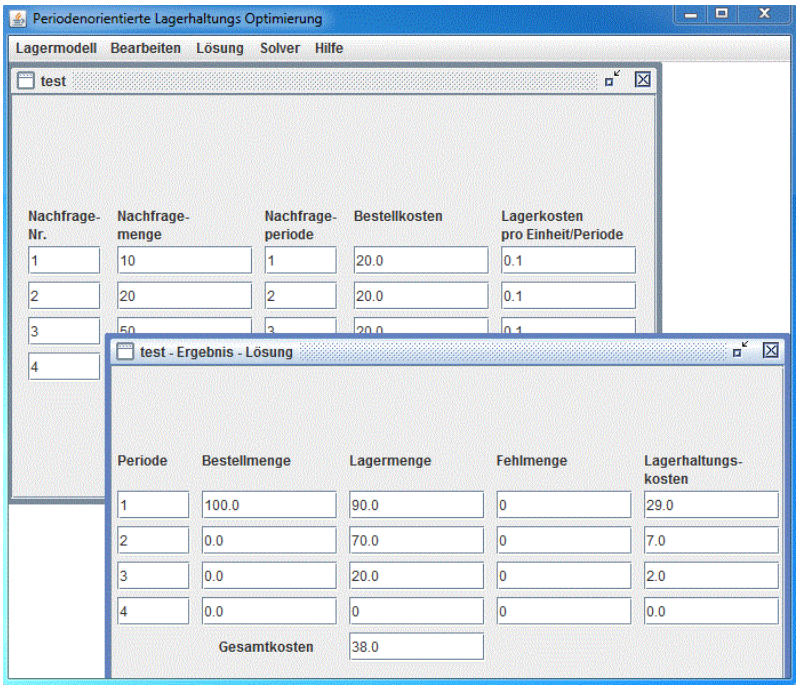
Projekt – Dokumentation

Wagner Whitin

LP 1.1



Teammitglieder

Eugen Gering - 287426
Melisa Gündüz - 289100

Projektname

Wagner Whitin LP 1.0

Projektnummer

ALO Projekt Nr.3
WS 2015/16

Abgabetermin

31.01.2016

Abbildungsverzeichnis

Tabellenverzeichnis

Inhaltsverzeichnis

1 Einleitung

Im Rahmen der Veranstaltung „Anwendung der linearen Optimierung“ sollen die in der Methodenbank OR-Alpha aufgeführten Methoden analysiert und verbessert werden. Um dieses Fach erfolgreich abschließen zu können, müssen drei Teilmodule der Methodenbank analysiert und die Ergebnisse vorgestellt werden. Weiterhin muss eine Projektarbeit in Zweierteams durchgeführt werden. Dieses Projekt beinhaltet die Entwicklung und Systemanalyse von Methoden im Bereich der logistischen Problemstellung.

Das Modul wird mit drei European Credit Transfer System (ECTS) – Punkten gewichtet und wird am 21. Januar 2016 abgeschlossen. Das Projekt beschäftigt sich mit der Methode „Wagner Whitin LP 1.0“. Diese hat zu Beginn des Semesters einige Mängel, die bis Ende des Projekts behoben worden sind.

2 Methodenbeschreibung

2.1 Historisch

Thomson M. Whitin und Harvey M. Wagner stellten 1958 ein Verfahren unter dem Namen „Wagner-Whitin-Algorithmus“ vor. Hierbei wird die optimale Losgröße für Produkte berechnet, die eine dynamische Nachfrage besitzt.[1] Die Methode ermittelt die Bestell- und die Lagerkosten, Bestell- und Lagermengen je Periode um die Gesamtkosten der Lagerhaltung möglichst gering zu halten. Die kostengünstigste Alternative wird analysiert.

2.2 Vorgehensweise der Wagner-Whitin Methode

Im Folgenden wird das Verfahren des Wagner-Whitins anhand eines Beispiels einer Produktionsstätte erläutert. [2]

2.2.1 Ausgangssituation

Eine Produktionsstätte bildet seinen Bedarf der folgenden vier Perioden ab. Die Lagerkosten betragen pro Periode 0,8 GE, falls eine Lagerung vorgenommen werden muss. Bei jedem Rüstvorgang entstehen Kosten in Höhe von 40 GE, diese Kosten sind die Rüstkosten. In Tabelle 1 ist die Ausgangssituation zusammengefasst und im Überblick. Im weiteren Verlauf der Methode werden die entsprechenden Kosten für die Produktion und Lagerung eingefügt. Dabei ist das Minimalprinzip anzuwenden, diese besagt, dass die Alternative mit geringsten Kosten ausgewählt wird.

Bedarf	**20**	**60**	**20**	**50**
Periode	1	2	3	4
1				
2				
3				
4				
Bestellkostensatz = 40 GE				
Lagerkosten = 0,8 GE				

Tabelle 1: Beispiel - Ausgangssituation

[1] Harvey M. Wagner und Thomson M. Whitin (1958).
[2] In Anlehnung an: https://www.youtube.com/watch?v=vUQBz1phBsU, Aufgerufen am: 27.12.2015.

2.2.2 Schritt 1

Im ersten Schritt wird die Periode 1 betrachtet, das Ein-Perioden-Problem wird angesprochen. Diese besagt Minimalkosten = Rüstkosten.
In Tabelle 2 sind nun die Rüstkosten eingetragen, da es keine weitere Alternative gibt, sind dies die einzigsten Kosten, die anfallen. Es fallen an 40 GE für die Rüstkosten und 0 GE für die Lagerung.

Bedarf	**20**	**60**	**20**	**50**
Periode	1	2	3	4
1	40			
2				
3				
4				
Bestellkostensatz = 40 GE				
Lagerkosten = 0,8 GE				

Tabelle 2: Beispiel - Schritt 1

2.2.3 Schritt 2

Im zweiten Schritt werden neben dem Bedarf der Periode 1, auch die Bedarfsmenge der Periode 2 betrachtet. Hier wird ein Zwei-Perioden-Problem betrachtet. Es ergeben sich zwei Möglichkeiten, aus dem die kostengünstigere ausgewählt wird.
Die erste Möglichkeit besagt, dass sowohl Bedarf 1, als auch Bedarf 2 in Periode 1 produziert wird. Dabei fallen einmalige Rüstkosten von 40 GE an und Lagerungskosten für die 60 ME (Bedarf 2) für die zweite Periode.

1. Rüstkosten 1 + (Lagerkosten * Bedarf 2)
   40 + (0,8 * 60) = 88

In der zweiten Möglichkeit wird der Bedarf 1 in Periode 1 produziert und Bedarf 2 in Periode 2. Dabei fallen Rüstkosten in Periode 1 und 2 an.

2. Rüstkosten 1 + Rüstkosten 2
   40 + 40 = 80

Bedarf	**20**	**60**	**20**	**50**
Periode	1	2	3	4
1	40	88		
2		80		
3				
4				
Bestellkostensatz = 40 GE				
Lagerkosten = 0,8 GE				

Tabelle 3: Beispiel - Schritt 2

In der Tabelle 3 wird veranschaulicht, dass die zweite Möglichkeit kostengünstiger ist, als die Möglichkeit 1. Daher wird das Unternehmen sich für Möglichkeit 2 entscheiden.

