ein Paar (t, m) zu realisieren, wird die Klasse **MenuPlanSolution** eingeführt. Diese ermöglicht die Indexierung einer SlotVariablen anhand eines Tages t und einer Mahlzeit m.
Die genetischen Operatoren werden durch die Klassen **MenuPlanCrossover** und **MenuPlanMutation** umgesetzt. Diese implementieren das Interface **Variation** und realisieren innerhalb der Methode **evolve** die im Lösungsansatz konzipierten Ein-Elter und 2-Elter-Operatoren. Die Anzahl der für die jeweilige Operation benötigten Individuen wird in der Methode *getArity* spezifiziert. Die Anwendung der Mutation und der Rekombination werden über Wahrscheinlichkeiten (*probability*-Variablen) gesteuert.

Ablauf des evolutionären Verfahrens

Ausgangspunkt der Optimierung ist stets eine konkrete Probleminstanz - repräsentiert durch ein Objekt der Klasse **MenuPlan**. Bei der Erstellung dieser Instanz werden die Wertebereiche der zu belegenden Problemparameter auf zulässige Gerichte beschränkt. Dieser Arbeitsschritt wird vom Terminologiedienst bewerkstelligt. Für jeden zu belegenden Slot werden verzehrbare Gerichte durch eine SPARQL-Anfrage ermittelt und in dem entsprechenden *SlotInformation*-Objekt hinterlegt. Die Zulässigkeit wird beeinflusst von diätischen

Einschränkungen, Nährstoffgrenzen sowie Zubereitungszeiten. Bezugnehmend auf das in Abschnitt 3.4.2 vorgestellte Informationsmodell findet sich dieses Wissen in den Entitäten *DietRestriction* und *NutritionInformationPerServing* und *Recipe* (genauer im Attribut *totalTime*) wieder. Jede SPARQL-Query basiert auf der in Listing 4.1 dargestellten Struktur. Diese beschreibt das abzufragende Graph-Muster durch eine Menge von Tripeln.

Listing 4.1: Grundstruktur der SPARQL-Query zur Abfrage zulässiger Gerichte

```
1  @prefix : <http://fokus.fraunhofer.de/eHealth/cts2infoModel/
       signatures#>.
2  @prefix psm: <http://fokus.fraunhofer.de/eHealth/
       smartMenuInfoModel/signatures#>.
3
4
5  select * {
6          ?recipe a psm:Recipe;
7                  :preferredTerm ?preferredTerm;
8                  :name ?name;
9                  psm:totalTime ?totalTime;
10                 psm:nutritionInfo ?nutrition.
11
12         ?nutrition
13                 psm:carbohydrates ?carbohydrates;
14                 psm:energy ?energy;
15                 psm:fat ?fat;
16                 psm:protein ?protein;
17                 psm:cholesterol ?cholesterol;
18                 psm:fiber ?fiber;
19                 psm:iron ?iron;
20                 psm:vitamin_d ?vitamin_d;
21                 psm:sodium ?sodium.
22 }
```

Diese Anfrage erzielt noch nicht den gewünschten Effekt, da alle Rezepte der Wissensbasis im Ergebnis enthalten wären. Um einen Ausschluss zu erreichen, werden dynamisch **FILTER** und **FILTER NOT EXISTS**-Ausdrücke am Ende der Query eingefügt. Diese ermöglichen die Definition von numerischen Restriktionen und stellen sicher, dass eingehende *excludesRecipe*-Relationen vermieden werden.

Beispiel: Für einen Slot soll die Menge der zulässigen Gerichte so eingeschränkt werden, dass jedes Gericht bei einer Laktoseintoleranz und einer vegetarischen Ernährungsweise verzehrt werden darf sowie mindestens 5 Gramm Protein und maximal 600 Kilokalorien enthält. Die diätischen Einschränkungen werden in der Anfrage durch *Uniform Resource Identifiers* (URIs) referenziert. Im Beispiel sind dies http://www.nlm.nih.gov/mesh14#oidM0012180 (Laktoseintoleranz) und http://www.nlm.nih.gov/mesh14#oidM0022577 (Vegetarische Diät). Eine Nährstoffgrenze wird in der Anfrage durch einen FILTER-Ausdruck abgebildet. Eine Allergie, Intoleranz oder Ernährungsweise wird in eine FILTER NOT EXISTS-Anweisung

eingebettet. Listing 4.2 zeigt die aus dem Beispiel resultierenden SPARQL-Fragmente.

Listing 4.2: Exemplarische FILTER- und FILTER NOT EXISTS-Anweisungen

```
1 FILTER ( ?energy <= 600.0 )
2 FILTER ( ?protein >= 5.0 )
3 FILTER NOT EXISTS { <http://www.nlm.nih.gov/mesh14#oidM0012180>
    psm:excludesRecipe  ?recipe }
4 FILTER NOT EXISTS { <http://www.nlm.nih.gov/mesh14#oidM0022577>
    psm:excludesRecipe  ?recipe }
```

Diese Anweisungen werden in die zuvor dargestellte Grundstruktur integriert. Eine Ausführung der vollständigen Query liefert für das Beispiel 115 Rezepte.

Für jeden Slot ist genau eine Abfrage notwendig. Da bei einer Filter-Operation alle zum Zeitpunkt der Anwendung gefundenen RDF-Tripel ausgewertet werden müssen, kann die Abfrage mit hohen Antwortzeiten verbunden sein.
Bevor der Optimierungsprozess beginnt, wird überprüft, ob von den Nutzern unzulässige Nährwertgrenzen definiert wurden. Für jede Mahlzeit und jeden Nährstoff wird sichergestellt, dass die geforderte minimale Menge nicht die geforderte maximale Menge überschreitet. Gleiches wird für jeden Nutzer und jeden Nährstoff auf Tages-Ebene kontrolliert.
Der evolutionäre Prozess wird durch Aufruf der Methode *run* in der Klasse **RunMenuPlanning** angestoßen. An diese wird die zu Beginn erstellte Instanz der Klasse **MenuPlan** übergeben. Mithilfe der Klassenvariablen *maximumTime* und *maxEvaluations* können die Ausführungsdauer bzw. die maximal zu tätigende Anzahl an Bewertungen festgelegt werden. Die Größe der Population, die Mutationsrate und die Crossover-Wahrscheinlichkeit werden durch die Variablen *populationSize*, *mutationRate* und *crossoverProbability* spezifiziert.
Bei der **Initialisierung** des Problems (genauer des *MenuPlanningProblems*) werden jeweils die Anzahl der Variablen (a), der Zielfunktionen (b) und der Constraints (c) angegeben. Dabei entspricht a der Anzahl der Slots, b ist stets drei und c ist gleich der Anzahl aller tagesbezogenen Restriktionen. Es wird eine initiale Population der Größe *populationSize* erstellt. Für jedes Individuum der Population werden a Instanzen der Klasse *SlotVariable* erstellt. Durch Setzen der Variable *meal* wird jeder Instanz zufällig ein Gericht aus der Menge erlaubter Gerichte zugeordnet. Im nächsten Schritt werden die Lösungen **bewertet**. Von Relevanz sind dafür die drei Zielfunktionen f_1, f_2 und f_3, welche im Abschnitt 3.2 formuliert und erläutert wurden. Da das Framework grundsätzlich auf die Minimierung von Zielfunktionen ausgelegt ist, werden die Werte der ersten und zweiten Zielfunktion negiert. Die Bewertung umfasst auch die Berechnung aller Constraint-Verletzungen. Anstatt die initiale Population wie in Abschnitt 3.3 beschrieben zu sortieren, wird die Dominanz-Relation zwischen zwei Lösungen im Rahmen der **Turnierselektion** bestimmt. Da die Selektion

ohne Zurücklegen stattfindet, ist eine Lösung stets nur ein einziges Mal Turnierkandidat. Wurden zwei Elternteile bestimmt, schließt sich das **1-Punkt-Crossover** an. Die daraus entstandenen Nachkommen werden der **Mutation** unterzogen, bewertet und anschließend der Kind-Population hinzugefügt.
Dieser Prozess aus Selektion, Rekombination, Mutation und Bewertung wird solange wiederholt, bis die Kind-Population *populationSize* Individuen enthält. Ist dies erreicht, werden die Eltern- und Kind-Populationen vereinigt und gemäß *Nondominated Sorting* in Pareto-Fronten (beginnend mit null) eingeordnet. Es schließt sich die **Umweltselektion** und die **Überprüfung der Terminierungsbedingung** an.
Nach Beendigung des genetischen Algorithmus wird die aktuelle Population abgefragt. Da nicht gewährleistet ist, dass alle Lösungen der Population zulässig sind, muss bei weiteren Verarbeitungsschritten überprüft werden, ob sie Summe aller Constraint-Verletzungen gleich null ist. Ebenfalls müssen die ursprünglichen Werte der ersten und zweiten Zielfunktion durch Negation wiederhergestellt werden.

4.3 Architektur

Es wird eine klassische Client-Server-Architektur - bestehend aus den logischen Subsystemen *Frontend* und *Backend* - umgesetzt. Die Server-Anwendung *WebSmartMenu* umfasst den Terminologiedienst CTS2-LE sowie die zur Menüplanung benötigten Komponenten. Über HTTP-Schnittstellen findet die Kommunikation mit einem Thin-Client statt. Dieser beherbergt lediglich das Frontend (*WebSmartMenuClient*) mit minimaler Anwendungslogik.

Backend

Die Server-Anwendung *WebSmartMenu* ermöglicht das komfortable Erforschen und Durchsuchen des Wissensnetzes durch den Terminologienavigator. Darüber hinaus kapseln HTTP-Schnittstellen den Zugriff auf die folgenden Komponenten:

- CTS2-LE Terminologiedienst
- Nutzerverwaltung
- Optimierte Menüplanung

Die folgenden Tabellen dokumentieren komponentenweise für jede Schnittstelle die **Funktionalität**, die **Methode**, den **Pfad** sowie **Parameter**. Beispiele für übermittelte Anfrage- und Antwort-Datensätze sind in Abschnitt A.8 im Anhang aufgeführt. Da die POST- und

Response-Inhalte für einige Schnittstellen der Menüplanung sehr umfangreich ausfallen, sind diese auf dem DVD-Anhang hinterlegt.
Für alle Schnittstellen gilt der Stamm-Pfad **/WebSmartMenu/rest**.

Tabelle 4.2: HTTP-Schnittstellen des CTS2-LE Terminologiedienstes

Funktionalität	Methode	Pfad	Parameter
Abfrage aller Intoleranzen	GET	/vocabularies/intolerances	-
Abfrage aller Allergien	GET	/vocabularies/allergies	-
Abfrage aller Ernährungsweisen	GET	/vocabularies/dietConcepts	-

Tabelle 4.3: HTTP-Schnittstellen der Nutzerverwaltung

Funktionalität	Methode	Pfad	Parameter
Abfrage aller Nutzer	GET	/users	-
Abfrage eines Nutzers	GET	/user/{userId}	Pfad-Parameter *userId*: ID des Nutzers
Erstellen eines Nutzers	POST	/user	Query-Parameter *userId*: ID des Nutzers
Aktualisieren eines Nutzers	PUT	/user/{userId}	Pfad-Parameter *userId*: ID des Nutzers
Löschen eines Nutzers	DELETE	/user/{userId}	Pfad-Parameter *userId*: ID des Nutzers
Löschen aller Nutzer	DELETE	/users	-

Tabelle 4.4: HTTP-Schnittstellen der optimierten Menüplanung

Funktionalität	Methode	Pfad	Parameter
Initiieren der Optimierung	POST	/menuplan/optimize	-
Abfrage aller Optimierungsergebnisse	GET	/menuplans	-
Abfrage eines Optimierungsergebnisses	GET	/menuplan/{planId}	Pfad-Parameter *planId*: ID des Optimierungsergebnisses
Speichern eines Optimierungsergebnisses	POST	/menuplan	Query-Parameter *planId*: ID des Optimierungsergebnisses
Löschen eines Optimierungsergebnisses	DELETE	/menuplan/{planId}	Pfad-Parameter *planId*: ID des Optimierungsergebnisses
Löschen aller Optimierungsergebnisse	DELETE	/menuplans	-

Frontend

Das Frontend wurde unter Verwendung des Javascript Frameworks *Angular*[8] entwickelt. Zur Gestaltung der Oberfläche wurden Komponenten des *Boostrap*-Frameworks[9] sowie *Cascading Style Sheets* (CSS) genutzt. Die Applikation gliedert sich in eine Nutzer- und Menüplanverwaltung. Erstere bedient die folgenden Anwendungsfälle:

- Abfrage aller Nutzer oder eines spezifischen Nutzers
- Hinzufügen eines neuen Nutzers
- Verändern eines Nutzers
- Löschen eines spezifischen Nutzers

Die Menüplanverwaltung ermöglicht das

- Anlegen eines Menüplans
- Initiieren eines Optimierungsprozesses
- Speichern eines Optimierungsergebnisses

[8] https://angular.io/, letzter Zugriff am 21.02.17
[9] http://getbootstrap.com/, letzter Zugriff am 21.02.17

- Abfragen aller Optimierungsergebnisse oder eines spezifischen Optimierungsergebnisses
- Löschen eines spezifischen Optimierungsergebnisses

Nachfolgend werden die Nutzung der Kern-Funktionalitäten sowie die Navigation durch das Frontend anhand von Screenshots geschildert. Die Adressen, unter welcher die Seiteninhalte erreicht werden, beginnen stets mit **http://{host}:{port}/**. Dies ist gleichzeitig auch der Eintrittspunkt, von welchem aus die Nutzer- und Menüplanverwaltung erreicht werden. Unter der URL http://{host}:{port}/#/users werden alle gespeicherten Nutzer aufgeführt. Abbildung 4.2 zeigt die Oberfläche beim Aufruf sowie nach Auswahl eines Nutzers.

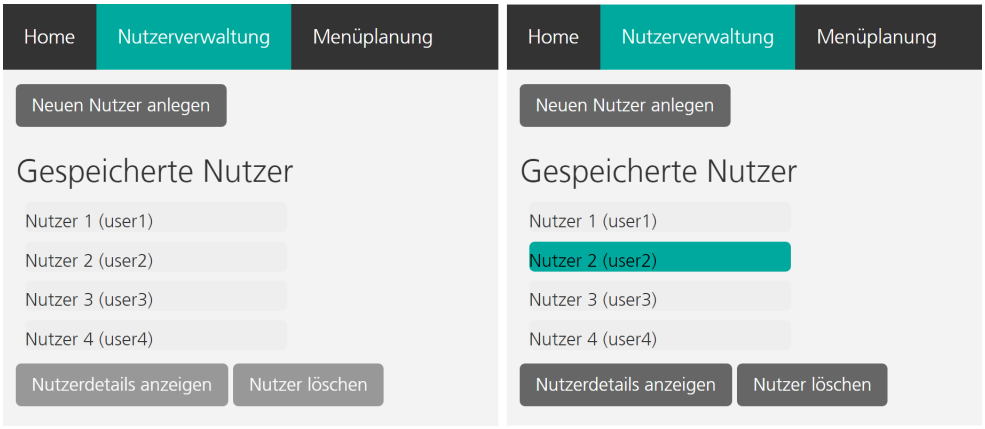


Abbildung 4.2: Seitenstruktur beim Aufruf von http://{host}:{port}/#/users, rechts mit Auswahl eines Nutzers

Detaillierte Angaben zu einem Nutzer können durch Anklicken des entsprechenden Feldes eingesehen werden. Die Oberfläche zeigt die ID, den Namen, die diätischen Einschränkungen sowie die Lebensmittelbewertungen eines Nutzers. Die Anordnung dieser Elemente wird exemplarisch für den Nutzer mit der ID *user2* in Abbildung 4.3 dargestellt. Bis auf den Identifikator sind alle Daten veränderbar. Allergien, Intoleranzen und Ernährungsweisen werden durch Anklicken der entsprechenden Felder aktiviert bzw. deaktiviert. Im unteren Bereich der Seite können Lebensmittel bewertet werden. Das Eingabefeld besitzt eine Autovervollständigungsfunktion, welche unter Zugriff auf CTS2-LE passende Einträge des BLS-Vokabulars vorschlägt. Die Auswahl eines Listeneintrags ist für die Herstellung einer eindeutigen Referenz auf Einträge des BLS unabdingbar. Erfolgt keine Auswahl, ist ein Speichern nicht möglich. Die Sterne symbolisieren eine ordinale Bewertungsskala, wobei eine starke Abneigung gegen ein Lebensmittel durch einen Stern und eine starke Präferenz für ein Lebensmittel durch fünf Sterne ausgedrückt werden. Die Bewertung kann durch Markierung des Kontrollkästchens auch auf Kind-Konzepte ausgeweitet werden. Dabei ist zu beachten, dass nicht nur direkt untergeordnete Konzepte, sondern alle Lebensmittel, welche

hierarchisch unterhalb des ausgewählten Konzepts liegen, eingeschlossen werden.

