HTWG Konstanz

TR-Optimizer 2.1

Dokumentation

Markus Rommelfanger (286812)
Nicolas Tedjadharma (286468)

INHALT

1. Einleitung

Im Rahmen des Faches Anwendung der linearen Optimierung im 7.Semester, sollte ein Tool aus der Methodenbank herangezogen werden, auf Mängel überprüft und dementsprechend verbessert werden. Unsere Auswahl fiel auf das Tool TR-Optimizer 2.0. Die nachfolgende Dokumentation beschreibt den Projektverlauf und das entstandene Resultat.

2. Die bisherige Version

Im WS 2002/2003 erstellte eine Gruppe von Studenten das in Java programmierte OR-Tool „TR-Optimizer“ in seiner ersten grafischen Version. Das Tool dient zur Lösung von Transportproblemen und bietet dem Benutzer eine grafische Oberfläche. Das Tool wurde im WS 2008/2009 überarbeitet zu "TR-Optimizer 2.0" mit verbesserter Methodenlogik sowie Bedienbarkeit. Beim genaueren Hinblick fielen uns jedoch noch einige Bedienschwächen auf, sowie fehlende Möglichkeiten, die das Tool in Zukunft jedoch bieten sollte, um in der Methodenbank weiter attraktiv zu bleiben. Unsere Ist-Analyse beschreibt die bisherige Version von TR-Optimizer und deren Nachteile.

2.1 Ist-Analyse

Was ist TR-Optimizer?

TR-Optimizer ist ein Programm der BESF-Methodenbank zum Optimieren von Transportproblemen. Die Besonderheit daran ist die Möglichkeit zwischen verschiedenen Transportalternativen zu unterscheiden. TR-Optimizer Version 2.1 wurde in Java unter Verwendung von Eclipse Luna realisiert. Zur

Berechnung der Lösungen ist der Solver LP-Solve, Weidenauer und MOPS eingebunden.

Was kann TR-Optimizer ?

TR-Optimizer berechnet kapazitätsabhängig die optimale Lösung von Transportproblemen. Dabei sind Lösungen für den Fall „ Produzierte Menge > Nachgefragte Menge“ berücksichtigt. Ebenfalls werden verschiedene Transportalternativen berücksichtigt. TR-Optimizer kann über Drag and Drop von grafischen Elementen bedient werden. Speichern und Öffnen bereits definierter Probleme sind möglich.

2.2 Nachteile

TR-Optimizer 2.0 verfügt bereits über fundamentale Funktionalitäten, die die Lösung eines Transportproblems realisieren können. Jedoch fehlen ein Paar Funktionen, welche die Nutzung dieses Programms optimieren können

zum Beispiel:

- Keine Schnell speichern Funktion
- Keine Autosave Funktion
- Weidenauer und MOPS Solver sind in dieser Version nicht lauffähig
- Es besteht keine Möglichkeit, die Solver Pfade frei anzupassen
- Es gibt noch einige optischen Mangeln

3. Planung der Verbesserungsmaßnahmen

3.1 Anforderungs-Analyse

Funktionale Anforderungen

• Die Funktion Schnell-Speichern mit Tastenkürzel STRG + S soll implementiert werden

• Nach 2 Minuten soll das eingegebene Modell automatisch gespeichert werden

• Das Ergebnis der Solver soll in der Methode angezeigt werden

• Der Benutzer soll Solver Pfade selbständig anpassen können

Nichtfunktionale Anforderungen

• Anpassung der bestehenden Dokumentation

• Änderungen werden durch Kommentare bzw. farbliche Markierungen im Code des Programms markiert

• Testen der Ergebnisausgabe und Plausibilitätsprüfung

3.2 Projektziel

Unsere Zielsetzung war es, das Tool durch Verbesserung der Bedienschwächen in seiner Nutzung komfortabler zu machen, technische Mängel zu beheben und durch weitere Funktionen ein breiteres Nutzungsspektrum zu schaffen. Die Methode wird so erweitert, dass der Benutzer die Einstellung für die Solver-Pfade selbstständig ändern kann. Die Funktionen „Schnell-Speichern“ sowie „Auto-Speichern“ werden implementiert. Außerdem werden die Solver Weidenauer Optimizer und MOPS in dieser Methode zum Laufen gebracht.