2.2.4 Schritt 3

Das Drei-Perioden-Problem wird nun betrachtet. Hierbei ist zu beachten, wie in Schritt 2 entschieden worden ist. Bis zu diesem Zeitpunkt ist nun entschlossen den Bedarf 1 in Periode 1 und den Bedarf 2 in Periode 2 zu produzieren. Nun kann wieder zwischen zwei Möglichkeiten ausgewählt werden.

Der Bedarf 3 kann entweder in Periode 2 vorproduziert und im Lager verstaut oder in Periode 3 produziert werden. Möglichkeit 1 bringt Kosten in Höhe der Rüstkosten der Periode 1, Rüstkosten der Periode 2 und die Lagerungskosten des Bedarfs 3 in Periode 3 mit sich.

1. Rüstkosten 1 + Rüstkosten 2 + (Lagerkosten * Bedarf 3)

   40 + 40 + (0,8 * 20) = 96

Die Kosten für Möglichkeit 2 ergeben sich durch die drei Rüstkosten für die drei Perioden.

2. Rüstkosten 1 + Rüstkosten 2 + Rüstkosten 3

   40 + 40 + 40 = 120

Bedarf	**20**	**60**	**20**	**50**
Periode	1	2	3	4
1	40	88		
2		80	96	
3			120	
4				
Bestellkostensatz = 40 GE				
Lagerkosten = 0,8 GE				

Tabelle 4: Beispiel - Schritt 3

Die Entscheidung fällt auf die zweite Möglichkeit. Tabelle 4 veranschaulicht die Situation nach dem Drei-Perioden-Problem.

2.2.5 Schritt 4

Bis hierher wurde nun die kostengünstigste Produktion bis Periode 3 ermittelt. Schritt 4 bietet wieder zwei Möglichkeiten den Bedarf der Periode 4 zu decken. Einerseits kann wieder vorproduziert werden. Da wir den Bedarf der Periode 3 in Periode 2 mit produzieren, kann bei dieser Möglichkeit der Bedarf der Periode 4 ebenfalls in Periode 2 produziert werden. Das bedeutet die Kosten ergeben sich durch die Rüstkosten in Periode 1, die einmaligen Rüstkosten für den Bedarf in Periode 2, 3 und 4, den Lagerkosten für Periode 3 und den Lagerkosten für Periode 4. Hier ist zu beachten, dass der Bedarf für Periode 4 über zwei Perioden gelagert wird.

1. Rüstkosten1+Rüstkosten2+(Lagerkosten*Bedarf3)+(2*(Lagerkosten*Bedarf4))
   40 + 40 + (0,8 * 20) + (2*(0,8*50))= 176

Die Kosten für Möglichkeit 2 ergeben sich durch die bisherige Entscheidung und den Rüstkosten für die Periode 4.

2. Rüstkosten 1 + Rüstkosten 2 + (Lagerkosten * Bedarf 3) + Rüstkosten 4
   40 + 40 + (0,8 * 20) + 40 = 136

Bedarf	**20**	**60**	**20**	**50**
Periode	1	2	3	4
1	40	88		
2		80	96	176
3			120	
4				136
Bestellkostensatz = 40 GE				
Lagerkosten = 0,8 GE				

Tabelle 5: Beispiel - Schritt 4

Nun haben wir die optimalen Losgrößen für alle vier Perioden mit dem Wagner Whitin-Verfahren herausgefunden:

- Losgröße für Periode 1: 20 ME
- Losgröße für Periode 2: 80 ME
- Losgröße für Periode 3: 0 ME
- Losgröße für Periode 4: 50 ME

Wir haben Produktionskosten von 136 GE.

2.3 Vorgehensweise des Wagner-Whitin-LP 1.1

Öffnet man die Methode Wagner-Whitin LP 1.1 und gibt ein Modell ein, werden alle eingegebenen Werte durch das Betätigen der Operation „Optimale Loesung“ in ein LP Modell überführt.