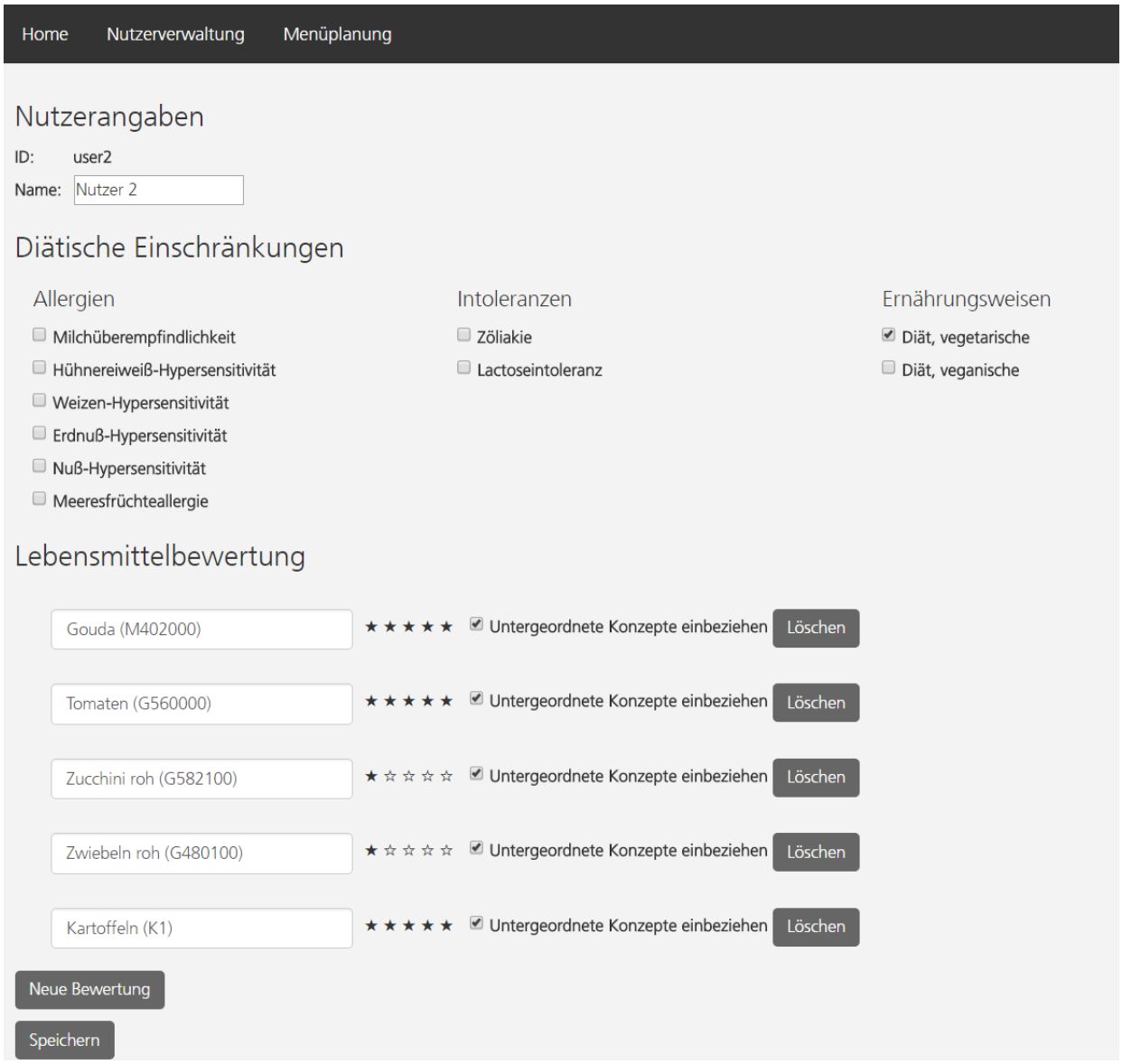


Abbildung 4.3: Seitenstruktur beim Aufruf von http://{host}:{port}/#/user-detail/user2

Die gleiche Seitenstruktur wird auch beim Erstellen eines Nutzers präsentiert. Dann ist das Festlegen einer ID erforderlich. Nach dem Aktualisieren oder Hinzufügen eines Nutzers gibt eine Meldung am oberen Rand Auskunft über den Erfolg einer getätigten Aktion. Im Falle eines Misserfolgs dienen angezeigte Hinweise der Fehlerbehebung.

Die **Menüplanverwaltung** wird durch Anklicken des in der Navigationsleiste aufgeführten Punktes *Menüplanung* aufgerufen. Die Oberfläche (dargestellt in Abbildung 4.4) listet gespeicherte Optimierungsergebnisse, welche nach Auswahl angezeigt oder gelöscht werden

können.

Abbildung 4.4: Seitenstruktur beim Aufruf von http://{host}:{port}/#/menuplans, rechts mit Auswahl eines gespeicherten Optimierungsergebnisses

Eine neue Menüplanung wird durch Anklicken des entsprechenden Bedienelementes initiiert. Die daraufhin geladene Oberfläche erlaubt den Aufbau beliebiger Tabellenstrukturen durch Verwendung der Schaltflächen *Tag hinzufügen*, *Mahlzeit hinzufügen*, *Tag entfernen* und *Mahlzeit entfernen*. Alle derzeitig gespeicherten Nutzer sind rechts von der Tabelle gelistet. Die Zuordnung eines Nutzers zu einem Tag kann nach Auswahl eines Nutzers und eines Tages vorgenommen werden. Die selektierten Elemente werden farblich hervorgehoben. Abbildung 4.5 zeigt die Spezifikation eines Menüplans für drei Tage.

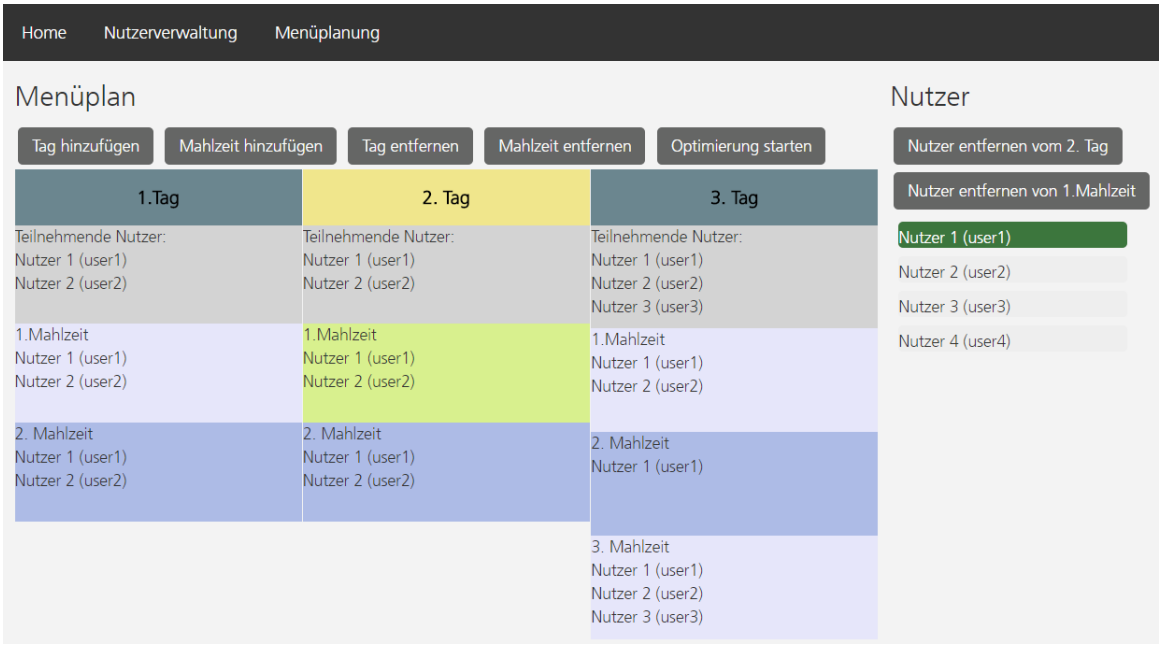


Abbildung 4.5: Oberfläche zum Anlegen einer Menüplanspezifikation

Für die ersten zwei Tage sind jeweils zwei Mahlzeiten und für den dritten Tag sind drei Mahlzeiten zu belegen. Jeder Nutzer, welcher an einer Mahlzeit eines Tages teilnimmt, wird in der ersten Reihe in einer grau gefärbten Zelle aufgeführt. Im Beispiel nehmen an allen

Tagen die Nutzer 1 und 2 teil. Für den dritten Tag wird zusätzlich Nutzer 3 eingeplant. Wird ein Nutzer innerhalb dieses Bereichs angeklickt, öffnet sich ein Fenster, welches ein Formular zur Spezifizierung tagesbezogener Nährwertgrenzen umfasst (siehe Abbildung 4.6 nach Auswahl des Nutzers 2 am 2.Tag).

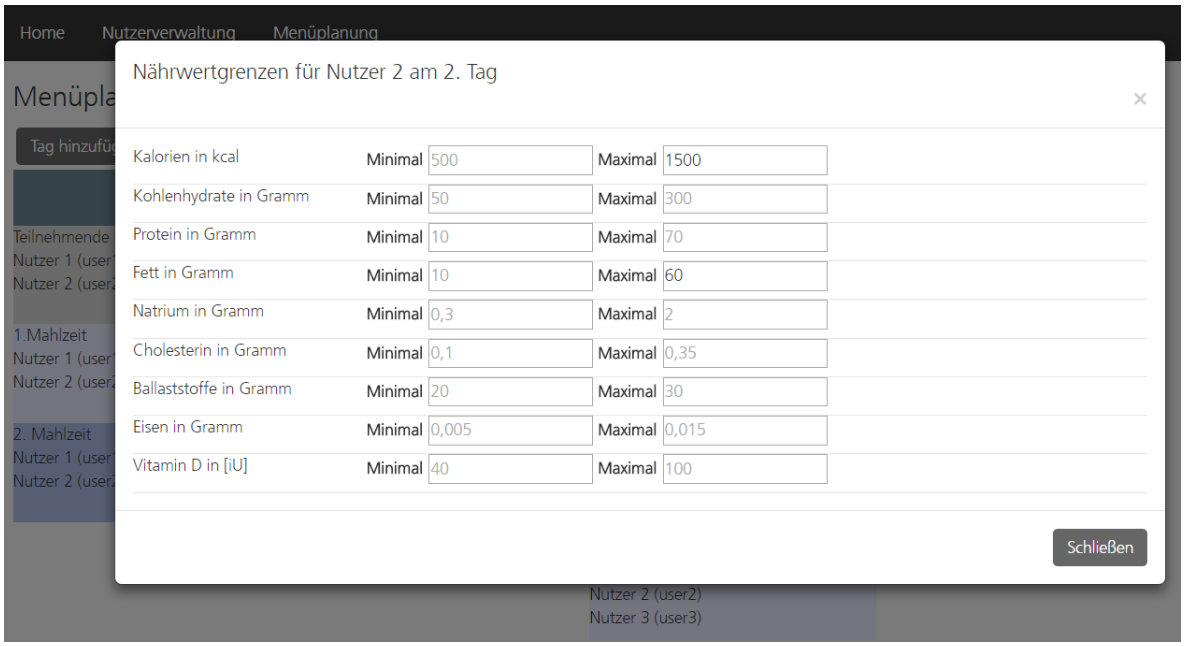


Abbildung 4.6: Fenster zur Angabe tagesbezogener Nährwertgrenzen

Die Benutzerfreundlichkeit wird durch grau kolorierte Platzhalter-Werte erhöht. Eingegebene Werte (im Beispiel eine Kalorienobergrenze von 1500 kcal und eine maximale Menge von 60 g Fett) sind verbindlich, d.h. sie müssen nicht durch ein Bedienelement bestätigt werden und werden nicht durch Schließen des Fensters über das Kreuz in der oberen rechten Ecke zurückgesetzt. Unzulässige Nährwertgrenzen werden farblich hervorgehoben, ansonsten jedoch toleriert. Beispielsweise führt die Eingabe eines Mindest- und Höchstwertes, wobei der Höchstwert kleiner als der Mindestwert ist, nicht dazu, dass das Verfahren nicht gestartet werden kann. Im Backend findet vor der Initiierung der Optimierung eine Überprüfung statt, welche ggf. zum Abbruch und der Präsentation einer entsprechenden Fehlermeldung führt. Ein ausgewählter Nutzer (dunkelgrün eingefärbt) kann nach Markierung einer Mahlzeiten-Zelle (hellgrün eingefärbt) dieser Mahlzeit hinzugefügt oder von dieser entfernt werden. Zur Festlegung von mahlzeiten- und nutzerspezifischen Nährwertgrenzen und maximalen Zubereitungszeiten wird ein Nutzer innerhalb der Mahlzeiten-Zelle angeklickt. Daraufhin öffnet sich ein Fenster, welches die Dateneingabe (ähnlich wie in Abbildung 4.6) ermöglicht. Auch an dieser Stelle wird eine invalide Eingabe in der Oberfläche hervorgehoben und von der Server-Anwendung behandelt.

Ist die Konfiguration des Plans abgeschlossen und ist für jede Mahlzeit mindestens ein Nutzer als Teilnehmer angegeben, kann das Optimierungsverfahren gestartet werden. Zuvor öffnet sich ein Dialog zur Parametrisierung des Verfahrens (siehe Abbildung 4.7). Innerhalb dieses Dialogs können die Abbruchbedingung, die Populationsgröße, die Mutationsrate sowie die Crossover-Wahrscheinlichkeit festgelegt werden. Bei fehlender Eingabe werden Standardwerte[10] verwendet.