4. Projektverlauf

Am Anfang des Projektes schauten wir uns zunächst die Präsentation von TR-Optimizer 2.0 an. Danach probierten wir die Methode etwas aus um die Funktionsweise besser zu verstehen. Die Hilfe-Funktion der Methode lieferte

etwas mehr Klarheit, wie der LP-Ansatz umgesetzt wurde. Anhand der Aufgabenstellung formulierten wir unser Pflichtenheft. Danach fingen wir an uns mit der Programmierung genauer zu befassen. Hierfür importierten wir den Source Code als ein neues Projekt in Eclipse. Als erstes lasen wir uns die einzelnen Klassen durch um einen genaueren Überblick zu erhalten. Dann gingen wir dazu über, einige Testläufe zu starten in denen wir nur minimale Änderungen durchführten. Probleme bestanden vor allem darin, sich in dem fremden Code zu Recht zu finden. Obwohl dieser gut auskommentiert war, hatten wir anfangs Schwierigkeiten die Zusammenhänge der Klassen und Methoden nachzuvollziehen. Als wir dann erste Fortschritte erzielten gingen wir dazu über unsere technischen Anforderungen zu implementieren. Als die Methode soweit lauffähig war, testeten wir die Änderungen auch unter Windows 8 um auch unsere Rahmenbedingungen des Pflichtenheftes zu erfüllen. Nach erfolgreichen Tests exportierten wir unser Projekt in eine ausführbare Datei und fertigten diese Dokumentation an.

5. Resultat

5.1 Geänderte Klassen

Im Laufe der Verbesserungen an TR-Optimizer haben wir folgende Klassen bearbeitet:

5.1.1 Neu erzeugte Klassen

AutoSave.java

- In dieser Klasse wurde die Autospeichern Funktion mit Timer implementiert.

GlobaleVariable.java

- Globale Variable für Statusidentifikation mit dem Ziel Werte einfacher zu ändern.

IniPaths.java

- Beinhaltet die Methode, die das Einlesen von Pfaden in einer ini-Datei ermöglicht.

5.1.2 Geänderte Klassen

MainFrame.java

In dieser Klasse wurden die Menüs der GUI angepasst. Folgende Änderungen wurden durchgeführt:

- Autosave einstellen
- Speichern (Strg+S)
- Menü für Einstellung Solverpfade
- Pfade Anzeigen
- Meldung bei erfolgreichem Speichervorgang
- Meldung, wenn die Autosave aktiviert wurde, aber die aktuelle Arbeit noch nicht gespeichert ist.

Solve.java

- Alle "manuell" geschriebenen Pfade wurden durch Variable von IniPaths.java ersetzt. Das ermöglicht Benutzer, die Solver Pfade anzupassen.
- Alte Methode "calculateMOPS98" wurde gelöscht, da das Programm diese nicht mehr benötigt.

SymbolPane.java

- Optische Verbesserungen durchgeführt

TransopController.java

- Name des verwendeten Solvers wird mit der Ausgabe des Ergebnisses angezeigt.

TROptimizer.java (Main Klasse)

- Autospeichern Status bei Programmstart: FALSE. Das bedeutet, bei jedem Programmstart muss der Benutzer die Autosave Funktion zuerst einstellen um sie verwenden zu können. Durch Änderung auf True ist die Autospeichern Funktion bei jedem Start verfügbar.

5.2 Neue Features

Autosave Funktion

- Die Arbeit wird nach 2 Minuten automatisch gespeichert
- Diese kann ein-/ausgeschaltet werden

Schnell speichern

- Mit Tastenkombination "STRG + S" können Benutzer die aktuelle Arbeit speichern.

Pfadeinstellung

- Solverpfade sind somit dynamisch. Benutzer können Solverpfade frei anpassen.