In diese folgende Text-Datei werden die Daten eingefügt und so dem LP Solve zum Berechnen zur Verfügung gestellt:

```
0x1 + 0x2 + 0x3 + 0x4 + 0x5 + 0x6 + 0x7 + 0x8 + 0.1x9 + 0.1x10 + 0.1x11 + 0.1x12 + 0.1x13 +
0.1x14 + 0.1x15 + 0.1x16 + 20.0x17 + 20.0x18 + 20.0x19 + 20.0x20 + 20.0x21 + 20.0x22 +
20.0x23 + 20.0x24 + 2x25 + 2x26 + 2x27 + 2x28 + 2x29 + 2x30 + 2x31 + 2x32
1x1 + 0x2 + 0x3 + 0x4 + 0x5 + 0x6 + 0x7 + 0x8 + -1x9 + 0x10 + 0x11 + 0x12 + 0x13 + 0x14 +
0x15 + 0x16 + 0x17 + 0x18 + 0x19 + 0x20 + 0x21 + 0x22 + 0x23 + 0x24 + 1x25 + 0x26 + 0x27 +
0x28 + 0x29 + 0x30 + 0x31 + 0x32 = 60
1x1 + 1x2 + 0x3 + 0x4 + 0x5 + 0x6 + 0x7 + 0x8 + 0x9 + -1x10 + 0x11 + 0x12 + 0x13 + 0x14 +
0x15 + 0x16 + 0x17 + 0x18 + 0x19 + 0x20 + 0x21 + 0x22 + 0x23 + 0x24 + 0x25 + 1x26 + 0x27 +
0x28 + 0x29 + 0x30 + 0x31 + 0x32 = 150
1x1 + 1x2 + 1x3 + 0x4 + 0x5 + 0x6 + 0x7 + 0x8 + 0x9 + 0x10 + -1x11 + 0x12 + 0x13 + 0x14 +
0x15 + 0x16 + 0x17 + 0x18 + 0x19 + 0x20 + 0x21 + 0x22 + 0x23 + 0x24 + 0x25 + 0x26 + 1x27 +
0x28 + 0x29 + 0x30 + 0x31 + 0x32 = 290
1x1 + 1x2 + 1x3 + 1x4 + 0x5 + 0x6 + 0x7 + 0x8 + 0x9 + 0x10 + 0x11 + -1x12 + 0x13 + 0x14 +
0x15 + 0x16 + 0x17 + 0x18 + 0x19 + 0x20 + 0x21 + 0x22 + 0x23 + 0x24 + 0x25 + 0x26 + 0x27 +
1x28 + 0x29 + 0x30 + 0x31 + 0x32 = 310
1x1 + 1x2 + 1x3 + 1x4 + 1x5 + 0x6 + 0x7 + 0x8 + 0x9 + 0x10 + 0x11 + 0x12 + -1x13 + 0x14 +
0x15 + 0x16 + 0x17 + 0x18 + 0x19 + 0x20 + 0x21 + 0x22 + 0x23 + 0x24 + 0x25 + 0x26 + 0x27 +
0x28 + 1x29 + 0x30 + 0x31 + 0x32 = 360
1x1 + 1x2 + 1x3 + 1x4 + 1x5 + 1x6 + 0x7 + 0x8 + 0x9 + 0x10 + 0x11 + 0x12 + 0x13 + -1x14 +
0x15 + 0x16 + 0x17 + 0x18 + 0x19 + 0x20 + 0x21 + 0x22 + 0x23 + 0x24 + 0x25 + 0x26 + 0x27 +
0x28 + 0x29 + 1x30 + 0x31 + 0x32 = 540
1x1 + 1x2 + 1x3 + 1x4 + 1x5 + 1x6 + 1x7 + 0x8 + 0x9 + 0x10 + 0x11 + 0x12 + 0x13 + 0x14 + -
1x15 + 0x16 + 0x17 + 0x18 + 0x19 + 0x20 + 0x21 + 0x22 + 0x23 + 0x24 + 0x25 + 0x26 + 0x27 +
0x28 + 0x29 + 0x30 + 1x31 + 0x32 = 660
1x1 + 1x2 + 1x3 + 1x4 + 1x5 + 1x6 + 1x7 + 1x8 + 0x9 + 0x10 + 0x11 + 0x12 + 0x13 + 0x14 +
0x15 + -1x16 + 0x17 + 0x18 + 0x19 + 0x20 + 0x21 + 0x22 + 0x23 + 0x24 + 0x25 + 0x26 + 0x27 +
0x28 + 0x29 + 0x30 + 0x31 + 1x32 = 760
1x1 + 0x2 + 0x3 + 0x4 + 0x5 + 0x6 + 0x7 + 0x8 + 0x9 + 0x10 + 0x11 + 0x12 + 0x13 + 0x14 +
0x15 + 0x16 + -1000x17 + 0x18 + 0x19 + 0x20 + 0x21 + 0x22 + 0x23 + 0x24 + 0x25 + 0x26 +
0x27 + 0x28 + 0x29 + 0x30 + 0x31 + 0x32 < 0
0x1 + 1x2 + 0x3 + 0x4 + 0x5 + 0x6 + 0x7 + 0x8 + 0x9 + 0x10 + 0x11 + 0x12 + 0x13 + 0x14 +
0x15 + 0x16 + 0x17 + -1000x18 + 0x19 + 0x20 + 0x21 + 0x22 + 0x23 + 0x24 + 0x25 + 0x26 +
0x27 + 0x28 + 0x29 + 0x30 + 0x31 + 0x32 < 0
0x1 + 0x2 + 1x3 + 0x4 + 0x5 + 0x6 + 0x7 + 0x8 + 0x9 + 0x10 + 0x11 + 0x12 + 0x13 + 0x14 +
0x15 + 0x16 + 0x17 + 0x18 + -100000x19 + 0x20 + 0x21 + 0x22 + 0x23 + 0x24 +
0x25 + 0x26 + 0x27 + 0x28 + 0x29 + 0x30 + 0x31 + 0x32 < 0
0x1 + 0x2 + 0x3 + 1x4 + 0x5 + 0x6 + 0x7 + 0x8 + 0x9 + 0x10 + 0x11 + 0x12 + 0x13 + 0x14 +
0x15 + 0x16 + 0x17 + 0x18 + 0x19 + -100000x20 + 0x21 + 0x22 + 0x23 + 0x24 + 0x25 + 0x26 +
0x27 + 0x28 + 0x29 + 0x30 + 0x31 + 0x32 < 0
0x1 + 0x2 + 0x3 + 0x4 + 1x5 + 0x6 + 0x7 + 0x8 + 0x9 + 0x10 + 0x11 + 0x12 + 0x13 + 0x14 +
0x15 + 0x16 + 0x17 + 0x18 + 0x19 + 0x20 + -100000x21 + 0x22 + 0x23 + 0x24 + 0x25 + 0x26 +
0x27 + 0x28 + 0x29 + 0x30 + 0x31 + 0x32 < 0
0x1 + 0x2 + 0x3 + 0x4 + 0x5 + 1x6 + 0x7 + 0x8 + 0x9 + 0x10 + 0x11 + 0x12 + 0x13 + 0x14 +
0x15 + 0x16 + 0x17 + 0x18 + 0x19 + 0x20 + 0x21 + -100000x22 + 0x23 + 0x24 + 0x25 + 0x26 +
0x27 + 0x28 + 0x29 + 0x30 + 0x31 + 0x32 < 0
0x1 + 0x2 + 0x3 + 0x4 + 0x5 + 0x6 + 1x7 + 0x8 + 0x9 + 0x10 + 0x11 + 0x12 + 0x13 + 0x14 +
0x15 + 0x16 + 0x17 + 0x18 + 0x19 + 0x20 + 0x21 + 0x22 + -100000x23 + 0x24 + 0x25 + 0x26 +
0x27 + 0x28 + 0x29 + 0x30 + 0x31 + 0x32 < 0
0x1 + 0x2 + 0x3 + 0x4 + 0x5 + 0x6 + 0x7 + 1x8 + 0x9 + 0x10 + 0x11 + 0x12 + 0x13 + 0x14 +
0x15 + 0x16 + 0x17 + 0x18 + 0x19 + 0x20 + 0x21 + 0x22 + 0x23 + -100000x24 + 0x25 + 0x26 +
0x27 + 0x28 + 0x29 + 0x30 + 0x31 + 0x32 < 0
```

Abbildung 1: Text-Datei für LP Solve

Im nächsten Schritt berechnet der LP Solve das LP-Modell und schickt das Ergebnis zurück an die Methode Wagner-Whitin. Dieses Ergebnis wird geparsed und für die Ausgabe ausgelesen und entsprechend der Oberfläche veranschaulicht.

3 Projektarbeit

Das angegangene Projekt gliedert sich in drei Hauptaufgaben. An erster Stelle soll der Rechenfehler behoben werden. Dazu muss die passende Stelle im Java-Code gefunden werden und der Fehler identifiziert und anschließend verbessert werden.

Die nicht vorhandene Hilfefunktion stellt ein weiteres Problem der Methode dar. Zunächst müssen wichtige Funktionen beschrieben und ausformuliert werden. Die Funktionsweise des Algorithmus wird dabei auch berücksichtigt und soll verständlich beschrieben werden.

Da das Ergebnis zukünftig nicht mehr über einen Text-Editor ausgegeben werden soll, sondern in der Methode selber, muss eine entsprechende Schnittstelle hergestellt werden. Dabei wird das im MPS-Format zugesandte Ergebnis ausgelesen und entsprechend der Ausgabe-Oberfläche zugewiesen und ausgegeben.

3.1 Fehlerstatus

Das in OR-Alpha bestehende Wagner-Whitin-LP hat einige Mängel vorzuweisen. Anbei eine Auflistung dieser:

- Der Pfad ist fix und eine dynamische Pfadanpassung wird nicht gewährleistet.
- Unvollständige Dokumentationen hinterlegt.
- Unzureichende Hilfestellung in der Methode.
- Ergebnisausgabe über Editor muss ausgewählt werden.
- Keine automatische Nummerierung.
- Rechenfehler bei Bedarfsmengen größer 1000.

Einige dieser Fehler werden im Rahmen dieses Projektes analysiert, angegangen und behoben.