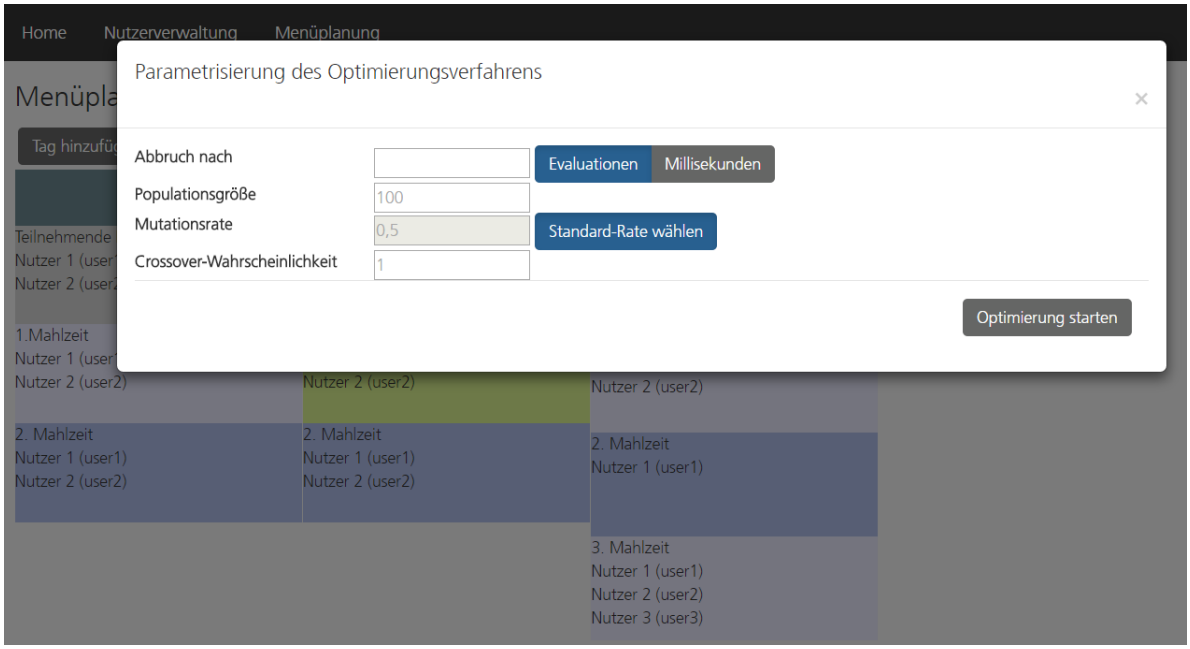


Abbildung 4.7: Dialog zur Angabe der Parameter des Optimierungsverfahrens

Nach Beendigung des Verfahrens findet eine Weiterleitung zur Darstellung des Optimierungsergebnisses (siehe Abbildung 4.8) statt. Die Bedienelemente oberhalb der Tabellenstruktur ermöglichen eine Navigation durch alle zulässigen Individuen der Ergebnis-Population. Im Beispiel sind alle Lösungen zulässig und gehören zur 0.Front. Nach Anklicken eines zugeordneten Gerichts öffnet sich ein Fenster, welches die Nährwertinformationen präsentiert und einen Seitenverweis auf die Konzept-Beschreibung des Terminologieservers bereitstellt.

[10] die Parameter sind standardmäßig wie folgt festgelegt: Terminierung nach 10.000 Evaluationen, Populationsgröße = 100, Mutationsrate = 1/Anzahl der Variablen, Crossover-Wahrscheinlichkeit = 1

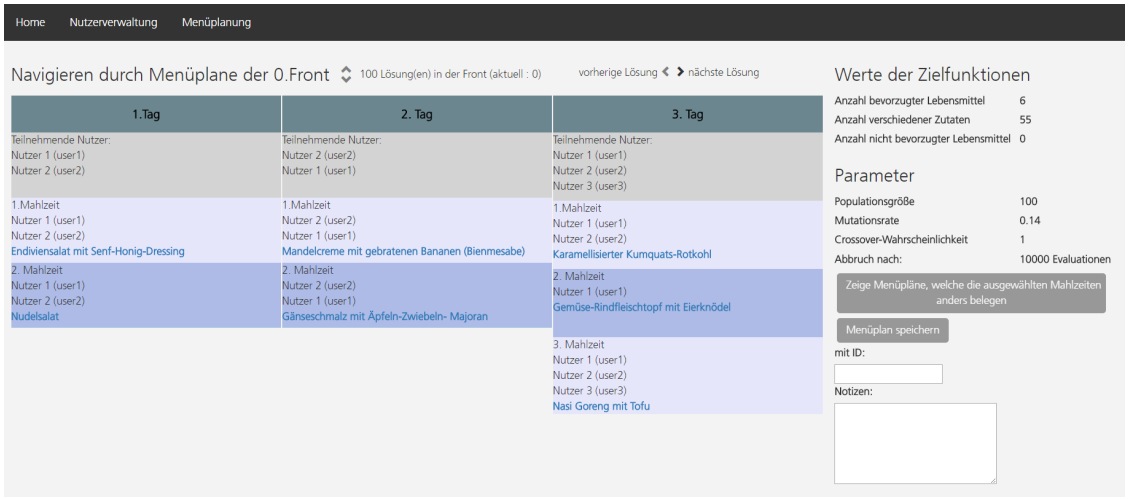


Abbildung 4.8: Darstellung des Optimierungsergebnisses

Die jeweiligen Zielfunktionswerte und die Parametrisierung des Verfahrens sind rechts neben der Tabelle zusammengefasst. Beispielsweise enthalten die zugeordneten Gerichte der aktuellen Lösung insgesamt 6 bevorzugte, 55 verschiedene und 0 nicht bevorzugte Lebensmittel.[11] Das Ergebnis kann nach Angabe eines Identifikators und einer Notiz (optional) gespeichert werden.

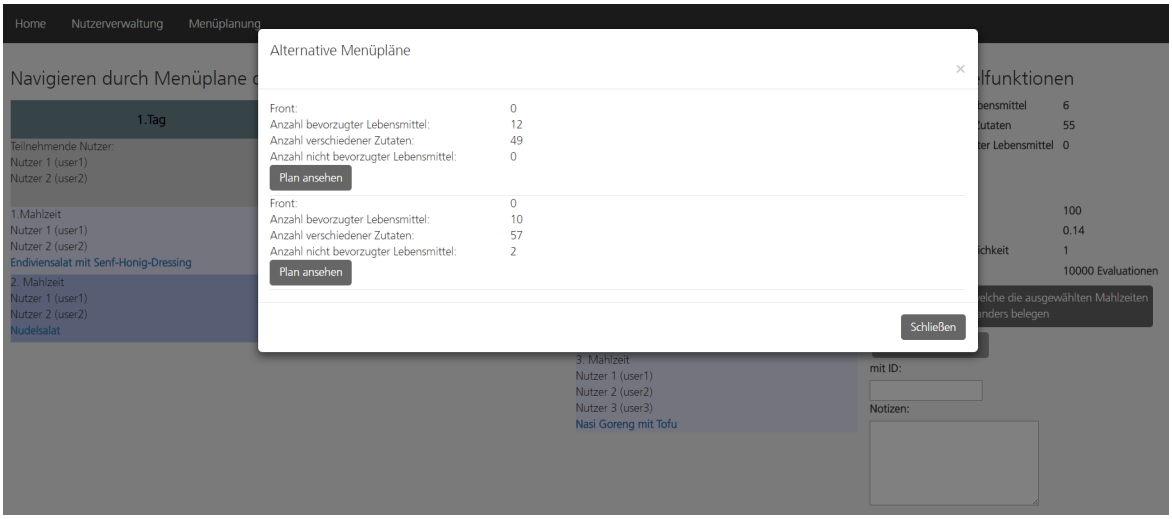


Abbildung 4.9: Präsentation alternativer Menüpläne

Weiterhin ist es möglich, nach ähnlichen Plänen innerhalb der Ergebnis-Population zu suchen. Im aktuell betrachteten Menüplan werden dafür diejenigen Mahlzeiten ausgewählt, welche anders belegt werden sollen. Dies geschieht durch Anklicken der entsprechenden

[11] besonders bevorzugte und nicht bevorzugte Lebensmittel werden entsprechend der definierten Zielfunktionen doppelt gezählt

Zellen-Überschriften.[12] Anschließend wird die Schaltfläche rechts neben der Tabelle betätigt und ein Fenster zeigt alternative Menüpläne mit ihren Fronten-Zugehörigkeiten und Zielfunktionswerten. Möchte ein Nutzer beispielsweise die ersten Mahlzeiten des zweiten und dritten Tages austauschen -die restliche Belegung aber beibehalten-, werden ihm die in Abbildung 4.9 gezeigten alternativen Menüpläne vorgeschlagen. Zur Auswahl stehen zwei unterschiedlich bewertete Menüpläne, welche ebenfalls zur 0.Front gehören. Nach Auswahl eines Menüplans wird dieser in der Oberfläche angezeigt.
Der Quellcode des Frontends ist vollständig auf der DVD im Ordner *WebSmartMenuClient* hinterlegt.

4.4 Evaluation der Optimierung

In diesem Abschnitt wird die Optimierungskomponente der entwickelten Web-Applikation evaluiert. Anhand eines einfachen und eines komplexen Beispiels (nachfolgend *Testszenarien* genannt) soll die Entwicklung der Population während der Optimierung untersucht werden. Von Interesse sind dabei die Veränderung der Zielfunktionswerte und die Veränderung der Anzahl zulässiger Lösungen bei gleichbleibender Parametrisierung. Nachfolgend werden die Testbeispiele und die gemessenen Werte skizziert. Anschließend werden die Auswertungen präsentiert.

Beschreibung der Testszenarien

An den Testszenarien sind die in Tabelle 4.5 aufgeführten Nutzer beteiligt. Für jeden Nutzer sind der Name, der Identifikator, diätische Einschränkungen und die Anzahl der Lebensmittelbewertungen (Kind-Konzepte eingeschlossen) gegeben.

[12] ein erneutes Anklicken hebt die Auswahl auf

Tabelle 4.5: Rahmendaten der Testnutzer

Name	Identifikator	Diätische Einschränkungen	Anzahl der Lebensmittel-bewertungen
Nutzer 1	user1	Lactoseintoleranz	180
Nutzer 2	user2	Vegetarische Diät	85
Nutzer 3	user3	-	63
Nutzer 4	user4	Meeresfrüchteallergie, Zöliakie	80
Nutzer 5	user5	Nuß-Hypersensitivität	49

Menüplanspezifikation des ersten Testszenarios

Zu planen sind insgesamt 10 Mahlzeiten, welche auf drei Tage verteilt werden. Am ersten und zweiten Tag müssen jeweils drei Zuordnungen, am dritten Tag vier Zuordnungen gefunden werden. Die Teilnahme der Nutzer 1, 2 und 4 wird in Abbildung 4.10 visualisiert.

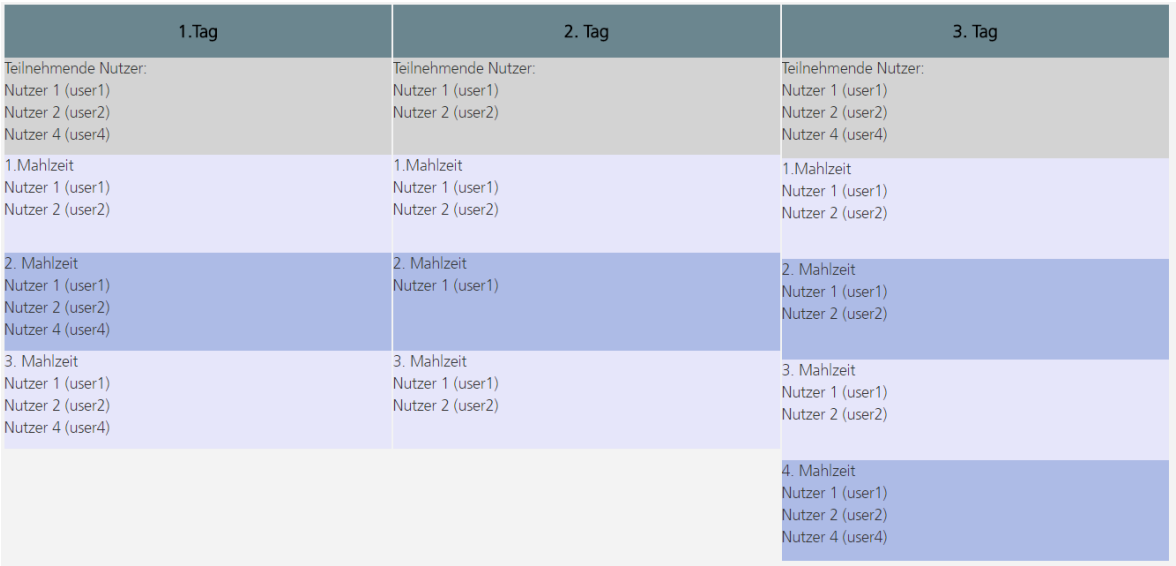


Abbildung 4.10: Menüplanstruktur und teilnehmende Nutzer des ersten Testszenarios

Es werden fünf tages- und fünf mahlzeitenbezogene Restriktionen festgelegt. Letztere führen zu einer Reduktion der zuordenbaren Gerichte. Die Anzahl der pro Mahlzeit verbleibenden Gerichte ist in Tabelle 4.6 zusammengefasst. Es wird die in Abschnitt 3.3 eingeführte Notation für Parameter $p_{t,m}$ und deren Wertebereiche $G_{p_{t,m}}$ verwendet.

Tabelle 4.6: Parameter des ersten Testszenarios und Anzahl der möglichen Zuordnungen

t, m	$\lvert G_{p_{t,m}} \rvert$	t, m	$\lvert G_{p_{t,m}} \rvert$	t, m	$\lvert G_{p_{t,m}} \rvert$	t, m	$\lvert G_{p_{t,m}} \rvert$	t, m	$\lvert G_{p_{t,m}} \rvert$
0,0	332	0,2	111	1,1	686	2,0	32	2,2	332
0,1	117	1,0	45	1,2	80	2,1	332	2,3	117

Die Optimierung muss in diesem Beispiel fünf Constraints berücksichtigen. Die vollständige Menüplanspezifikation ist im JSON-Format auf der DVD in der Datei *menuplanspecification_scenario_1.json* hinterlegt.[13]

Menüplanspezifikation des zweiten Testszenarios

Die Komplexität einer Menüplanung wird nur marginal von der Anzahl der insgesamt zu belegenden Mahlzeiten bestimmt. Das Finden zulässiger, guter Lösungen wird vielmehr durch die Definition tagesbezogener Restriktionen erschwert. Bleiben diese unberücksichtigt, entstehen konfliktäre Tages-Kombinationen von Gerichten. Daraus resultierende unzulässige Lösungen müssen während der Optimierung alleinig durch die Anwendung genetischer Operationen in zulässige Lösungen überführt werden. Die Frage, ob die Operatoren eine reparierende und - im Sinne der Zielfunktionen - verbessernde Wirkung besitzen, soll durch das zweite Testszenario beantwortet werden.
In diesem sind 30 Mahlzeiten, welche über sieben Tage verteilt sind, zu belegen. Die Struktur des Plans und die Teilnahme der Nutzer ist Abbildung 4.11 zu entnehmen.

[13] Diese Datenstruktur ist gleichzeitig der zu übermittelnde POST-Body bei der Initiierung der Optimierung durch die HTTP-Schnittstelle. Für nähere Erläuterungen des Aufbaus und der Bedeutung der Elemente sei auf Abschnitt A.8.3 im Anhang verwiesen.