MOPS und Weidenauer

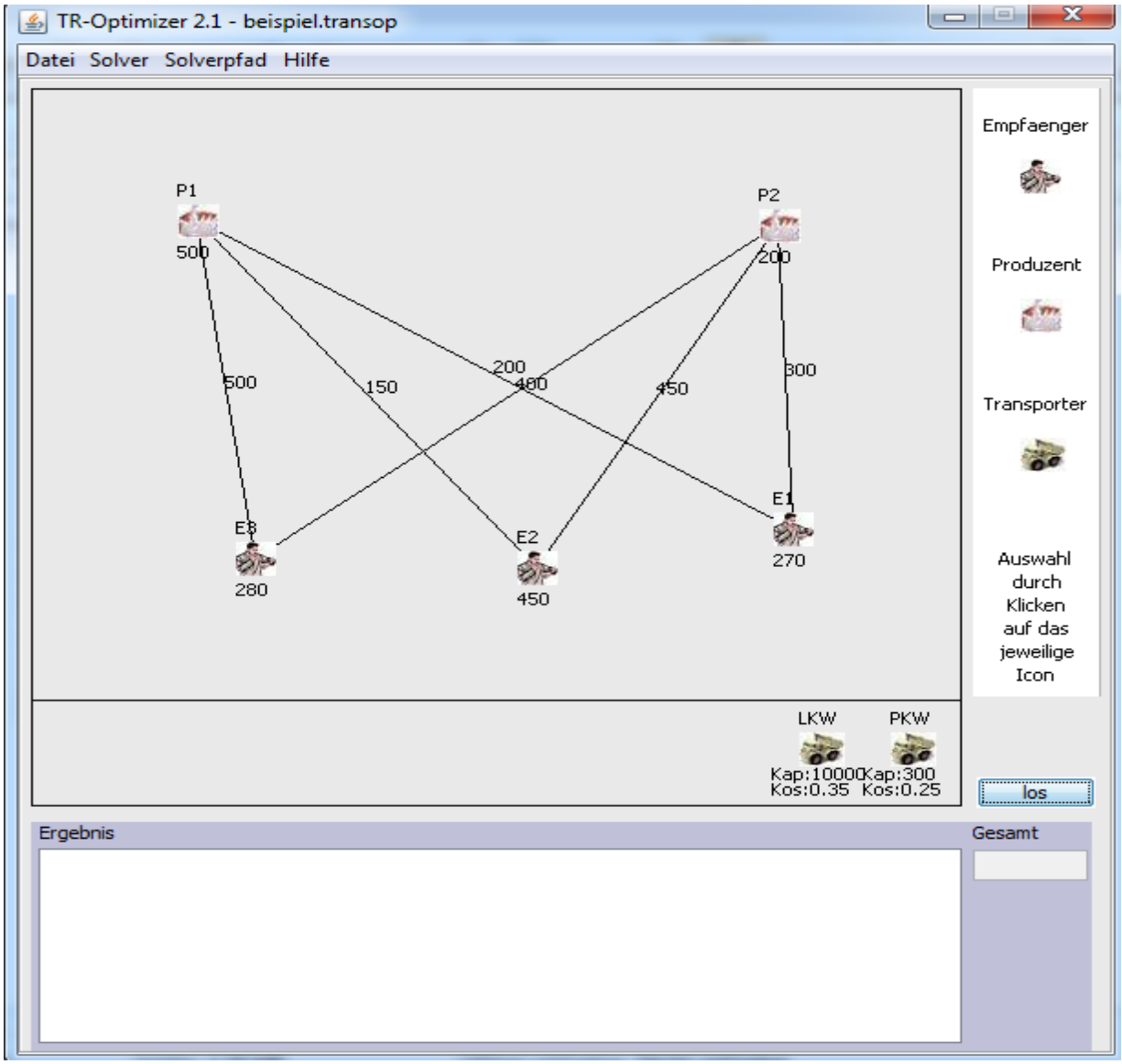
- sind nun als Solver Alternativen verfügbar