3.2 Fehlerbehebung

3.2.1 Ergebnisausgabe

Der Wagner Whitin benötigt für die Lösungsausgabe den Editor, um die Daten des LP-Solves im MPS-Format anzeigen zu können. Die Losgrößen sind nicht leicht zu erkennen und nähere Informationen zu den Kosten werden nicht angezeigt. In Abbildung 1 wird die ursprüngliche Lösungsausgabe angezeigt:

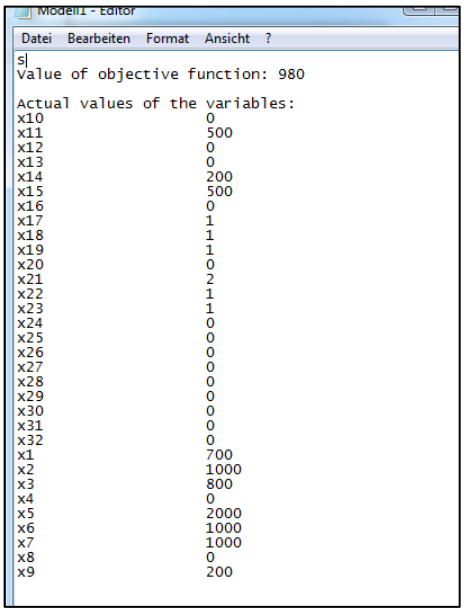


Abbildung 2: Ausgabe der Lösung über den Editor

Um eine Lösungsausgabe in der Methode selbst zu ermöglichen, muss einerseits die Ausgabe im MPS-Format für die Methode übersetzt werden und der Editoraufruf unterbunden werden.

Die in der Klasse LPSolver enthaltene try-catch-Methode, die einen LP-Solver-Aufruf startet und gleichzeitig versucht den Editor aufzurufen wurde entsprechend angepasst.

```
try
{ // Stapeldatei erzeugen fuer Solver Aufruf, da direkter Aufruf nicht
  // funktioniert..!
  FileOutputStream fileOutStream = new FileOutputStream(executeBatch);
  PrintWriter out = new PrintWriter(fileOutStream);
  out.println("@echo of");
  out.println("echo Automatisch erzeugte Stapeldatei startet den LP-Solver");
  out.println(executeSolver);

  //Auskommentiert damit die Lösung nicht erst über ein Texteditor geöffnet wird
  //out.println(parameter2);
  out.println("exit");
  out.close();
}
catch (Exception e)
{
  _errMsg = "Konnte Start.bat zum starten des Sovers nicht schreiben";
  return -1;
}
```

Abbildung 3: Editoraufruf auskommentiert

Des Weiteren wurde eine Schleife eingebaut, um die Datei in MPS-Format auszulesen und die Auswertung starten zu können.

```
    //Iteration über den "neuen" Ausgabestring: Actual values of the variables:
    for (int i = 0; i < 35; i++) {
        _token = getNextElement();
    }
    return macheAuswertung();
}
```

Abbildung 4: Aufruf der macheAuswertung() Methode

Diese Methode `private int macheAuswertung()` befindet sich in der Klasse `LPSolveParser` und ist für das Auslesen der LP-Solver Ausgabe zuständig.

3.2.2 Rechenfehler

In der Klasse `plo_Eingabemaske` gibt es eine Methode **`public void`** `produceMatrix()`, welche den LP-Ansatz des Modells erstellen soll. Die Methode arbeitet mit einer M-Variable mit einer ursprünglich Höhe von 1'000. Diese Variable wurde von uns auf 100'000 erhöht. In den Abbildungen 1 und 2 wird der Unterschied nochmals aufgezeigt:

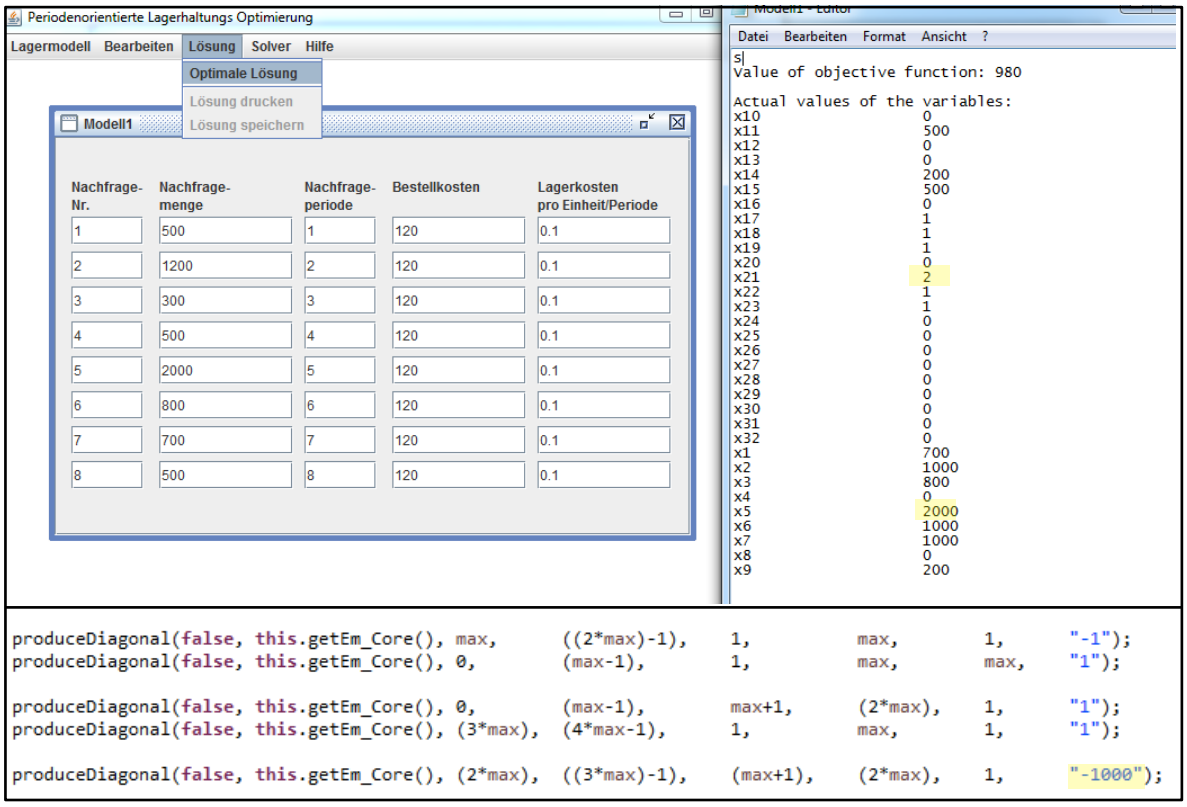


Abbildung 5: Falsche Bestellmenge

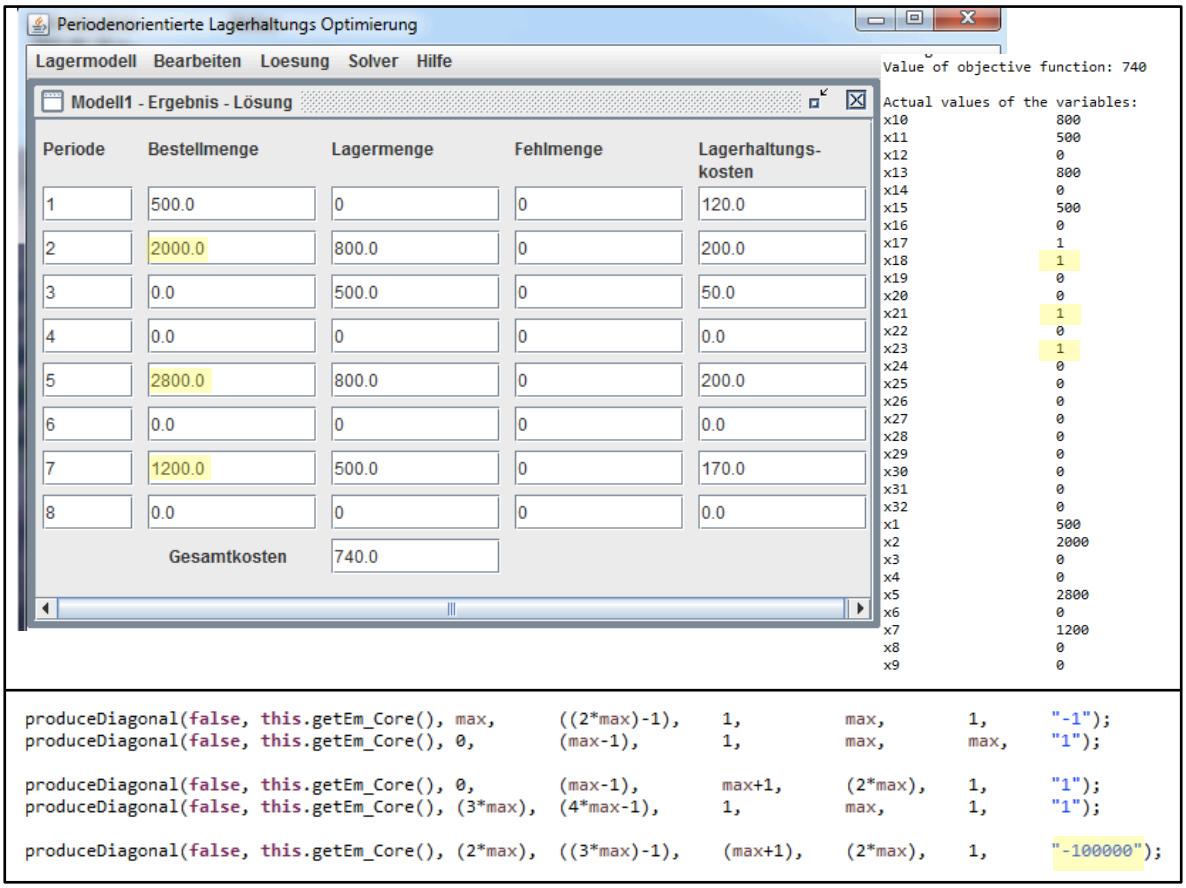


Abbildung 6: Verbesserte Bestellmengenangaben

Wie man in Abbildung erkennen kann, ist der Fehler nun behoben. Bei Bestellmengen größer 1000 wird einmal und nicht doppelt bestellt.

3.2.3 Hilfefunktion

In Abbildung 1 ist die ursprüngliche Hilfefunktion für Wagner Whitin dargestellt.