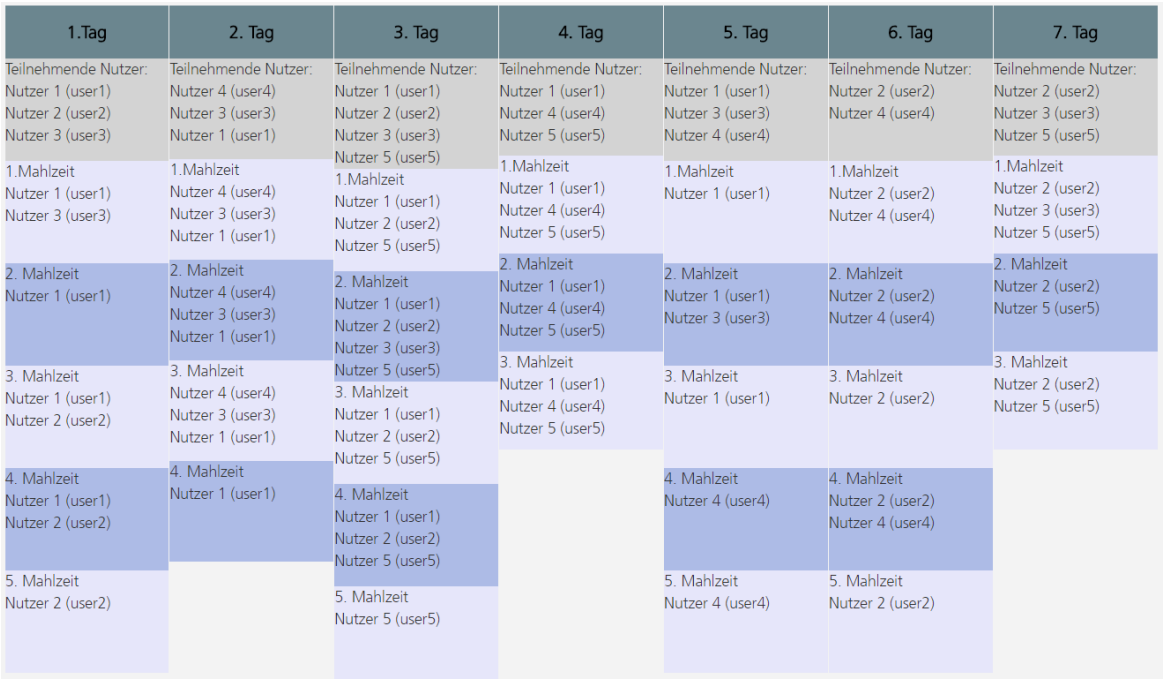


Abbildung 4.11: Menüplanstruktur und teilnehmende Nutzer des zweiten Testszenarios

Für jeden Tag werden fünf Restriktionen und für jede Mahlzeit wird zusätzlich eine Restriktion formuliert. Durch die Vorverarbeitung werden die Wertebereiche auf die in Tabelle 4.7 gelisteten Mengen eingeschränkt. Es verbleiben 35 zu erfüllende Constraints. Die auf dem DVD Anhang hinterlegte Datei *menuplanspecification_scenario_2.json* enthält die vollständige Spezifikation des Menüplans.

Tabelle 4.7: Parameter des zweiten Testszenarios und Anzahl der möglichen Zuordnungen

t, m	$\lvert G_{p_{t,m}} \rvert$	t, m	$\lvert G_{p_{t,m}} \rvert$	t, m	$\lvert G_{p_{t,m}} \rvert$	t, m	$\lvert G_{p_{t,m}} \rvert$	t, m	$\lvert G_{p_{t,m}} \rvert$
0,0	59	1,1	170	2,3	70	4,1	590	5,2	334
0,1	36	1,2	248	2,4	886	4,2	165	5,3	522
0,2	75	1,3	319	3,0	82	4,3	685	5,4	166
0,3	223	2,0	24	3,1	95	4,4	1043	6,0	307
0,4	53	2,1	136	3,2	98	5,0	87	6,1	1154
1,0	36	2,2	190	4,0	113	5,1	205	6,2	160

Gemessene Größen

Die zu messenden Größen sind

a die durchschnittlichen Zielfunktionswerte des besten Individuums

b die Anzahl der Generationen, nach welchen die Population im Durchschnitt die erste zulässige Lösung enthält

c die Anzahl der Generationen, nach welchen die Population im Durchschnitt nur noch zulässige Lösungen enthält

Zur Ermittlung dieser Werte wird das Optimierungsverfahren mit den präsentierten Testszenarien jeweils 100-malig mit der Standard-Parametrisierung ausgeführt. Untersucht wird die Entwicklung des besten Individuums in einer Population der Größe 100 über 500 Generationen.

Auswertung der Tests

Abbildung 4.12 zeigt die durchschnittlichen Zielfunktionswerte des besten Individuums über einen Verlauf von 500 Generationen bei der Ausführung des ersten Testszenarios. Alle Werte entwickeln sich wie beabsichtigt (Maximierung von (1) und (2), Minimierung von (3)), jedoch mit unterschiedlicher Intensität. Die Anzahl verschiedener Zutaten erhöht sich von anfänglich 41 auf 60 und die Anzahl der bevorzugten Lebensmittel steigt über den gesamten Verlauf von 8 auf 20. Demgegenüber steht die Minimierung der Anzahl nicht bevorzugter Lebensmittel von 9 auf 6.

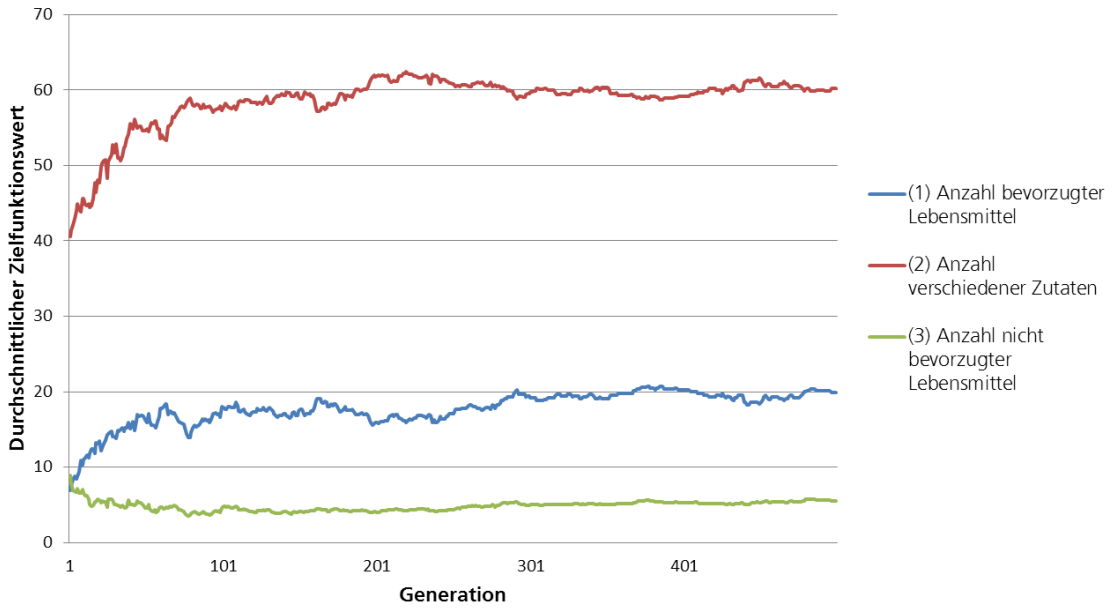


Abbildung 4.12: Verlauf der durchschnittlichen Zielfunktionswerte des besten Individuums bei der Optimierung des ersten Testszenarios

Dem Diagramm ist zu entnehmen, dass die durchschnittlichen Werte der Zielfunktionen (1) und (2) zu Beginn steil (für (1) ca. bis zur 60. und für (2) ca. bis zur 80.Generation) und dann

zunehmend flacher steigen. Stellenweise führen Wert-Schwankungen zu vorübergehenden Verschlechterungen, zum Beispiel fällt der Wert der zweiten Zielfunktion zwischen der 150. und 160.Generation leicht ab. Im Falle der dritten Zielfunktion wird innerhalb der ersten 80 Generationen ein leichter Abfall, gefolgt von einem nahezu konstanten Verlauf beobachtet. Die Abbildung zeigt einen leichten Zusammenhang zwischen den Werten der ersten und zweiten Zielfunktion: sinkt die Anzahl der verschiedenen Zutaten, steigt die Anzahl bevorzugter Zutaten und umgekehrt. Dies ist beispielsweise zwischen der 60. und 75. sowie zwischen der 160. und 290.Generation festzustellen. Diese Beobachtung führt zu der Vermutung, dass positiv bewertete Lebensmittel zugunsten einer gesteigerten Vielfalt durch abwechslungsreichere Menükomponenten ersetzt werden. Die Anzahl der nicht bevorzugten Lebensmittel bleibt davon unberücksichtigt.

Die Untersuchung der zulässigen Individuen zeigt, dass bereits nach einer Iteration eine Lösung gefunden wird, welche keine Restriktion verletzt. Im Durchschnitt enthält die Population in der achten Generation nur noch zulässige Lösungen. Diese zügige Verdrängung unzulässiger Lösungen ist nicht überraschend, da der umgesetzte genetische Algorithmus dem Hervorbringen zulässiger Lösungen eine höhere Priorität beimisst als der Optimierung. Dies bedeutet allerdings auch, dass das Genmaterial guter unzulässiger Individuen verloren geht. Das Erkunden vielversprechender und zulässiger Bereiche des Lösungsraums ist damit nicht möglich.

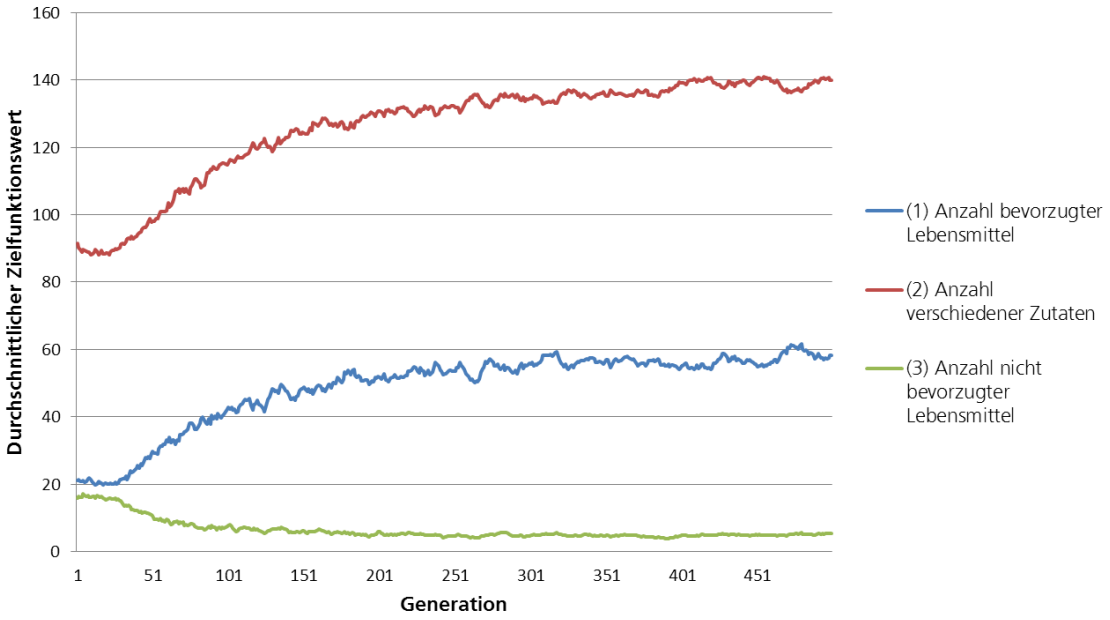


Abbildung 4.13: Verlauf der durchschnittlichen Zielfunktionswerte des besten Individuums bei der Optimierung des zweiten Testszenarios

Abbildung 4.13 zeigt die Ergebnisse des zweiten Testszenarios. Über den gesamten Verlauf von 500 Generationen verbessern sich auch in diesem komplexen Beispiel die durchschnittlichen Werte der drei Zielfunktionen. Die Anzahl der bevorzugten Lebensmittel kann durch den imitierten evolutionären Prozess von 21 auf 57 gesteigert werden. Ebenso wird die Vielfalt (ausgedrückt durch Zielfunktion (2)) deutlich verbessert. Umfasst der beste Plan der initialen Population noch 90 verschiedene Zutaten, so kann dieser Wert auf ca. 140 in der letzten Generation erhöht werden. Die Wechselwirkung zwischen der ersten und zweiten Zielfunktion wird nun anders charakterisiert. Das im ersten Beispiel vermutete konfliktäre Verhalten der Zielfunktionen wird an dieser Stelle widerlegt. Bis auf wenige Generationen ist ein gleichmäßiger Anstieg beider Fitnesswerte wahrnehmbar. Weiterhin zeigt das Diagramm eine Reduktion der Anzahl nicht bevorzugter Lebensmittel von 16 auf 5. Im Gegensatz zum ersten Szenario setzt eine Verbesserung aller durchschnittlichen Zielfunktionswerte des besten Individuums nicht sofort ein. Bis zur ca. 20.Generation verringern sich die Werte der ersten und zweiten Zielfunktion minimal, die Anzahl nicht bevorzugter Lebensmittel bleibt annähernd konstant. Dies ist dadurch zu erklären, dass im Durchschnitt die erste zulässige Lösung in der Population der 31.Generation enthalten ist. Der Selektionsprozess bevorzugt bis zu diesem Zeitpunkt stets das Individuum mit der geringsten gesamten Constraint-Verletzung. Die durch die drei Funktionen gemessene Fitness ist dabei nicht relevant. Dieser Selektionsdruck setzt erst dann ein, wenn zwei zulässige Lösungen miteinander konkurrieren. Im Durchschnitt umfasst die Population der 63.Generation ausschließlich zulässige Lösungen. Im Vergleich zum ersten Szenario mit lediglich fünf Constraints wird deutlich, dass die Suche mit steigender Constraint-Anzahl erschwert wird. Da keine Reparatur unzulässiger Lösungskandidaten stattfindet, muss darauf gehofft werden, dass durch die von Zufallsereignissen begleiteten Operatoren eine zulässige Lösung konstruiert wird. Ist dies geglückt, tritt die Exploration des Suchraums zunehmend in den Hintergrund. Die Suche konzentriert sich dann auf den zulässigen Bereich. Die Selektion sorgt dafür, dass das Genmaterial von Individuen, welche die Restriktionen nicht erfüllen, rasch verloren geht. Für das vorliegende Menüplanungsproblem bedeutet das, dass die Diversität der Population abnimmt und sich die Lösungskandidaten im Verlauf der Evolution immer mehr ähneln. Das Erreichen eines globalen Optimums wird damit in hohem Maße erschwert.

5 Zusammenfassung

Im Rahmen dieser Arbeit wurde eine Web-Applikation entwickelt, welche eine optimierte Menüplanung vornimmt. Zu Beginn der Arbeit erfolgte eine Festlegung und Klassifikation zu erfüllender Anforderungen. Von diesen wurden einige aufgrund unzureichender oder gänzlich fehlender Daten im späteren Bearbeitungsverlauf verworfen, sodass die Menüplanung letztlich

- beliebige Planstrukturen unterstützt,
- diätische Einschränkungen (Lebensmittelallergien, -intoleranzen und Ernährungsweisen), Nährstoffunter und -obergrenzen und Zubereitungsdauern berücksichtigt und
- eine Optimierung der Planvielfalt sowie individueller Lebensmittelvorlieben und -abneigungen anstrebt.

