Zürcher Hochschule
für angewandte Wissenschaften
School of Engineering

Kurzfassung der Bachelorthesis (Roman Lickel, 14. Mai 2015)

EVALUATION VON LÖSUNGSKONZEPTEN VON NP-VOLLSTÄNDIGEN PROBLEMEN IN WEBAPPLIKATIONEN

Am Beispiel des Travelling Salesman Problems

Ein Endkunde (EK) führt eine Reiseunternehmung und verwaltet die Reiseziele und Hotels in einem auf FileMaker basierten Computerprogramm. Damit EK die Reiserouten optimieren kann, ist eine Systemerweiterung des Programms gefordert, welche mittels Webtechnologien umgesetzt werden soll. Die Routenoptimierung soll die auf die Fahrzeit bezogen schnellste Rundreise berechnen, wobei die Erweiterbarkeit gewährleistet werden soll, damit beispielsweise zu einem späteren Zeitpunkt die optimale Rundreise auch bezogen auf die Distanz berechnet werden kann.

Die Optimierung von Wegrouten von einem Startpunkt über mehrere Wegpunkte (WP) zurück zum Startpunkt wird als Travelling Salesman Problem (TSP, Problem des Handelsreisenden) bezeichnet. Da beim TSP der Aufwand für die exakte Berechnung der Lösung für eine steigende Anzahl WP mehr als exponentiell zunimmt, wird mittels geeigneter Architekturen und Näherungsalgorithmen versucht das Problem für den jeweiligen Anwendungsfall effizient zu lösen. Solche Probleme werden als NP-vollständig bezeichnet, da sich diese nichtdeterministisch in Polynomialzeit lösen lassen.

In der vorliegenden Bachelorarbeit werden mittels einer Markt- und einer methodengestützte Anforderungsanalyse die genauen Anforderungen an das System erarbeitet. Es wird geprüft, inwiefern die aus der Literatur zusammengetragenen und aus anderen Systemen bekannten Architekturen auf das Webumfeld anwendbar sind. Um die Architekturvarianten in der anschliessenden Nutzwertanalyse vergleichen zu können, werden nach Auswahl eines für den Versuch geeigneten Algorithmus Konzepte erstellt. Aus allen Konzepten ergibt sich ein Gesamtkonzept für das Lösen des TSP im Webumfeld.

Die Gegenüberstellung der ausgewählten Algorithmen ergibt, dass für den Vergleich der Konzepte der Branch-and-Bound-Algorithmus der geeignete ist, da dieser parallelisier- und verteilbar ist und die exakte Lösung für das TSP berechnet.
Die ausgearbeiteten Konzepte sind in clientseitige, severseitige und verteilte Systeme unterteilt. Während bei clientseitigen Konzepten die Berechnung direkt im Webbrowser erfolgt, wird die Lösung bei serverseitigen Konzepten auf einem Server berechnet. Es wird zwischen sequentiellen Berechnungen und parallelisierten Ansätzen unterschieden. Die verteilten Systeme lösen das Problem mittels Verteilung der Berechnung auf mehrere Clients, mehrere Server oder auf ein gemischtes System von mehreren Clients und Servern. Als Proof of Concepts werden die clientseitigen, serverseitigen und ein verteiltes Konzept umgesetzt. Es stellt sich heraus, dass serverseitige Architekturen für den vorliegenden Fall ungeeignet sind, da die Serverinfrastruktur grosse Kosten verursachen und Rechenleistungsengpässe entstehen können, falls viele Anfragen gleichzeitig eintreffen. Auch die verteilten Systeme sind wegen dem grossen Implementationsaufwand und der nicht ausreichenden Performanz ungeeignet.
Neben der Architekturwahl stellt sich die Algorithmenwahl als weitaus wichtigeren Einflussfaktor auf die Berechnungsgeschwindigkeit heraus, weil die Webtechnologien nicht für rechenintensive Aufgaben entworfen wurden und somit nur durch den Einsatz eines besseren Algorithmus die nötige Effizienz erreichen. Messungen mit einer vergleichbaren Implementierung in Java zeigen, dass sowohl die sequentielle als auch die parallelisierte Berechnung mit der systemnäheren Programmiersprache Java derjenigen von JavaScript hinsichtlich der Berechnungsgeschwindigkeit weit überlegen ist.
Im resultierenden Konzept wird aufgezeigt wie die Realisierung eines Systems zur Berechnung des TSP im Webumfeld sinnvoll umgesetzt werden kann. Um die bestmögliche Lösung des Problems in der vorgegeben Zeit zu erhalten bietet sich die Lösung mittels Aufteilung in Teilprobleme nach der Anzahl WP an. Aufgrund davon wird entschieden, welcher Algorithmus und welche Architektur für die Berechnung verwendet wird. Für wenige WP erfolgt die Berechnung mittels dem Held-Karp-Algorithmus direkt im Applikationsthread auf dem Client. Ab 4 WP würde der Browser zu lange blockiert und abstürzen, weshalb die Berechnung in einen separaten Berechnungsthread (Webworker in JavaScript) ausgelagert wird. Alle nachfolgenden Stufen werden ebenfalls mit einem separaten Berechnungsthread gelöst. Ab 20 WP kann nicht mehr die exakte Lösung berechnet werden und die Berechnung erfolgt mit dem 3-Opt Algorithmus mit anschliessender Ameisen-Kolonie-Optimierung. Alle Probleme mit 300 oder mehr WP werden mit dem unpräzisen, dafür weitaus schnelleren Nearest-Neighbour-Algorithmus gelöst. Der erstellte Prototyp des resultierenden Konzepts zeigt, dass die Optimierung des TSP mit den gewünschten Anforderungen als Webapapplikation umgesetzt werden kann. Zusätzlich wird das Konzept durch das nicht im Rahmen der Arbeit umgesetzte Produkt erfolgreich bestätigt.