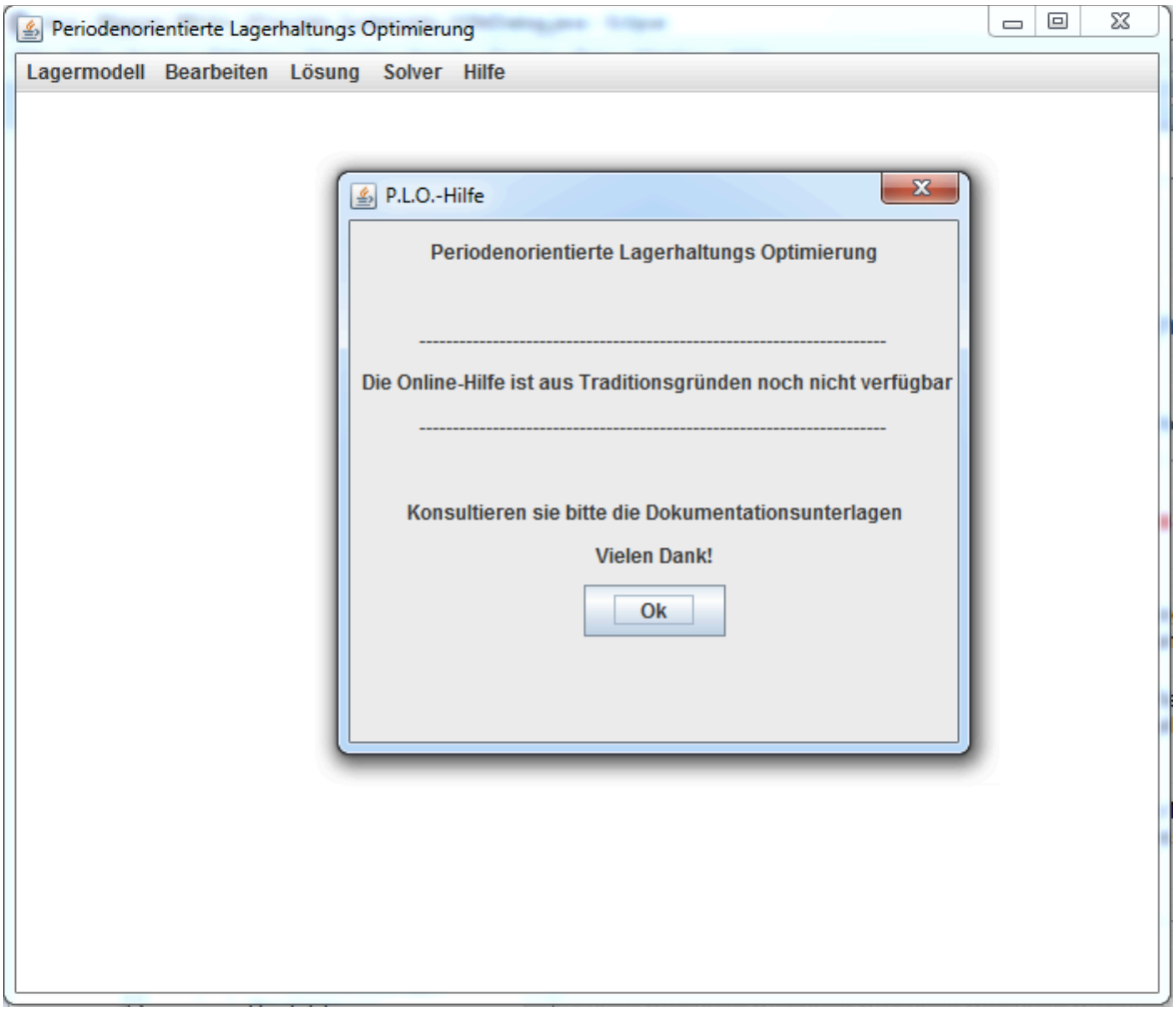


Abbildung 7: P.L.O.-Hilfe – ursprünglich

Da diese keine Hilfestellung leistet, war es unsere Aufgabe die Hilfe-Funktion zu editieren. Nach einigen Anpassungen im Quellcode (siehe Abbildung 4), ist eine Vorgehensweise beschrieben, die ein reibungsloses Benutzen der Software gewährleisten kann.

```
// Konstruktor
public plo_HilfeDialog() {
    // Initialisieren des Referenzobjekts fuer dispose()-Aufrufe
    final plo_HilfeDialog ref = this;

    hilfeFrame = new JInternalFrame();
    this.setTitle("P.L.O.-Hilfe");
    hilfeFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

    hilfePanel      = new JPanel();
    hilfeLabel1     = new JLabel("   Periodenorientierte Lagerhaltungs Optimierung");
    hilfeLabel2     = new JLabel(" ");
    hilfeLabel3     = new JLabel("-------------------------------------------------------------------------");
    hilfeLabel4     = new JLabel("   Neue Modelle erstellen");
    hilfeLabel5     = new JLabel("-------------------------------------------------------------------------");
    hilfeLabel7     = new JLabel("   1.   Klicken Sie auf Lagermodell --> Neues Modell.");
    hilfeLabel8     = new JLabel("    2.   Nun geben Sie die Anzahl der Perioden ein.");
    hilfeLabel9     = new JLabel("   3.   Im nächsten Schritt tippen Sie alle Daten ein.");
    hilfeLabel10    = new JLabel("   4.   Jetzt klicken Sie auf Loesung --> Optimale Loesung.");
    hilfeLabel11    = new JLabel("   5.   Die Lösung mit den optimalen Lossgroessen erscheint.");
    hilfeLabel12    = new JLabel(" ");
    hilfeLabel13    = new JLabel("-------------------------------------------------------------------------");
    hilfeLabel14    = new JLabel("   Modelle speichern");
    hilfeLabel15    = new JLabel("-------------------------------------------------------------------------");
    hilfeLabel16    = new JLabel("   1.   Klicken Sie auf Lagermodell --> Modell speichern.");
    hilfeLabel17    = new JLabel("   2.   Nun waehlen Sie einen Speicherort und bestaetigen mit OK.");
    hilfeLabel18    = new JLabel(" ");
    hilfeLabel19    = new JLabel("-------------------------------------------------------------------------");
    hilfeLabel20    = new JLabel("   Modelle laden");
    hilfeLabel21    = new JLabel("-------------------------------------------------------------------------");
    hilfeLabel22    = new JLabel("   1.   Gespeicherte Modelle können geladen werden, indem Sie auf ");
    hilfeLabel23    = new JLabel("          Lagermodell --> Modell laden klicken.");
    hilfeLabel24    = new JLabel("   2.   Waehlen Sie das zu oeffnende Modell und bestaetigen mit OK.");
    hilfeLabel25    = new JLabel(" ");
    hilfeLabel26    = new JLabel("   Vielen Dank!   ");

    jb_Ok           = new JButton("     Ok     ");
    jb_Ok.setSize(150, 50);

    hilfeGridBagLayout = new GridBagLayout();
    hilfeGridBagConstraints = new GridBagConstraints();

    hilfePanel.setLayout(hilfeGridBagLayout);
```

Abbildung 8: Anpassung der Hilfe-Funktion im Programcode

In Abbildung 8 ist dargestellt, wie die Hilfe-Funktion nach der Anpassung aussieht.

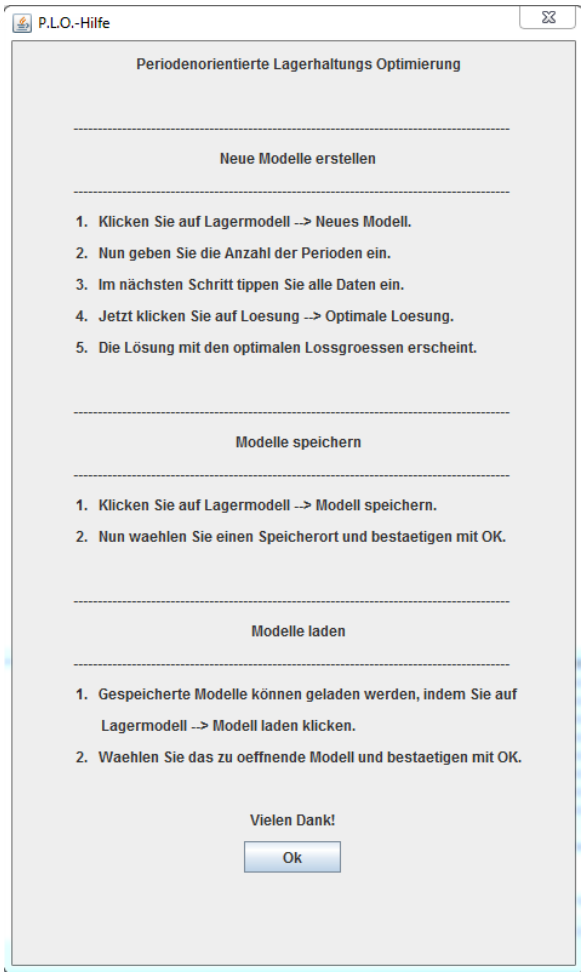


Abbildung 9: Neue Hilfe-Funktion

3.2.4 Automatische Nummerierung der Perioden

Beim Erstellen neuer Modelle, ist ursprünglich die Spalte der Nachfrage-Nr. und der Nachfrageperiode auf „0“ gesetzt. Es war notwendig, die Nummerierung manuell durchzuführen. Nach einer Anpassung in der Klasse `plo_Eingabemaske` sind nun default-Werte eingebettet. Die automatische Belegung der Nachfragenummer und Nachfrageperiode ist nun möglich. In den folgenden Abbildungen ist zu sehen, an welcher Stelle im Code eine Anpassung gemacht wurde und wie die Eingabemaske nach Erstellen eines Modells mit 8 Perioden aussieht.

```
int i;
for(i = 2; i < (anzahlNachfragen+2); i++)                    //Hinzufuegen der leeren Nachfragen in der
{                                                            //Anzahl, die vom Benutzer eingegeben worden ist.
    em_nach = new nachfrage();                               //
    dTemp = new Double(this.getRoot().getDefaultBestellkosten());
    em_nach.setTf_BestellkostenText(dTemp.toString());
    dTemp = new Double(this.getRoot().getDefaultLagerkosten());
    em_nach.setTf_LagerkostenText(dTemp.toString());
    //Setzen der Nachfragennummer und Periode aufsteigend
    em_nach.setTf_NummerText(String.valueOf(i-1));
    em_nach.setTf_PeriodeText(String.valueOf(i-1));

    this.setEm_NachfragenListe((i-2), em_nach);
```