Das Menüplanungsproblem wurde formal beschrieben und als multikriterielles Optimierungsproblem mit Randbedingungen eingeordnet. Dieses wird mithilfe des genetischen Algorithmus NSGA-II gelöst. Die Anwendung des evolutionären Verfahrens erforderte das Festlegen einer geeigneten Individuen-Kodierung und darauf aufbauender genetischer Operatoren. Gewählt wurde die Repräsentation eines Menüplans als Parameter-Array. Nachkommen werden durch Anwendung des 1-Punkt-Crossovers und der Integer-Mutation erzeugt. Lösungskandidaten, welche die Restriktionen missachten, gelten als unzulässig. Das Ausmaß der Unzulässigkeit wird dabei durch die Constraint-Verletzung quantifiziert. Die nachempfundene Evolution ist durch die folgenden Eigenschaften charakterisiert:

- Optimierung der Werte dreier Zielfunktionen
- Nondominated Sorting mit Schlechterstellung unzulässiger Individuen
- binäre Turnierselektion mit Crowded-Comparison-Operator zur Auswahl der Eltern
- Umweltselektion auf Basis der Fronten-Zugehörigkeit und ggf. der Crowding Distance
- Elite-Strategie zur Übernahme der besten Individuen

Das zur Optimierung benötigte Wissen über Zusammenhänge zwischen Gerichten, Lebensmitteln, Ernährungs- und Stoffwechselkrankheiten sowie Ernährungsweisen wird in einem

Netz gespeichert. Zur Verwaltung wird der auf semantischen Technologien fußende Terminologiedienst CTS2-LE benutzt. Unter Verwendung von RDFS und eines dienstspezifischen Schema-Vokabulars wurde ein Informationsmodell entwickelt, welches die abzubildenden Entitäten und Beziehungen formal darstellt. Das Faktenwissen stammt überwiegend aus standardisierten Begriffssystemen wie dem MeSH, UCUM oder dem Bundeslebensmittelschlüssel. Ferner wurde eine umfangreiche Rezeptdatenbank durch Abfrage der REST-Schnittstelle der Yummly-Plattform integriert. Lebensmittelproduktdaten konnten nicht in die Wissensbasis aufgenommen werden, da für diese keine frei zugänglichen Datenquellen gefunden wurden. Die einzelnen Vokabularien wurden auf unterschiedliche Art und Weise miteinander vernetzt. Um die Verwendung eines Lebensmittels als Zutat in einem Rezept zu disambiguieren, wurde ein pragmatisches Matching von Zutatenbezeichnern auf eindeutige Lebensmitteleinträge des BLS umgesetzt. Ausschlussbeziehungen zwischen diätischen Einschränkungen und Lebensmitteln wurden manuell definiert und bildeten die Grundlage für eine regelbasierte Ableitung nicht zu verzehrender Gerichte. Die Wissensbasis wird im Rahmen der Optimierung genutzt, um die Menge der pro Mahlzeit zuordenbaren Gerichte einzuschränken.
Es wurden vier Java-Bibliotheken für die Implementierung des evolutionären Algorithmus in Betracht gezogen. Aufgrund der Aktualität und der hohen Qualität der Dokumentation wurde das MOEA-Framework ausgewählt. Mit diesem wurde der konzipierte Lösungsansatz mitsamt der problemangepassten Komponenten (Individuen-Kodierung, genetische Operatoren) auf Code-Ebene umgesetzt. Die Web-Applikation setzt sich aus einer Server- und einer Client-Anwendung zusammen. Erstere beinhaltet die Anwendungslogik der Nutzerverwaltung (mit den Operationen Hinzufügen, Lesen, Aktualisieren und Löschen), der Planverwaltung (mit den Operationen Hinzufügen, Lesen und Löschen) und der Menüplanung. Gleichzeitig koordiniert die Server-Anwendung den Zugriff auf den Terminologiedienst. Die Client-Anwendung stellt die Bedienoberfläche bereit und basiert technologisch auf Javascript, HTML und CSS. Sie beherbergt lediglich die Logik zur Bereitstellung alternativer Menüpläne innerhalb einer Ergebnis-Population. Backend und Frontend sind lose über HTTP-Schnittstellen gekoppelt. Die Ergebnisse der Optimierung wurden anhand zweier Beispielszenarien ausgewertet. Es wurde gezeigt, dass auch für komplexe Probleminstanzen mit vielen Restriktionen zulässige Lösungen gefunden und hinsichtlich der Fitnesswerte iterativ verbessert werden konnten. Dabei wirkt sich die Dominanz weniger zulässiger Individuen negativ auf die Diversität der Population aus.

5.1 Diskussion

Im Fokus dieser Arbeit stand die Beantwortung der zu Beginn gestellten zweiteiligen Forschungsfrage. Unter diesem Gesichtspunkt werden nachfolgend die Ergebnisse der Arbeit reflektiert.

Zunächst wird Bezug auf den zweiten Teil der Forschungsfrage genommen und der Aufbau und die Qualität des für die Optimierung genutzten Wissensnetzes werden diskutiert. Für den konzipierten Einsatz innerhalb der Menüplanung sind Umfang und Qualität der Wissensbasis als akzeptabel zu bewerten. Durch die Wiederverwendung standardisierter, eindeutiger Vokabularien wird ein terminologischer Wildwuchs vermieden. Gleichzeitig können verlässliche, verifizierte Strukturen, wie zum Beispiel die Hierarchie des BLS, konstruktiv in die Anwendung eingebettet werden. Demgegenüber stehen die nicht geprüften Rezeptdaten der Yummly Plattform sowie die manuell hinzugefügten und regelbasiert abgeleiteten Zusammenhänge zwischen diätischen Einschränkungen und nicht zu verzehrenden Lebensmitteln. Dieses erschlossene Wissen ist unabdingbar für die Umsetzung der Menüplanung; eine Korrektheit kann jedoch nicht garantiert werden. Gleiches gilt für das Matching von Zutatenbezeichnern auf BLS-Einträge sowie extrahierte Mengenangaben und Einheiten. Dieser Umstand führte dazu, dass die vom BLS bereitgestellten Nährwertangaben nicht zur Ableitung der Nährwerte eines Rezepts genutzt werden konnten. Stattdessen wurden die Daten der Yummly-Rezept-Datenbank verwendet. Grundsätzlich wäre auch eine Generierung von Rezepten auf der Basis des BLS vorstellbar gewesen. Der entscheidende Nachteil dieses Vorgehens liegt darin, dass die Arbeit mit einer künstlichen Datensammlung wenig Akzeptanz genießt.

Das integrierte Faktenwissen zu Allergien, Intoleranzen und Ernährungsweisen deckt nur einen geringen Bereich der im Alltag relevanten Einschränkungen ab. Fehlend sind Ernährungstrends (bspw. Trennkost) und weitere Diätformen (z.B. Paleo- oder Atkins-Diät). Aus Anwendersicht ist zu bemängeln, dass Rezepte nicht in typische Kategorien wie Frühstücke, Mittag-, Abendessen, Snacks, etc. eingeteilt sind. Der Konsum von Getränken, welcher oftmals einen nicht zu vernachlässigenden Einfluss auf die tägliche Nährstoffzufuhr hat, bleibt ebenfalls unberücksichtigt.

Das Informationsmodell, welches die Struktur des Wissensnetzes formal beschreibt, spiegelt die erforderlichen Entitäten und Relationen adäquat wider. Rückblickend sind die Modellierung der Zutaten und der Ausschlussbeziehungen zu überdenken. Erstere ist zu spezifisch, da eine Aufsplittung in Mengenangaben und Einheiten überflüssig wurde. Die Ausschlussbeziehungen werden im Zuge der Menüplanung innerhalb von SPARQL-Filter-Ausdrücken aufgegriffen. Da die Filterung eine äußerst zeitaufwendige Operation ist, sollte sie so sparsam wie möglich eingesetzt werden. Durch Einführung einer zusätzlichen Relation

zur Definition erlaubter Gerichte kann die SPARQL-Abfrage so umstrukturiert werden, dass Filterungen nur noch zur Abbildung von Nährwertgrenzen angewandt werden müssen. Im Rahmen der Optimierung konnte das Wissensnetz gewinnbringend eingesetzt werden. Die Isolierung des Problemwissens führte dazu, dass bestimmte Zusammenhänge nicht während der Optimierung berücksichtigt werden mussten. So konnte die Erfüllung mahlzeitenbezogener Restriktionen vorab durch die Filterung gewährleistet werden. Alternativ hätte der genetische Algorithmus diese zusätzlich zu den tagesbezogenen Restriktionen behandeln müssen. Dies hätte das Finden zulässiger Lösungen erschwert.

Der umgesetzte genetische Algorithmus eignete sich zur Lösung des formulierten Menüplanungsproblems. Kritisch zu betrachten sind die Auswirkungen der Selektion auf die Diversität der Population. Ein zulässiges Individuum darf sein Genmaterial durch Rekombination und Mutation weitergeben. Durch die verfolgte Elite-Strategie ist eine Übernahme in die nächste Generation garantiert. Dies gilt auch für genetisch ähnliche, zulässige Nachkommen. Insbesondere, wenn keine oder wenige zulässige Individuen in der Anfangspopulation enthalten sind, bilden sich Populationen, welche ausschließlich aus miteinander verwandten Individuen bestehen. Um diese Entwicklung zu vermeiden, muss dafür gesorgt werden, dass das genetische Material unzulässiger, aber fitter Individuen länger berücksichtigt wird. Dies wird durch den angewandten Vergleichsoperator nicht erreicht, da die Dominanz bei zwei unzulässigen Individuen durch die Constraint-Verletzung und nicht durch die Werte der Zielfunktionen bestimmt wird.

Die Constraint-Verletzung soll dabei das Ausmaß der Unzulässigkeit einer Lösung beschreiben. Dies wurde im Rahmen der Arbeit nicht beachtet, da keine Normierung der einzelnen Constraint-Verletzungen stattfand. Beispielsweise führt das Vernachlässigen einer Nährstoffgrenze für Eisen zu einer sehr geringen Verletzung (kleiner 1), während eine Über- oder Unterschreitung einer Kaloriengrenze durch ein Hundertfaches repräsentiert wird. Die aufsummierte Constraint-Verletzung ist daher nicht aussagekräftig und lenkt die Suche keinesfalls in die Richtung zulässiger Bereiche.

Weiterhin ist der Entwurf der genetischen Operatoren zu diskutieren. Um die Erzeugung zulässiger Nachkommen zu fördern, wären zwei Ansätze vorstellbar gewesen: 1.) die gezielte Änderung derjenigen Zuordnungen, welche hauptsächlich zur Unzulässigkeit beitragen und 2.) die Anwendung von Reparaturmechanismen.

Bezüglich der Auswertung der Ergebnisse ist anzumerken, dass eine Adaption der Parameter des Optimierungsverfahrens vernachlässigt wurde. Daher kann keine Aussage zum Einfluss der genetischen Operatoren auf die Ergebnisse getroffen werden.

Das ausgewählte MOEA-Framework konnte aufgrund der verständlichen und umfänglichen

Dokumentation (sowohl im Handbuch als auch im Quellcode) ideal zur Implementierung des konzipierten Lösungsansatzes genutzt werden.
Die entwickelte Web-Applikation besitzt den Charakter eines Forschungsprototypen und erlaubt die Steuerung des evolutionären Prozesses durch den Nutzer. Gleichzeitig werden sinnvolle Auswertungen in angemessener Art und Weise präsentiert. Dies äußert sich insbesondere bei der Präsentation der Optimierungsergebnisse. Der Nutzer hat die Möglichkeit, alle zulässigen Lösungen der Ergebnis-Population einzusehen. Um die Exploration der Population zu erleichtern, können auf der Grundlage eines ausgewählten Plans und einer Menge anders zu belegender Mahlzeiten ähnliche Menüpläne ermittelt werden. Damit wird die gewünschte, aber nicht notwendige, assistierende Komponente zumindest im Ansatz umgesetzt.
Zu bemängeln ist die fehlende Diagnose-Möglichkeit, wenn für einen Menüplan keine Lösung gefunden wurde. Die Bedienoberfläche ist übersichtlich und intuitiv gestaltet. Es wurde eine lose gekoppelte Architektur umgesetzt, welche die Anbindung des Backends an andere Systeme ermöglicht.

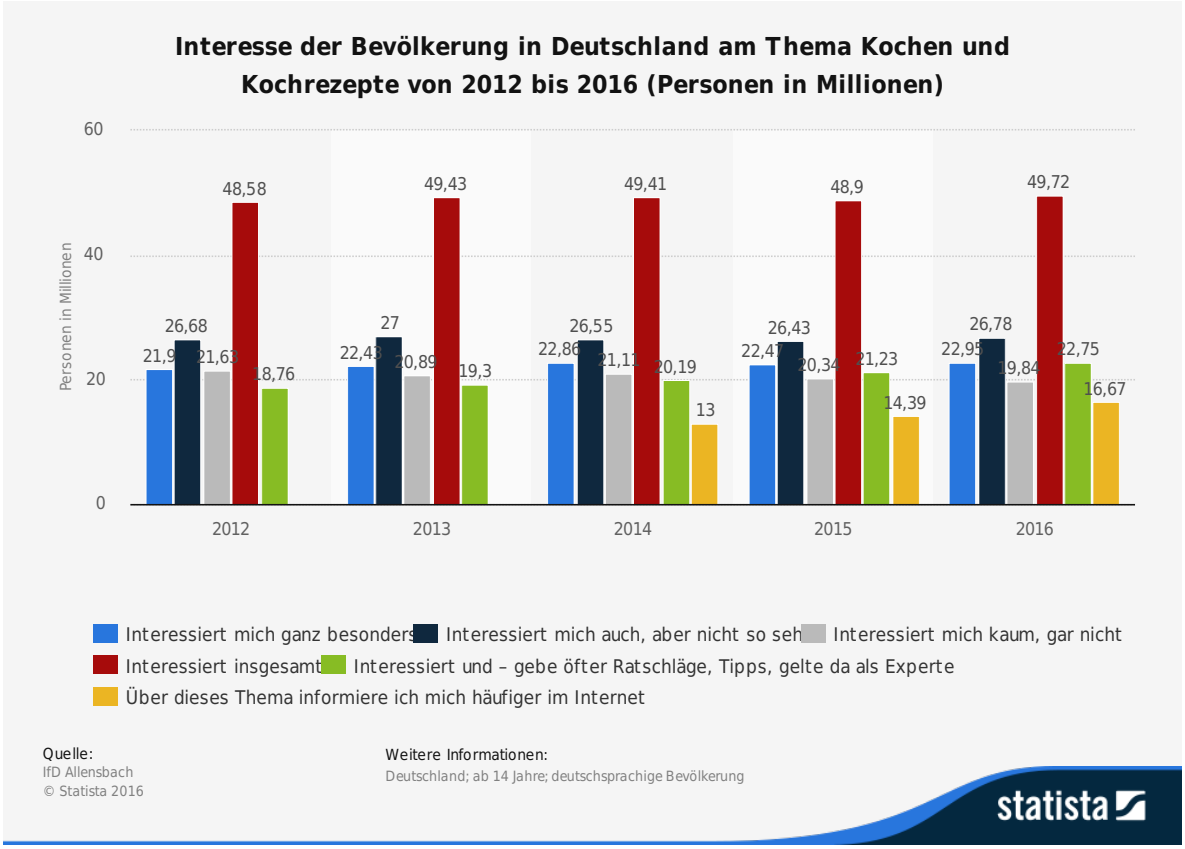
5.2 Ausblick

Diese Arbeit leistet einen wissenschaftlichen Beitrag zur Lösung eines alltäglichen Problems. Als solche kann sie vielfältig fortgeführt werden. Um die Verwendung des Wissensnetzes innerhalb realer Anwendungen zu forcieren, muss eine Überprüfung der Korrektheit durch einen Experten der Domäne erfolgen. In diesem Zusammenhang erscheint es sinnvoll, die Anwendung um eine Editierungsfunktion zu erweitern. Dadurch wird es Nutzern ermöglicht, gezielt neues Wissen (z.B. eigene Rezepte) zu integrieren oder Änderungen vorzunehmen. Unter der Annahme, dass stets zuverlässige Informationen eingespeist werden, würde die Qualität des Faktenwissens gesteigert werden. Dies ist insbesondere für diejenigen Teile des Wissensnetzes relevant, welche das Ergebnis eines automatisierten Prozesses sind - beispielsweise das BLS Matching oder die Erkennung von Einheiten und Mengenangaben.
Ebenfalls nicht Bestandteil dieser Arbeit, jedoch ein wertvoller Beitrag wäre ein Vergleich der Optimierungsergebnisse, wenn andere genetische Algorithmen genutzt werden. MOEA fungiert dabei als ideales Rahmenwerk, da zahlreiche alternative Algorithmen offeriert werden. Die innerhalb dieser Arbeit implementierten Bausteine (spezifische Variablen, genetische Operatoren, etc.) können dabei umstandslos übernommen werden.
Ebenfalls sinnvoll erscheint die Erweiterung der Web-Applikation um eine Auswertungsfunktionalität. Nutzer könnten dann für beliebige Parametrisierungen Testläufe initiieren und statistische Werte zur Entwicklung der Population erheben. Auf dieser Basis kann auch die

automatische Generierung und Bereitstellung von Diagrammen im Frontend in Betracht gezogen werden.