Optische Verbesserungen

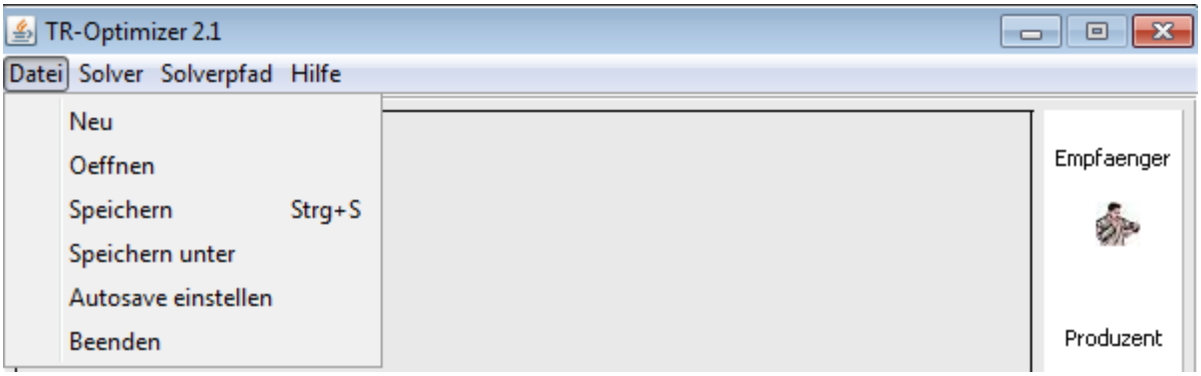
- Dateiname wird nun neben dem Titel angezeigt und der Solvertyp wird im Ergebnis angezeigt.

5.3 Benutzerschnittstellen

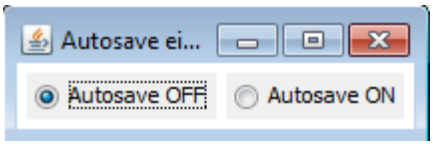
- *Main Interface*



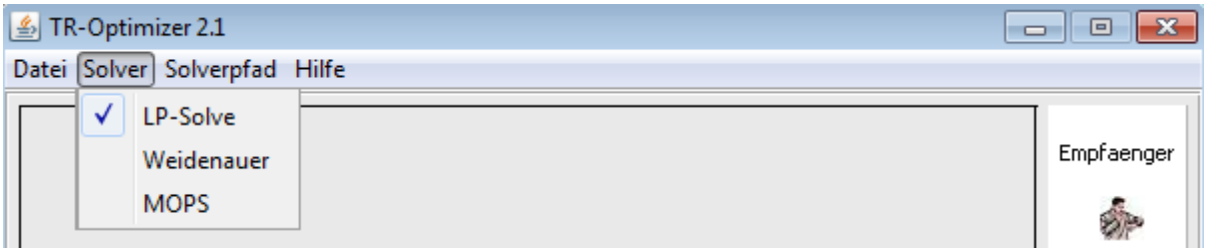
- *Menüs unter Menüpunkt "Datei"*



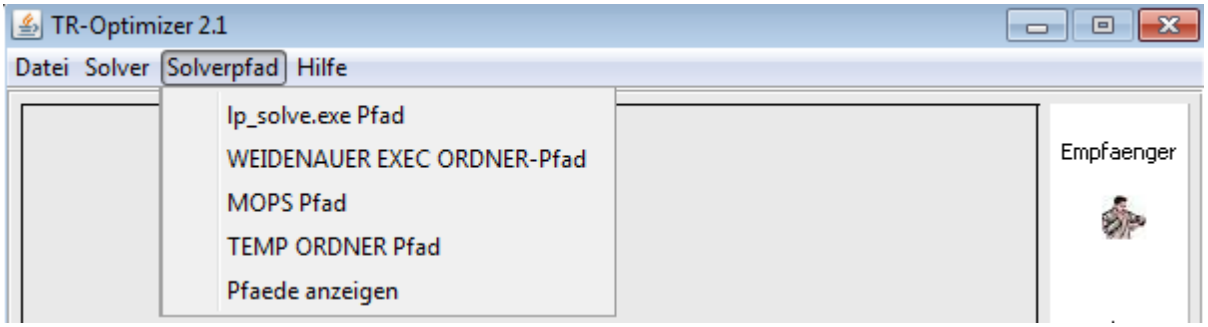
- *Interface für die Autosave Einstellung Funktion*