Abbildung 10: Anpassung im Code zur automatischen Nummerierung

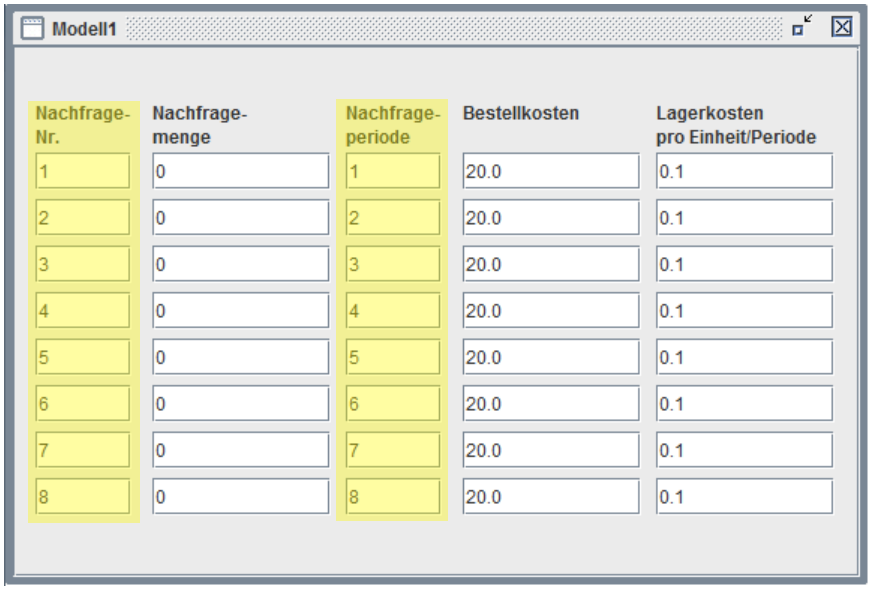


Abbildung 11: Automatische Nummerierung im Modell

3.2.5 Automatische Pfadanpassung

Bei erstem Programmstart müssen folgenden Angaben zu den Verzeichnissen gemacht werden:

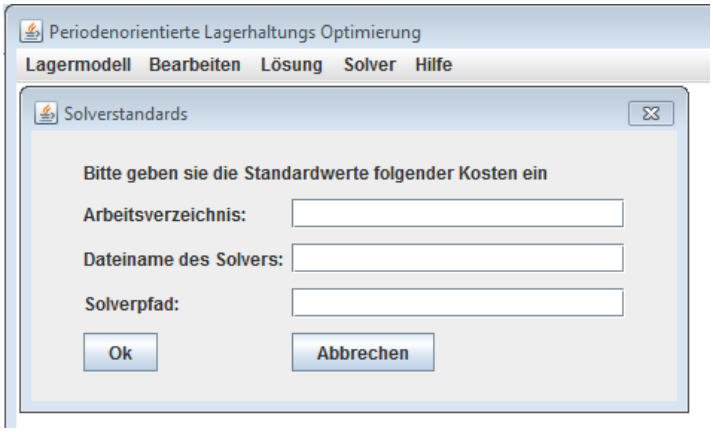


Abbildung 11: Pfadanpassung

Diese Daten werden dann in einer *solverini.txt* Datei (siehe Abbildung 13) gespeichert und bei weiteren Programmstarts ausgelesen.

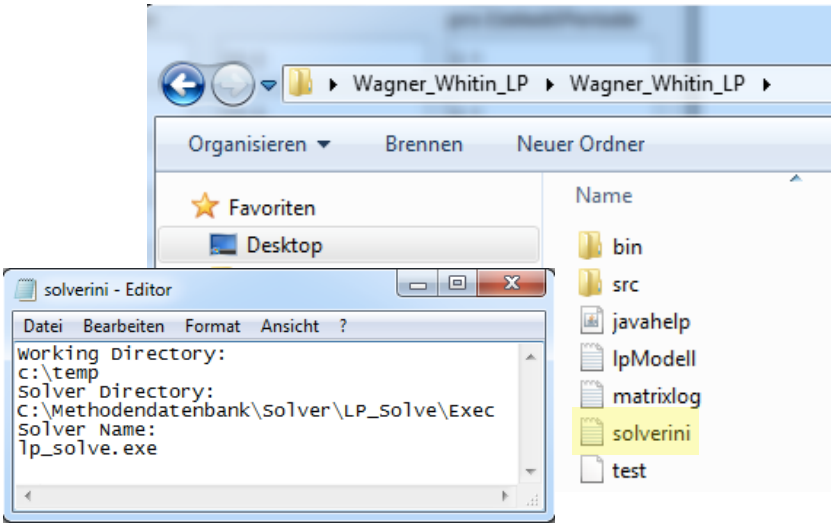


Abbildung 12: solverini.txt - Datei

Um eine automatisierte Einbettung dieser Pfaddaten zu generieren, wurden Änderungen in der Klasse `plo_Eingabemaske` vorgenommen. Siehe folgende Abbildung:

```
try                                                         //Anlegen eines Stromobjekts fuer
{                                                           //die Datei "solverini.txt"
    BufferedInputStream bis = new BufferedInputStream (new FileInputStream("solverini.txt"));
}
catch (FileNotFoundException fnfe){

    //Falls keine solverini.txt exestiert, wird eine mit vordefenierten Werten erzeugt
    //und die Methode zum einlesen wird erneut durchlaufen
    try {

        File file = new File("solverini.txt");
        FileWriter writer = new FileWriter(file);
        // Text wird in den Stream geschrieben
        writer.write("Working Directory:");
        // Platformunabhängiger Zeilenumbruch wird in den Stream geschrieben
        writer.write(System.getProperty("line.separator"));
        // Text wird in den Stream geschrieben
        writer.write("c:\\temp");
        // Platformunabhängiger Zeilenumbruch wird in den Stream geschrieben
        writer.write(System.getProperty("line.separator"));
        // Text wird in den Stream geschrieben
        writer.write("Solver Directory:");
        // Platformunabhängiger Zeilenumbruch wird in den Stream geschrieben
        writer.write(System.getProperty("line.separator"));
        // Text wird in den Stream geschrieben
        writer.write("C:\\Methodendatenbank\\Solver\\LP_Solve\\Exec");
        // Platformunabhängiger Zeilenumbruch wird in den Stream geschrieben
        writer.write(System.getProperty("line.separator"));
        // Text wird in den Stream geschrieben
        writer.write("Solver Name:");
        // Platformunabhängiger Zeilenumbruch wird in den Stream geschrieben
        writer.write(System.getProperty("line.separator"));
        // Text wird in den Stream geschrieben
        writer.write("lp_solve.exe");
        writer.flush();

        solverIniEinlesen();

    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

Abbildung 13: Anpassung im Programmcode zur Pfadanpassung

Der try-catch-Block ist für das Erstellen der txt-Datei und das auslesen dieser verantwortlich. Es wird erst überprüft, ob eine *solverini.txt* Datei besteht. Falls nicht, wird die Datei im Hintergrund erstellt und abgelegt. Der Vorteil ist, dass keine manuelle Eingabe der Pfade nötig ist.

3.2.6 Beheben der Schreibfehler

In der Menü-Leiste Lagermodell sind Rechtschreibfehler (siehe nächste Abbildung), die wir verbessert haben:

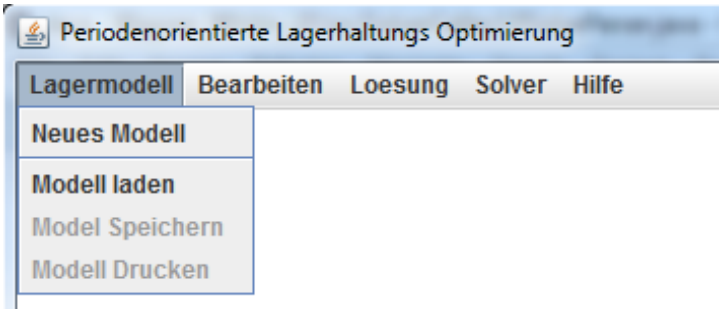


Abbildung 14: Rechtschreibfehler in der Navigationsleiste

In der Klasse `plo_MenuBar` wurden entsprechende Anpassungen gemacht.