A Anhang

A.1 Befragung zum Interesse an den Themen Kochen und Kochrezepte



Die Befragung wurde im Rahmen der Allensbacher Markt- und Werbeträger-Analyse (AWA 2016) erhoben und durch das Institut für Demoskopie Allensbach veröffentlicht. Die Ergebnisse basieren auf einer Stichprobe von 23.854 Personen und wurden hochgerechnet auf 69,56 Millionen Personen. Die Grundgesamtheit bildet die deutschsprachige Bevölkerung ab dem 14.Lebensjahr.

A.2 Auflistung der gewählten Instanzen zur Abbildung der Konzepte

Tabelle A.1: Konzepte und jeweils gewählte Instanzen

Konzept	Vorzugsbegriff und Code der Instanz
Allergien	Hühnereiweiß-Hypersensitivität (M0351829)
	Nuß-Hypersensitivität (M0351847)
	Erdnuß-Hypersensitivität (M0351836)
	Weizen-Hypersensitivität (M0351833)
	Milchüberempfindlichkeit (M0024863)
	Meeresfrüchteallergie (91937001)
Ernährungsweisen	Diät, vegetarische (M0022577)
	Diät, veganische (M0416012)
Unverträglichkeiten	Zöliakie (M0003736)
	Laktoseintoleranz (M0012180)
Einheiten	Centi Meter (cm)
	Milli Gram (mg)
	Gram (g)
	Kilo Gram (kg)
	Milli Liter (mL)
	Centi Liter (cL)
	Deci Liter (dL)
	Liter (L)
	International Unit ([iU])
	Tropfen (drop)
	Spritzer (dash)
	Prise (pinch)
	Teelöffelvoll (teaspoon)
	Esslöffelvoll (tablespoon)
	Dose (can)
	Packung (package)
	Becher (cup)

Einheiten	Scheibe (slice)
	Bund (bunch)
	Zehe (clove)
	Knolle (bulb)
	Stange (stalk)
	Zweig (branch)
	Würfel (block)
	Glas (glass)
	Handvoll (handful)
	Stück (piece)
	Topf (pot)
	Kugel (bowl)
	Flasche (bottle)
	Kopf (head)
	Beutel (bag)

Die Tabelle listet für jedes Konzept die zum Aufbau des Faktenwissens verwendeten Vokabulareinträge mit ihrem Code.

A.3 EBNF-spezifizierte Grammatik des Mealmaster-Formats

```
1  Mealmaster   = recipe Mealmaster
2                 / *( VCHAR / SP ) eol Mealmaster
3                 /
4  recipe       = header title categories servings ingredients
       instructions footer
5  header       = separator *( VCHAR / SP ) "Meal-Master" *( VCHAR /
       SP ) eol
6  title        = '     Title: ' 0*60( VCHAR / SP ) eol
7  categories   = 'Categories: ' categlist eol
8  categlist    = 1*11( VCHAR / SP ) ["," categlist]
9  servings     = '  Servings: ' 1*4( DIGIT ) [ SP [ 1*10( VCHAR /
       SP ) ] ] eol
10 ingredients  = onecolumn
11                / twocolumn
12 onecolumn    = section onecolumn
13                / ingredone eol onecolumn
14                / *( SP ) eol onecolumn
15                /
16 twocolumn    = section twocolumn
17                / ingredtwo SP ingredone eol twocolumn
18                / ingredone eol [ *( eol ) section twocolumn ]
```

```
19                  /
20  ingredone     = amount SP unit SP 1*28( VCHAR / SP )
21  ingredtwo     = amount SP unit SP 28( VCHAR / SP )
22  amount        = 7( SP / DIGIT / "." / "/" )
23  unit          = 'x ' / 'sm' / 'md' / 'lg' / 'cn' / 'pk' / 'pn' / '
        dr' / 'ds'
24                  / 'ct' / 'bn' / 'sl' / 'ea' / 't ' / 'ts' / 'T ' /
                        'tb' / 'fl'
25                  / 'c ' / 'pt' / 'qt' / 'ga' / 'oz' / 'lb' / 'ml' /
                        'cb' / 'cl'
26                  / 'dl' / 'l ' / 'mg' / 'cg' / 'dg' / 'g ' / 'kg' /
                        '  '
27  instructions  = 0*255( VCHAR ) eol instructions
28                  / section instructions
29                  /
30  section       = separator *( VCHAR / SP ) eol
31  footer        = separator eol
32  separator     = 'MMMMM'
33                  / '-----'
34  eol           = [ CR ] LF
```

Es werden die folgenden Zeichen mit den jeweiligen Bedeutungen verwendet:

- / für alternative Regeln
- **VCHAR** für beliebige druckbare Zeichen (außer Leerzeichen)
- **DIGIT** für eine Ziffer von 0 bis 9
- **SP** für ein Leerzeichen
- **CR** für ein Carriage Return Zeichen
- **LF** für ein Line Feed Zeichen
- ***(VCHAR / SP)** für eine beliebige Anzahl druckbarer Zeichen sowie Leerzeichen oder für überhaupt kein Zeichen
- **1*4(DIGIT)** für 1 bis 4 Ziffern
- **[CR]** für ein optionales Carriage Return Zeichen
- **5('M')** für genau 5 große "M" Zeichen

Diese Spezifikation sowie weitere Anmerkungen sind [Wed10] zu entnehmen.

A.4 Yummly Dictionaries

Tabelle A.2: Dictionary-Einträge der Parameter *diet, allergy* und *course*

Parameter	Dictionary-Eintrag
diet	Lacto vegetarian
	Ovo vegetarian
	Pescetarian
	Vegan
	Lacto-ovo vegetarian
	Paleo
allergy	Gluten-Free
	Peanut-Free
	Seafood-Free
	Sesame-Free
	Soy-Free
	Dairy-Free
	Egg-Free
	Sulfite-Free
	Tree Nut-Free
	Wheat-Free
course	Main Dishes
	Desserts
	Side Dishes
	Appetizers
	Salads
	Breakfast and Brunch
	Breads
	Soups
	Beverages
	Condiments and Sauces

Parameter	Dictionary-Eintrag
course	Cocktails
	Snacks
	Lunch

A.5 Informationsmodell zur Abbildung des Domänenwissens

Listing A.1: Signaturen des Informationsmodells

```
@prefix :       <http://fokus.fraunhofer.de/eHealth/
    smartMenuInfoModel/signatures#>.
@prefix signatures:     <http://fokus.fraunhofer.de/eHealth/
    cts2infoModel/signatures#>.
@prefix psm: <http://fokus.fraunhofer.de/eHealth/
    smartMenuInfoModel/signatures#>.

@prefix sig:    <urn:negros:signatures#>.
@prefix rdfs:   <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd:    <http://www.w3.org/2001/XMLSchema#> .


:Recipe
        a rdfs:Class;

        sig:propertyConstraint [sig:onProperty :totalTime;
                rdfs:comment """
                given in minutes
                """;
                sig:range xsd:integer;
                sig:max 1;
        ];

        sig:propertyConstraint [sig:onProperty :dietConcept;
                sig:range :DietConcept;
        ];

        sig:propertyConstraint [sig:onProperty :ingredientRel;
                sig:range :IngredientRel;
                sig:min 1;
        ];

        sig:propertyConstraint [sig:onProperty :servings;
                sig:range xsd:integer;
                sig:min 1; sig:max 1;
        ];

        sig:propertyConstraint [sig:onProperty :nutritionInfo;
                sig:range :NutritionInfoPerServing;
                sig:min 1; sig:max 1;
        ].
```

```
40
41
42 :IngredientRel
43         a rdfs:Class;
44
45         sig:propertyConstraint[sig:onProperty :
               originalIngredientTerm;
46                 sig:range xsd:string;
47                 sig:min 1; sig:max 1;
48         ];
49
50         sig:propertyConstraint [sig:onProperty :ingredient;
51                 sig:range :Food;
52                 sig:min 1; sig:max 1;
53         ];
54
55         sig:propertyConstraint [sig:onProperty :amountRel;
56                 sig:range :AmountRel;
57                 sig:max 1;
58         ].
59
60
61 :AmountRel
62         a rdfs:Class;
63
64         sig:propertyConstraint [sig:onProperty :amount;
65                 sig:range xsd:string;
66                 sig:min 1;sig:max 1;
67         ];
68
69         sig:propertyConstraint [sig:onProperty :unit;
70                 sig:range :Unit;
71                 sig:max 1;
72         ].
73
74
75 :NutritionInfoPerServing
76         a rdfs:Class;
77         rdfs:subClassOf :NutritionInformation.
78
79
80 :NutritionInfoPer100g
81         a rdfs:Class;
82         rdfs:subClassOf :NutritionInformation.
83
84 :NutritionInformation
85         a rdfs:Class;
86         sig:isAbstract true;
87
88         sig:propertyConstraint [sig:onProperty :energy;
89                 rdfs:comment """
90                 given in kcal/100g or given in kcal per serving
91                 """;
92                 sig:range xsd:float;
93                 sig:min 1; sig:max 1;
94         ];
```

```
95
96          sig:propertyConstraint [sig:onProperty :carbohydrates;
97                  rdfs:comment """
98                  the amount of absorbable carbohydrates given in
                       mg/100g or in g per serving
99                  """;
100                 sig:range xsd:float;
101                 sig:min 1; sig:max 1;
102         ];
103
104         sig:propertyConstraint [sig:onProperty :protein;
105                 rdfs:comment """
106                 given in mg/100g  or in g per serving
107                 """;
108                 sig:range xsd:float;
109                 sig:min 1; sig:max 1;
110         ];
111
112         sig:propertyConstraint [sig:onProperty :fat;
113                 rdfs:comment """
114                 given in mg/100g or in g per serving
115                 """;
116                 sig:range xsd:float;
117                 sig:min 1; sig:max 1;
118         ];
119
120         sig:propertyConstraint [sig:onProperty :sodium;
121                 rdfs:comment """
122                 given in mg/100g  or in g per serving
123                 """;
124                 sig:range xsd:float;
125                 sig:min 1; sig:max 1;
126         ];
127
128         sig:propertyConstraint [sig:onProperty :cholesterol;
129                 rdfs:comment """
130                 given in mg/100g  or in g per serving
131                 """;
132                 sig:range xsd:float;
133                 sig:min 1; sig:max 1;
134         ];
135
136         sig:propertyConstraint [sig:onProperty :fiber;
137                 rdfs:comment """
138                 given in mg/100g  or in g per serving
139                 """;
140                 sig:range xsd:float;
141                 sig:min 1; sig:max 1;
142         ];
143
144         sig:propertyConstraint [sig:onProperty :iron;
145                 rdfs:comment """
146                 given in microgram/100g  or in g per serving
147                 """;
148                 sig:range xsd:float;
149                 sig:min 1; sig:max 1;
```

```
150         ];
151
152         sig:propertyConstraint [sig:onProperty :vitamin_d;
153                 rdfs:comment """
154                 given in microgram/100g or in [iU] (international
                        unit) per serving
155                 """;
156                 sig:range xsd:float;
157                 sig:min 1; sig:max 1;
158         ].
159
160
161
162 :Food
163         a rdfs:Class;
164
165         sig:propertyConstraint [sig:onProperty :isBLS_original;
166                 sig:range xsd:boolean;
167                 sig:max 1;
168         ];
169
170         sig:propertyConstraint [sig:onProperty :nutritionInfo;
171                 sig:range :NutritionInfoPer100g;
172                 sig:min 1; sig:max 1;
173         ].
174
175
176 :DietRestriction
177         a rdfs:Class;
178
179         sig:propertyConstraint [sig:onProperty :excludesFood;
180                 sig:range :Food;
181         ];
182
183         sig:propertyConstraint [sig:onProperty :excludesRecipe;
184                 sig:range :Recipe;
185         ].
186
187 :DietConcept
188         a rdfs:Class;
189         rdfs:subClassOf :DietRestriction.
190
191
192 :Allergy
193         a rdfs:Class;
194         rdfs:subClassOf :DietRestriction.
195
196
197 :Intolerance
198         a rdfs:Class;
199         rdfs:subClassOf :DietRestriction.
200
201
202 :Unit
203         a rdfs:Class;
```

A.6 Strategien zur Erweiterung der Taxonomie des BLS

1.) Subsumtion bestehender BLS Einträge

Bei diesem Vorgehen wird versucht, Verfeinerungen durch eine Rückführung auf den Grundschlüssel (die ersten vier Stellen des Codes) abzuleiten. Es wird die folgende Strategie angewandt:

1 Gruppiere alle Einträge, deren Schlüssel in den ersten vier Stellen gleich sind

2 Für jede Gruppe in 1:

  2.1 Prüfe, ob es einen Eintrag gibt, dessen Schlüssel sich aus den ersten vier Stellen und drei Nullen zusammensetzt

  2.2 Falls ja, ordne alle Einträge dieser Gruppe dem Eintrag unter

Beispiel:
Es gibt eine Gruppe von Einträgen, für welche gilt, dass alle Schlüssel mit "G200" beginnen. Die Gruppe enthält die Einträge *Blattgemüse (G200000), Blattgemüse gebacken (G200162), Blattgemüse gegart (G200122) und Blattgemüse roh (G200100)*. Es gibt einen Eintrag, dessen Schlüssel sich aus "G200" und "000" zusammensetzt, namentlich das Lebensmittel Blattgemüse. Alle anderen Einträge werden diesem Eintrag untergeordnet.