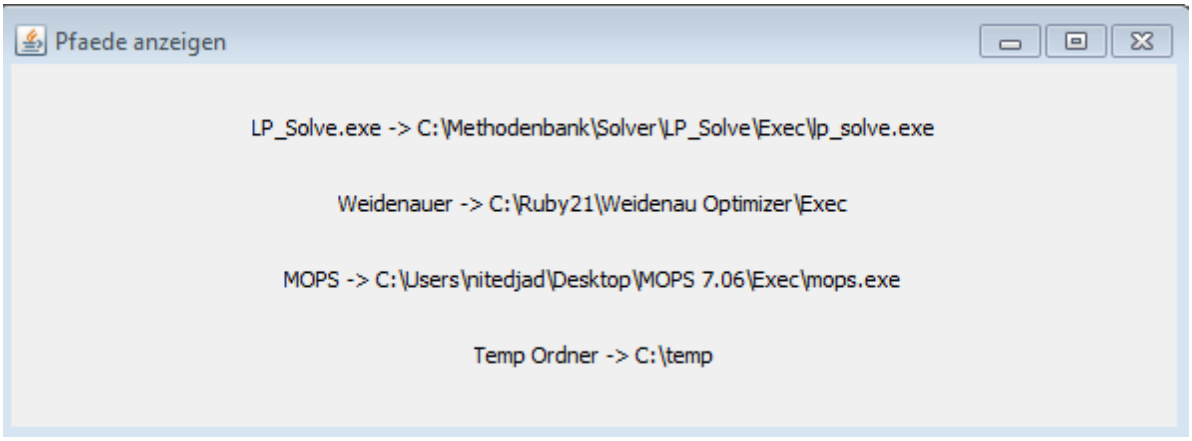
- *Solver können unter Menüpunkt "Solver" ausgewählt werden*



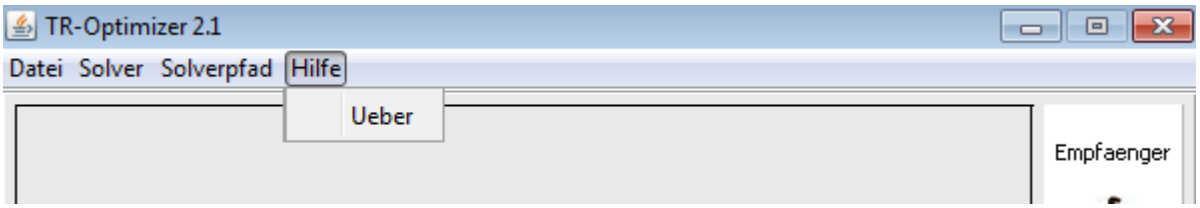
- *Menüs unter Menüpunkt "Solverpfad"*



- *Ausgabe für die Menü Pfade anzeigen*



- *Menü unter Menüpunkt "Hilfe"*



6. Installationshinweise

Um TR-Optimizer 2.1 verwenden zu können, sind folgende Punkte zu beachten:

- Windows 7, Windows 8 oder Windows 8.1
- Java 8 Update 45

Die Datei "TROptimizer2.1.jar" kann direkt gestartet werden. Für die Nutzung außerhalb des Campusnetzwerks müssen die Solver kopiert werden und die Pfade in der Methode angepasst werden.

7. **Projektaufwand** und Erfahrungen

7.1. Allgemeine Projekterfahrungen

Vorerst gestaltete sich der Verlauf des Projektes zeitlich und aufwandbezogen als problemlos machbar. Allerdings benötigten wir mehr Zeit als vermutet um uns in den fremden Code einzuarbeiten. Unsere Vorgehensweise war zielgerichtet und wir arbeiteten schrittweise die Punkte der Anforderungs-Analyse ab.

7.2. Projektaufwand

- Auto Speichern Funktion ca. 15 Stunden
- Schnell Speichern Funktion ca. 2 Stunden
- Pfad Einstellung Funktion ca. 5 Stunden
- Solver zum Laufen bringen ca. 2 Wochen
- Source-Code auskommentieren ca. 1 Stunde
- Source-Code "reinigen ca. 2 Stunden
- Hilfe / Dokumentation aktualisieren ca. 10 Stunden
- Optische Darstellung verbessern ca. 2 Stunden
- Testen ca. 3 Stunden