```
m_Lagermodell = new JMenu("Lagermodell");                    //Initialisieren der Menues
m_Bearbeiten = new JMenu("Bearbeiten");
m_Loesung = new JMenu("Lösung");
m_Solver = new JMenu("Solver");
m_Hilfe = new JMenu("Hilfe");

mi_NeuesModell = new JMenuItem("Neues Modell");              //Initialisieren der Menuepunkte
mi_ModellLaden = new JMenuItem("Modell laden");
mi_ModellSpeichern = new JMenuItem("Model speichern");
mi_ModellSpeichern.setEnabled(false);
mi_ModellDrucken = new JMenuItem("Modell drucken");
mi_ModellDrucken.setEnabled(false);

mi_NachfrageEinfuegen = new JMenuItem("Nachfrage einfügen");
mi_NachfrageEinfuegen.setEnabled(false);
mi_NachfrageEntfernen = new JMenuItem("Nachfrage entfernen");
mi_NachfrageEntfernen.setEnabled(false);
mi_AllesAendern = new JMenuItem("Alles ändern");
mi_AllesAendern.setEnabled(false);
mi_Defaultkosten = new JMenuItem("Defaultkosten");

mi_OptimaleLoesung = new JMenuItem("Optimale Lösung berechnen");
mi_OptimaleLoesung.setEnabled(false);
mi_LoesungDrucken = new JMenuItem("Lösung drucken");
mi_LoesungDrucken.setEnabled(false);
mi_LoesungSpeichern = new JMenuItem("Lösung speichern");
mi_LoesungSpeichern.setEnabled(false);

mi_SolverConfigAendern = new JMenuItem("Solver Konfiguration ändern");
mi_SolverConfigAendern.setEnabled(true);

mi_PloHilfe = new JMenuItem("PLO-Hilfe");
mi_ueber = new JMenuItem("Über");
```

Abbildung 15: Behobene Rechtschreibfehler im Programmcode

3.2.7 Anpassung der „Über“-Daten

Die über die Leiste Hilfe → Über zu findenden Informationen wurden entsprechend der neuen Version angepasst. In der folgenden Abbildung ist die aktuelle Klasse `plo_ueberDialog` zu sehen.

```
//*** Konstruktor ***-----------------------------------------------
    public plo_ueberDialog()
    {
        final plo_ueberDialog ref = this;                          //Erstellen eines Referen

        überFrame = new JInternalFrame();
        this.setTitle("Über P.L.O.");
        überFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

        überPanel = new JPanel();
        überLabel1 = new JLabel("Periodenorientierte Lagerhaltungs Optimierung");
        überLabel2 = new JLabel("(Version 1.1)");
        überLabel3 = new JLabel("(c) 2015/20016 ");
        überLabel4 = new JLabel("-----------------------------------------------------------------
        überLabel5 = new JLabel("Programm von:");
        überLabel6 = new JLabel("Eugen Gering");
        überLabel7 = new JLabel("Melisa Gündüz");
        überLabel8 = new JLabel("Francis Göltner");
        überLabel9 = new JLabel("Helmut Lindinger");
        überLabel10 = new JLabel("Bernd Saile");
        jb_Ok = new JButton("     Ok     ");
        jb_Ok.setSize(150,25);
        überGridBagLayout = new GridBagLayout();
        überGridBagConstraints = new GridBagConstraints();
        überPanel.setLayout(überGridBagLayout);

        this.buildConstraints(überGridBagConstraints, überLabel1, 0, 0, 1, 1, GridBagConstrai
        this.buildConstraints(überGridBagConstraints, überLabel2, 0, 1, 1, 1, GridBagConstrai
        this.buildConstraints(überGridBagConstraints, überLabel3, 0, 2, 1, 1, GridBagConstrai
        this.buildConstraints(überGridBagConstraints, überLabel4, 0, 3, 1, 1, GridBagConstrai
        this.buildConstraints(überGridBagConstraints, überLabel5, 0, 4, 1, 1, GridBagConstrai
        this.buildConstraints(überGridBagConstraints, überLabel6, 0, 5, 1, 1, GridBagConstrai
        this.buildConstraints(überGridBagConstraints, überLabel7, 0, 6, 1, 1, GridBagConstrai
        this.buildConstraints(überGridBagConstraints, überLabel8, 0, 7, 1, 1, GridBagConstrai
        this.buildConstraints(überGridBagConstraints, überLabel9, 0, 8, 1, 1, GridBagConstrai
        this.buildConstraints(überGridBagConstraints, überLabel10, 0, 9, 1, 1, GridBagConstra
        this.buildConstraints(überGridBagConstraints, jb_Ok, 0, 12, 1, 1, GridBagConstraints.
```

Abbildung 16: Angepasstes Über-Dialog im Programmcode

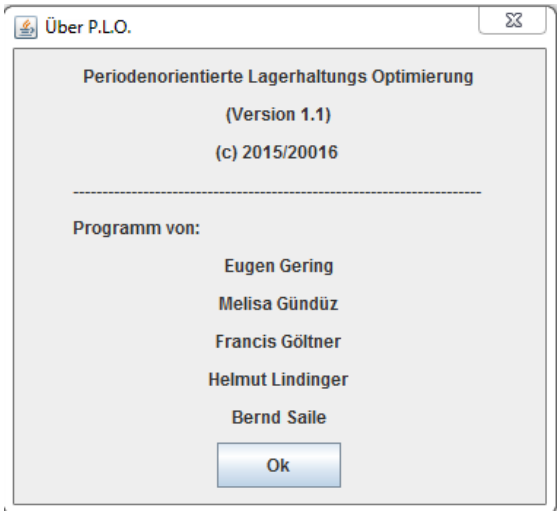


Abbildung 17: Neues Über-Dialog

3.3 Gegenüberstellung des Commitment-Inhalts und des Umgesetzten

Aufgabenbereich	Commitment-Inhalt	Umsetzung
Hilfefunktion	Die Hilfe-Funktion soll editiert werden.	✓
Lösungsausgabe	Das Ergebnis soll nicht im Editor, sondern im Programm selbst ausgegeben werden. (Layout)	✓
Rechenfehler	Bei Bestellmengen höher als 1000 werden doppelte Bestellkosten bestimmt. Herausfinden wieso das so ist und anpassen.	✓
⊕ **LOGO**		
⊕ **Automatisierte Pfadanpassung**		
⊕ **Vollständige Dokumentation**		
⊕ **Automatische Nummerierung der Perioden**		
⊕ **Beheben der Rechtschreibfehler**		

Tabelle 6: Gegenüberstellung

Quellenverzeichnis

[1] Harvey M. Wagner und Thomson M. Whitin (1958) Dynamic Version of the Economic Lot Size Model. In: Management Science 5(1958)1, 89-96.

[2] https://www.youtube.com/watch?v=vUQBz1phBsU, Angeboten von ingenieur-kurse, Aufgerufen am: 27.12.2015.