2.) Subsumtion durch Hinzufügen neuer Einträge mit dreistelligen Schlüsseln

Bei dieser Strategie werden zusätzlich zum Schlüssel die Bezeichner eines Lebensmittels einbezogen. Im Gegensatz zum vorherig beschriebenen Vorgehen wird der BLS an dieser Stelle um Einträge zur Repräsentation eines unverarbeiteten, nicht näher spezifizierten Lebensmittels erweitert. Dabei wird wie folgt verfahren:

1 Gruppiere alle Einträge, deren Schlüssel in den ersten drei Stellen gleich sind

2 Für jede Gruppe in 1.

  2.1 Ermittle die längste Zeichenkette, mit welcher alle Bezeichner der Einträge in dieser Gruppe beginnen

  2.2 Wenn eine längste Zeichenkette gefunden wurde, verfahre wie folgt:

    2.2.1 Kürze die Zeichenkette durch Weglassen führender oder endender Leerzeichen

    2.2.2 Wenn die verbleibende Zeichenkette aus weniger als drei Zeichen besteht, fahre mit der nächsten Gruppe fort

2.2.3 Wenn die verbleibende Zeichenkette aus mehreren Wörtern besteht, wird das letzte Wort entfernt, wenn es nur aus einem Zeichen besteht oder das Wort gleich “mit“, “mi“, “un“, “mind“, “für“, “with“ oder “for“ ist

2.3 Erstelle einen neuen Eintrag, dessen Bezeichner gleich der verbleibenden Zeichenkette aus 2.2.3 ist und dessen Schlüssel dem dreistelligen Gruppenschlüssel aus 1 entspricht und ordne diesem Eintrag alle zu dieser Gruppe gehörenden Lebensmittel unter

2.4 Wenn die gleiche Zeichenkette für mehrere Gruppen in 2.1 gefunden wird, dann wird der Schlüssel für den hinzugefügten Eintrag durch Aneinanderreihung der dreistelligen Gruppenschlüssel und Separierung durch ein Pipe-Zeichen (’|’) erreicht

2.5 Kann nach dem gleichen Vorgehen kein englischer Term identifiziert werden, breche den Schritt ab und fahre mit der nächsten Gruppe fort

Beispiel:
Es gibt eine Gruppe von Einträgen, für welche gilt, dass alle Schlüssel mit “G51“ beginnen. Die Gruppe enthält 20 Einträge, darunter z.B. *Aubergine (Eierfrucht) gegart (G510022), Aubergine Konserve abgetropft (G510902), Aubergine Konserve in Öl, nicht abgetropft (G510700), Aubergine Konserve, nicht abgetropft (G510900), Aubergine frittiert (zubereitet ohne Fett) (G510192), Aubergine gegart (G510122), Aubergine gegrillt (G510172)*. Die längste Zeichenkette, mit welcher alle Einträge in dieser Gruppe beginnen, ist “Aubergine“. Da auch für alle englischen Bezeichner eine solche führende Zeichenkette gefunden wurde, wird ein neuer Eintrag mit dem Schlüssel G51 und der entsprechenden Zeichenkette angelegt. Diesem werden alle Gruppeneinträge untergeordnet. Gleichzeitig stellt das neue Konzept eine Spezialisierung des zweistelligen Gruppenbegriffs (in diesem Fall G5) dar.

A.7 BLS-Matching

Die nachfolgende Tabelle A.3 umfasst Beispiele zur Verdeutlichung des Matching-Verfahrens von Zutatenbezeichnern auf Einträge des BLS. Eine Zeile enthält den ursprünglichen Zutatenbezeichner (1.Spalte), das Ergebnis der Vorverarbeitung mitsamt erkannter Mengenangaben und Einheiten (2.Spalte), den Vorzugsbegriff und Schlüssel des zugeordneten Konzeptes (3.Spalte) sowie die für das Matching herangezogene Funktionalität (4.Spalte).

Tabelle A.3: Beispiele getätigter Zuordnungen von Zutatenbezeichnern auf BLS Einträge

Zutatenbezeichner	Zeichenkette nach Vorverarbeitung (Mengenangabe, Einheit)	Vorzugsbegriff und Schlüssel des zugeordneten BLS-Konzepts	Zuordnung erfolgt durch
4 Wiener Würstchen (à ca. 60 g)	Wiener Würstchen (4, -)	Wiener Würstchen (W211200)	Suche
12 Scheibe/n Toastbrot (American Toast, große Scheiben)	Toastbrot (12, slice)	Graubrot-Toastbrot (B204000)	Vorschläge
2 kleine Paprikaschote(n), rot und grün	Gemüsepaprika (2, -)	Gemüsepaprika grün (G541000)	Vorschläge
1 Tasse Nudeln, (Fadennudeln)	Pasta (1, cup)	Teigwaren (E40)	Konzept-Matching
50 g Frühlingszwiebeln	Lauchzwiebel (50, g)	Lauchzwiebel roh (G486100)	Vorschläge
500 g Hähnchenbrustfilet(s)	Geflügel (500, g)	Geflügel (V4)	Suche
1 Stück Ingwer	Ingwerknolle (1, piece)	Ingwerknolle (R211200)	Suche
2 Zehe/n Knoblauch	Knoblauch (2, clove)	Knoblauch (G490000)	Konzept-Matching
1 Beutel KNORR Asia Gebratene Nudeln Bami Goreng	Asia Gebratene Pasta Bami Goreng (1, bag)	Teigwaren (E40)	Konzept-Matching
1 EL (5 g) gehobelte Mandeln	Mandeln (1, tablespoon)	Brötchen mit Mandeln (B506600)	Vorchläge
1 Pkg. Strudelteig	Strudelteig (1, package)	Feinbackwaren aus Strudelteig (D640000)	Vorschläge
getrockneter Thymian	Thymian (-, -)	Thymian (G082000)	Konzept-Matching

Tabelle A.4 listet Zutatenbezeichner, für welche kein BLS-Konzept zugeordnet werden konnte. In einer Zeile sind der im Rezept angegebene Zutatenbezeichner sowie die vorverarbeitete Zeichenkette und extrahierte Mengenangaben und Einheiten aufgeführt.

Tabelle A.4: Beispiele für nicht zuordenbare Zutatenbezeichner

Zutatenbezeichner	Zeichenkette nach Vorverarbeitung (Mengenangabe, Einheit)
1 Packung (200 g=4 Stück) Hot Dog Brötchen	Hot Dog Brötchen (1, package)
750 g Dinkelmehl, Vollkorn	Dinkelmehl Vollkorn (750, g)
125 g Erdnussbutter, grobe	Erdnussbutter (125, g)
1 Beutel KNORR Basis Paprika-Rahmschnitzel	Gemüsepaprika Rahmschnitzel (1, bag)
30 g Weizenflocken	Weizenflocken (30, g)
1 Glas (320 ml; Abtr.gew.: 110 g) Cranberries	Cranberries (1, glass)
400 g ausgelöstes Kasselerkotelett	Kasselerkotelett (400, g)
ca. 8 TL Aprikosen-Konfitüre	Aprikosen Konfitüre (8, teaspoon)

Für weitere Auswertungen kann die auf der DVD hinterlegte Datei matching_process.ssv herangezogen werden. Eine Zeile enthält acht durch Semikolons separierte Spalten und beschreibt das Durchlaufen des Matching-Verfahrens für einen gegebenen Zutatenbezeichner (1.Spalte). Die darauf folgenden Spalten dokumentieren das Ergebnis der Vorverarbeitung, erkannte Mengenangaben, erkannte Einheiten, den Score des Konzept-Matchings, den Score der Suche, ob ein durch die Suche gefundenes Konzept in den Vorschlägen enthalten ist und den Schlüssel und Vorzugsbegriff des letztlich zugeordneten BLS Eintrags. Ein Bindestrich besitzt die folgenden Bedeutungen:

- eine Menge oder Einheit wurde nicht erkannt (3. und 4.Spalte),
- das Konzept-Matching verlief erfolglos (5.Spalte),
- die Suche wurde übersprungen oder verlief erfolglos (6.Spalte),
- die Überprüfung der Vorschläge wurde übersprungen (7.Spalte) oder
- das Verfahren endet ohne eine Zuordnung (8.Spalte)

A.8 Beispiele für Anfrage- und Antwort-Datensätze

A.8.1 HTTP-Schnittstellen des CTS2-LE Terminologiedienstes

Listing A.2: Ergebnis des Requests GET /WebSmartMenu/rest/vocabularies/intolerances


```
{
  "Intolerance": [
    {
      "resourceID": "Mesh2014en-de",
      "preferredTerm_de": "Zöliakie",
      "code": "M0003736",
      "preferredTerm_en": "Celiac Disease"
    },
    {
      "resourceID": "Mesh2014en-de",
      "preferredTerm_de": "Lactoseintoleranz",
      "code": "M0012180",
      "preferredTerm_en": "Lactose Intolerance"
    }
  ]
}
```


Listing A.3: Ergebnis des Requests GET /WebSmartMenu/rest/vocabularies/allergies (Auszug)


```
{
  "Allergy": [
    {
      "resourceID": "Mesh2014en-de",
      "preferredTerm_de": "Milchüberempfindlichkeit",
      "code": "M0024863",
      "preferredTerm_en": "Milk Hypersensitivity"
    },
    {
      "resourceID": "SnomedCT-20150131",
      "preferredTerm_de": "Meeresfrüchteallergie",
      "code": "91937001",
      "preferredTerm_en": "Allergy to seafood"
    }
  ]
}
```


Listing A.4: Ergebnis des Requests GET /WebSmartMenu/rest/vocabularies/dietConcepts


```
{
  "DietConcept": [
    {
      "resourceID": "Mesh2014en-de",
      "preferredTerm_de": "Diät, vegetarische",
      "code": "M0022577",
      "preferredTerm_en": "Diet, Vegetarian"
    },
    {
```

```
      "resourceID": "Mesh2014en-de",
      "preferredTerm_de": "Diät, veganische",
      "code": "M0416012",
      "preferredTerm_en": "Diet, Vegan"
    }
  ]
}
```

A.8.2 HTTP-Schnittstellen der Nutzerverwaltung

Listing A.5: Ergebnis des Requests GET /WebSmartMenu/rest/users

```
[
  {
    "userId": "user1",
    "userName": "Test User"
  },
  {
    "userId": "user2",
    "userName": "Brotliebhaber"
  }

]
```

Listing A.6: Ergebnis des Requests GET /WebSmartMenu/rest/user/user2

```
{
  "userId": "user2",
  "userName": "Brotliebhaber",
  "dietRestrictions": [
    {
      "resourceID": "Mesh2014en-de",
      "preferredTerm_de": "Zöliakie",
      "code": "M0003736",
      "preferredTerm_en": "Celiac Disease"
    }
  ],
  "ratings": [
    {
      "concept": {
        "preferredTerm": "Broterzeugnisse",
        "code": "B8",
        "resourceID": "Bundeslebensmittelschluessel-v15"
      },
      "includeChildConcepts": true,
      "rating": 3
    },
    {
      "concept": {
        "preferredTerm": "Gouda",
        "code": "M402000",
        "resourceID": "Bundeslebensmittelschluessel-v15"
      },
      "includeChildConcepts": true,
      "rating": 5
```

```
    }
  ]
}
```

Listing A.7: Datensatz des Requests POST /WebSmartMenu/rest/user?userId=user3

```
{
    "userId": "user3",
    "userName": "Martina Musternutzerin",
    "dietRestrictions": [
      {
        "resourceID": "Mesh2014en-de",
        "preferredTerm_de": "Weizen-Hypersensitivität",
        "code": "M0351833",
        "preferredTerm_en": "Wheat Hypersensitivity"
      }
    ],
    "ratings": [
      {
        "concept": {
          "preferredTerm": "Zwiebeln",
          "code": "G480000",
          "resourceID": "Bundeslebensmittelschluessel-v15"
        },
        "includeChildConcepts": true,
        "rating": 5
      },
      {
        "concept": {
          "preferredTerm": "Rind",
          "code": "U10",
          "resourceID": "Bundeslebensmittelschluessel-v15"
        },
        "includeChildConcepts": true,
        "rating": 5
      }
    ]
}
```

Listing A.8: Ergebnis des Requests POST /WebSmartMenu/rest/user?userId=user3

```
{ "message":"created user with id user3"}
```

Listing A.9: Datensatz des Requests PUT /WebSmartMenu/rest/user/user1

```
{
    "userId": "user1",
    "userName": "Max Musternutzer",
    "dietRestrictions": [
      {
        "resourceID": "Mesh2014en-de",
        "preferredTerm_de": "Diät, vegetarische",
        "code": "M0022577",
        "preferredTerm_en": "Diet, Vegetarian"
      }
    ],
```

```
    "ratings": [
      {
        "concept": {
          "preferredTerm": "Kartoffeln",
          "code": "K1",
          "resourceID": "Bundeslebensmittelschluessel-v15"
        },
        "includeChildConcepts": true,
        "rating": 5
      },
      {
        "concept": {
          "preferredTerm": "Pasta",
          "code": "E4",
          "resourceID": "Bundeslebensmittelschluessel-v15"
        },
        "includeChildConcepts": true,
        "rating": 4
      }
    ]
}
```

Listing A.10: Ergebnis des Requests PUT /WebSmartMenu/rest/user/user1

```
{
  "message": "updated user with id user1"
}
```

Listing A.11: Ergebnis des Requests DELETE /WebSmartMenu/rest/user/user3

```
{
  "message": "deleted user with id user3"
}
```

Listing A.12: Ergebnis des Requests DELETE /WebSmartMenu/rest/users

```
{
  "message": "deleted all users"
}
```

A.8.3 HTTP-Schnittstellen der optimierten Menüplanung

Der nachfolgende Datensatz spiegelt die in Abschnitt 4.4 skizzierte Menüplanspezifikation des ersten Testszenarios wider. Die Population soll 100 Individuen umfassen und der Algorithmus soll nach 20000 Evaluationen terminieren. Es sollen die Standardraten gewählt werden. Dies wird erreicht, indem die Mutationsrate und die Crossover-Wahrscheinlichkeit nicht definiert werden.
Das Ergebnis dieses Requests ist auf der DVD in der Datei POST_optimize_response.json hinterlegt.

Listing A.13: Datensatz des Requests POST /WebSmartMenu/rest/menuplan/optimize

```
{
   "populationSize":100,
   "maximumEvaluations":20000,
   "days": [
      {
         "text": "1.Tag",
         "participatingUsers": [
            {
               "userId": "user1",
               "userName": "Nutzer 1",
               "minProteinPerDay": 30,
               "maxProteinPerDay": 70
            },
            {
               "userId": "user2",
               "userName": "Nutzer 2",
               "maxEnergyPerDay": 2000
            },
            {
               "userId": "user4",
               "userName": "Nutzer 4"
            }
         ],
         "meals": [
            {
               "text": "1.Mahlzeit",
               "participatingUsers": [
                  {
                     "userId": "user1",
                     "userName": "Nutzer 1"
                  },
                  {
                     "userId": "user2",
                     "userName": "Nutzer 2"
                  }
               ]
            },
            {
               "text": "2. Mahlzeit",
               "participatingUsers": [
                  {
                     "userId": "user1",
                     "userName": "Nutzer 1"
                  },
                  {
                     "userId": "user2",
                     "userName": "Nutzer 2"
                  },
                  {
                     "userId": "user4",
                     "userName": "Nutzer 4"
                  }
               ]
            },
            {
               "text": "3. Mahlzeit",
```

```
                "participatingUsers": [
                    {
                        "userId": "user1",
                        "userName": "Nutzer 1",
                        "maxCarbohydratesPerMeal": 100
                    },
                    {
                        "userId": "user2",
                        "userName": "Nutzer 2"
                    },
                    {
                        "userId": "user4",
                        "userName": "Nutzer 4"
                    }
                ]
            }
        ]
    },
    {
        "text": "2. Tag",
        "participatingUsers": [
            {
                "userId": "user1",
                "userName": "Nutzer 1"
            },
            {
                "userId": "user2",
                "userName": "Nutzer 2",
                "maxFatPerDay": "null",
                "maxCarbohydratesPerDay": 150
            }
        ],
        "meals": [
            {
                "text": "1.Mahlzeit",
                "participatingUsers": [
                    {
                        "userId": "user1",
                        "userName": "Nutzer 1"
                    },
                    {
                        "userId": "user2",
                        "userName": "Nutzer 2",
                        "minCarbohydratesPerMeal": 100
                    }
                ]
            },
            {
                "text": "2. Mahlzeit",
                "participatingUsers": [{
                    "userId": "user1",
                    "userName": "Nutzer 1"
                }]
            },
            {
                "text": "3. Mahlzeit",
```

```
                "participatingUsers": [
                    {
                        "userId": "user1",
                        "userName": "Nutzer 1",
                        "maxTimeInMinutes": 20
                    },
                    {
                        "userId": "user2",
                        "userName": "Nutzer 2"
                    }
                ]
            }
        ]
    },
    {
        "text": "3. Tag",
        "participatingUsers": [
            {
                "userId": "user1",
                "userName": "Nutzer 1"
            },
            {
                "userId": "user2",
                "userName": "Nutzer 2",
                "maxCholesterolPerDay": 0.3
            },
            {
                "userId": "user4",
                "userName": "Nutzer 4"
            }
        ],
        "meals": [
            {
                "text": "1.Mahlzeit",
                "participatingUsers": [
                    {
                        "userId": "user1",
                        "userName": "Nutzer 1",
                        "minVitaminDPerMeal": 40
                    },
                    {
                        "userId": "user2",
                        "userName": "Nutzer 2"
                    }
                ]
            },
            {
                "text": "2. Mahlzeit",
                "participatingUsers": [
                    {
                        "userId": "user1",
                        "userName": "Nutzer 1"
                    },
                    {
                        "userId": "user2",
                        "userName": "Nutzer 2"
```

```
                    }
                ]
            },
            {
                "text": "3. Mahlzeit",
                "participatingUsers": [
                    {
                        "userId": "user1",
                        "userName": "Nutzer 1"
                    },
                    {
                        "userId": "user2",
                        "userName": "Nutzer 2"
                    }
                ]
            },
            {
                "text": "4. Mahlzeit",
                "participatingUsers": [
                    {
                        "userId": "user1",
                        "userName": "Nutzer 1"
                    },
                    {
                        "userId": "user2",
                        "userName": "Nutzer 2"
                    },
                    {
                        "userId": "user4",
                        "userName": "Nutzer 4",
                        "maxFiberPerMeal": 30
                    }
                ]
            }
        ]
    }
]
}
```

Interpretation des Datensatzes

Das JSON-formatierte Objekt spezifiziert eine konkrete Instanz des Menüplanungsproblems. Darüber hinaus kann der evolutionäre Prozess durch die folgenden Parameter gesteuert werden:

maximumTime: Anzahl der Millisekunden, nach denen der Algorithmus terminiert; wird bevorzugt, wenn gleichzeitig der Parameter *maximumEvaluations* gegeben ist

maximumEvaluations: Anzahl der Bewertungen, nach denen der Algorithmus terminiert

populationSize: Anzahl der Individuen einer Population; fehlt der Parameter oder ist der Wert kleiner gleich 0, wird der Standardwert 100 verwendet

mutationRate: Mutationsrate; fehlt der Parameter oder ist der Wert kleiner 0 oder größer

1, wird der Standardwert (1/Anzahl der Variablen) verwendet
crossoverProbability: Crossover-Wahrscheinlichkeit; fehlt der Parameter oder ist der Wert kleiner 0 oder größer 1, wird der Standardwert 1 verwendet

Das zu lösende Menüplanungsproblem wird durch die folgenden Objekte und Felder beschrieben[1]:
days: enthält die zu planenden Tage
days.participatingUsers: referenziert die Nutzer, die an mindestens einer Mahlzeit an diesem Tag teilnehmen; Nährstoffgrenzwerte werden in Kilokalorien (für Energie), iU (international Unit, für Vitamin D) oder Gramm (für alle weiteren Nährstoffe) angegeben, wobei der Schlüssel durch den folgenden regulären Ausdruck beschrieben wird[2]: (min|max) (Carbohydrates|Cholesterol|Energy|Fat|Fiber|Iron|Protein|Sodium|VitaminD)PerDay
days.meals: enthält die zu planenden Mahlzeiten
days.meals.participatingUsers: referenziert die Nutzer, die an der Mahlzeit teilnehmen; Nährstoffgrenzwerte werden in Kilokalorien (für Energie), iU (international Unit, für Vitamin D) oder Gramm (für alle weiteren Nährstoffe) angegeben, wobei der Schlüssel durch den folgenden regulären Ausdruck beschrieben wird[2]: (min|max) (Carbohydrates|Cholesterol|Energy|Fat|Fiber|Iron|Protein|Sodium|VitaminD)PerMeal; die maximale Zubereitungszeit wird durch den Schlüssel *maxTimeInMinutes* angegeben

Listing A.14: Ergebnis des Requests GET /WebSmartMenu/rest/menuplans

```
[
  {
    "planId": "Beispiel_Woche_1",
    "notes": "ein umfangreicher Plan für 7 Tage"
  },
  {
    "planId": "Beispiel_Woche_2",
    "notes": "ein kleiner Plan mit nur einer einzigen Mahlzeit"
  }
]
```

Auf der DVD sind weiterhin die folgenden Dateien hinterlegt:

- POST_save_menuplan.json enthält den POST-Body des Requests POST /WebSmartMenu/rest/menuplan?planId=Testszenario 1
- GET_menuplan.json enthält die Response des Requests GET /WebSmartMenu/rest/menuplan/Testszenario 1

[1] die Verschachtelung ist dem Beispiel A.13 zu entnehmen; aus Gründen der Übersichtlichkeit werden tiefer liegende Strukturen durch einen Punkt gekennzeichnet
[2] der Zeilenumbruch ist nicht Teil des Ausdrucks

Listing A.15: Ergebnis des Requests DELETE /WebSmartMenu/rest/menuplan/Testszenario 1

```
{
  "message": "deleted plan with plan id Testszenario 1"
}
```

Listing A.16: Ergebnis des Requests DELETE /WebSmartMenu/rest/menuplans

```
{
  "message": "deleted all menuplans"
}
```

Literaturverzeichnis

[Bal64] Balintfy, J. L.: Menu Planning by Computer. In: *Commun. ACM* 7 (1964), April, Nr. 4, S. 255–259

[BBL07] Billig, A. ; Blomqvist, E. ; Lin, F.: Semantic Matching Based on Enterprise Ontologies. In: *Proceedings of the 2007 OTM Confederated International Conference on On the Move to Meaningful Internet Systems: CoopIS, DOA, ODBASE, GADA, and IS - Volume Part I.* Berlin, Heidelberg : Springer-Verlag, 2007 (OTM'07), S. 1161–1168

[BDCBVL04] Burke, E. K. ; De Causmaecker, P. ; Berghe, G. V. ; Van Landeghem, H.: The State of the Art of Nurse Rostering. In: *Journal of Scheduling* 7 (2004), Nr. 6, S. 441–499

[BHS07] Boersch, I. ; Heinsohn, J. ; Socher, R.: *Wissensverarbeitung.* 2. Auflage. Spektrum Akademischer Verlag, 2007

[BK14] Billig, A. ; Krebs, F.: Ein wissensbasierter Terminologie-Dienst zur Unterstützung von Konzept-getriebenen E-Health-Prozessen. In: Beck, E. (Hrsg.) ; Schrader, T. (Hrsg.) ; Wikarski, D. (Hrsg.) ; FH Brandenburg (Veranst.): *MedPro 2014. Der informierte Mensch in der Medizin - Prozesse, Daten und Entscheidungen.* 2014

[Box57] Box, G. E. P.: Evolutionary Operation: A Method for Increasing Industrial Productivity. In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 6 (1957), Nr. 2, S. 81–101

[BZ12] Burkard, R.E. ; Zimmermann, U.T.: *Einführung in die Mathematische Optimierung.* Springer Spektrum, 2012

[CA15] Choosri, N. ; Anprasertphon, S.: Hospital dietary planning system using constraint programming. In: *Fifth International Conference on Innovative Computing Technology (INTECH)*, 2015, S. 17–22

[CLRS09] Cormen, T. H. ; Leiserson, C. E. ; Rivest, R. L. ; Stein, C.: *Introduction to Algorithms.* 3. Auflage. MIT Press, 2009

[Coh78] Cohon, J. L.: Multiobjective Programming and Planning. In: *Mathematics in Science and Engineering - Volume 140*, Elsevier, 1978, S. 1 – 12

[Dan66] Dantzig, G. B.: Stiglers Ernährungsmodell, Beispiel einer Formulierung und Lösung. In: Jaeger, A. (Hrsg.): *Lineare Programmierung und Erweiterungen.* Berlin, Heidelberg : Springer Berlin Heidelberg, 1966, S. 625–643

[DD05] Domschke, W. ; Drexl, A.: *Einführung in Operations Research.* 6. Auflage. Springer, 2005

[Deu] Deutsche Gesellschaft für Ernährung e. V.: *Vollwertig essen und trinken nach den 10 Regeln der DGE.* https://www.dge.de/ernaehrungspraxis/vollwertige-ernaehrung/10-regeln-der-dge/. – letzter Zugriff am 29.05.16

[DN11] Durillo, Juan J. ; Nebro, Antonio J.: jMetal: A Java framework for multi-objective optimization. In: *Advances in Engineering Software* 42 (2011), Nr. 10, S. 760 – 771. – aktuelle Version unter http://jmetal.github.io/jMetal/, letzter Zugriff am 06.01.17

[Dom95] Domschke, W.: *Logistik: Transport.* 4. Auflage. Oldenbourg Wissenschaftsverlag, 1995

[DPAM02] Deb, K. ; Pratap, A. ; Agarwal, S. ; Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. In: *IEEE Transactions on Evolutionary Computation* 6 (2002), Apr, Nr. 2, S. 182–197

[DPS12] Dickerson, J. P. ; Procaccia, A. D. ; Sandholm, T.: Optimizing Kidney Exchange with Transplant Chains: Theory and Reality. In: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)* (2012)

[ETV07] Erkut, E. ; Tjandra, S.A. ; Verter, V.: Handbooks in Operations Research and Management Science. Elsevier, 2007, S. 539 – 621

[Fel99] Feldmann, M.: *Naturanaloge Verfahren - Metaheuristiken zur Reihenfolgeplanung.* 1.Auflage. Deutscher Universitätsverlag, 1999

[Fra57] Fraser, A. S.: Simulation of genetic systems by automatic digital computers. I. Introduction. In: *Australian Journal of Biological Science* 10 (1957), S. 484–491

[Fre97] Freuder, E. C.: In Pursuit of the Holy Grail. In: *Constraints* 2 (1997), Nr. 1, S. 57–61

[GKK04] Gerdes, I. ; Klawonn, F. ; Kruse, R.: *Evolutionäre Algorithmen, Genetische Algorithmen - Strategien und Optimierungsverfahren - Beispielanwendungen.* Vieweg, 2004

[Gru93] Gruber, T. R.: A Translation Approach to Portable Ontology Specifications. In: *Knowledge Acquisition* 5 (1993), Nr. 2, S. 199–220

[HB86] Holloran, T. J. ; Byrn, J. E.: United Airlines Station Manpower Planning System. In: *Interfaces* 16 (1986), Februar, Nr. 1, S. 39–50

[HL01] Hillier, F. S. ; Lieberman, G. J.: *Introduction to Operations Research.* 7. Auflage. McGraw-Hill, 2001

[Hol75] Holland, J. H.: *Adaptation in Natural and Artificial Systems.* Reprint 1992. The University of Michigan Press, 1975

[HSS16] Hartmann, B. M. ; Schmidt, C. ; Sandfuchs, K.: *Bundeslebensmittelschlüssel (BLS) Version 3.02.* https://www.blsdb.de/bls?background. Version: 2016. – letzter Zugriff am 16.11.16

[Ihr99] Ihringer, T.: *Diskrete Mathematik: eine Einführung in Theorie und Anwendungen.* 2. durchgesehene Auflage. B.G. Teubner Stuttgart Leipzig, 1999

[KLW95] Kifer, M. ; Lausen, G. ; Wu, J.: Logical Foundations of Object-oriented and Frame-based Languages. In: *J. ACM* 42 (1995), Juli, Nr. 4, S. 741–843

[Kre16] Krebs, F.: Aufbau eines RDFS-Wissensnetzes für ein Speiseempfehlungssystem im Krankenhaus / FH Brandenburg. 2016. – Studienarbeit. – auf CD-Anhang hinterlegt

[KV12] Korte, B. ; Vygen, J.: *Kombinatorische Optimierung.* 2. Auflage. Springer Spektrum, 2012

[LGRT11] Lukasiewycz, M. ; Glass, M. ; Reimann, F. ; Teich, J.: Opt4J - A Modular Framework for Meta-heuristic Optimization. In: *Proceedings of the Genetic and Evolutionary Computing Conference (GECCO 2011).* Dublin, Ireland, 2011, S. 1723–1730

[Luk15] Luke, S.: *The ECJ Owner's Manual - A User Manual for the ECJ Evolutionary Computation Library.* http://cs.gmu.edu/~eclab/projects/ecj/docs/manual/manual.pdf. Version: 2015. – letzter Zugriff am 08.01.17

[MCVH+14] Mak, T. W. K. ; Coffrin, C. ; Van Hentenryck, P. ; Hiskens, I. A. ; Hill, D.: Power System Restoration Planning with Standing Phase Angle and

Voltage Difference Constraints. In: *Proceedings of the 18th Power Systems Computation Conference* (2014)

[MPS99] Marling, C. R. ; Petot, G. J. ; Sterling, L. S.: Integrating case-based and rule-based reasoning to meet multiple design constraints. In: *Computational Intelligence* 15 (1999), Nr. 3, S. 308–332

[NM01] Noy, N. F. ; McGuinness, D.L. ; Stanford University (Hrsg.): *Ontology Development 101: A Guide to Creating Your First Ontology*. Stanford University, 2001

[OVGB10] O'Neill, M. ; Vanneschi, L. ; Gustafson, S. ; Banzhaf, W.: Open Issues in Genetic Programming. In: *Genetic Programming and Evolvable Machines* 11 (2010), September, Nr. 3-4, S. 339–363

[PFL16] Prud'homme, C. ; Fages, J.-G. ; Lorca, X.: *Choco Documentation*. http://choco-solver.org. Version: 2016. – letzter Zugriff am 06.01.17

[RN09] Russel, S. ; Norvig, P.: *Artificial Intelligence: A Modern Approach*. 3. Auflage. Prentice Hall, 2009

[SD94] Srinivas, N. ; Deb, K.: Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. In: *Evolutionary Computation* 2 (1994), Nr. 3, S. 221–248

[Sel09] Seljak, B. K.: Computer-based dietary menu planning. In: *Journal of Food Composition and Analysis* 22 (2009), Nr. 5, S. 414 – 420. – 7th International Food Data Conference: Food Composition and Biodiversity

[Soc08] Socher, R.: *Theoretische Grundlagen der Informatik*. 3. aktualisierte und erweiterte Auflage. Hanser Verlag, 2008

[Spe92] Spears, W. M.: Crossover or Mutation? In: *Foundations of genetic algorithms* 2 (1992), S. 221–237

[Spe95] Spears, W. M.: Adapting Crossover in Evolutionary Algorithms. In: *Evolutionary programming*, 1995, S. 367–384

[SSV11] Schutt, A. ; Stuckey, P. J. ; Verden, A. R.: Optimal Carpet Cutting. In: *Principles and Practice of Constraint Programming – CP 2011: 17th International Conference, CP 2011, Perugia, Italy, September 12-16, 2011. Proceedings* (2011), S. 69–84

[Wed10] Wedekind, J.: *Mealmaster format*. http://www.wedesoft.de/anymeal-api/mealmaster.html#mealmasterFormatGrammarLegend. Version: 2010. – letzter Zugriff am 10.10.16

[Wei02] Weicker, K.: *Evolutionäre Algorithmen*. Teubner, 2002

Abkürzungsverzeichnis

BLS Bundeslebensmittelschlüssel

CP Constraint Programmierung

CSS Cascading Style Sheets

CTS2 Common Terminology Services Release 2

CTS2-LE CTS2 - Linked Data Edition

HL7 Health Level 7

JSON Javascript Object Notation

MeSH Medical Subject Headings

NSGA-II Nondominated Sorting Genetic Algorithm II

OMG Object Management Group

OP Operations Research

PDF Portable Document Format

REST Representational State Transfer

RDF Resource Description Framework

RDFS Resource Description Framework Schema

SNOMED-CT Systematized Nomenclature of Human and Veterinary Medicine - Clinical Terms

UCUM Unified Code for Units of Measure

URI Uniform Resource Identifier

W3C World Wide Web Consortium

Abbildungsverzeichnis

Tabellenverzeichnis