Wagner-Whitin

Anwendung Linearer Optimierung

Julia Martin
Viktoria Maier

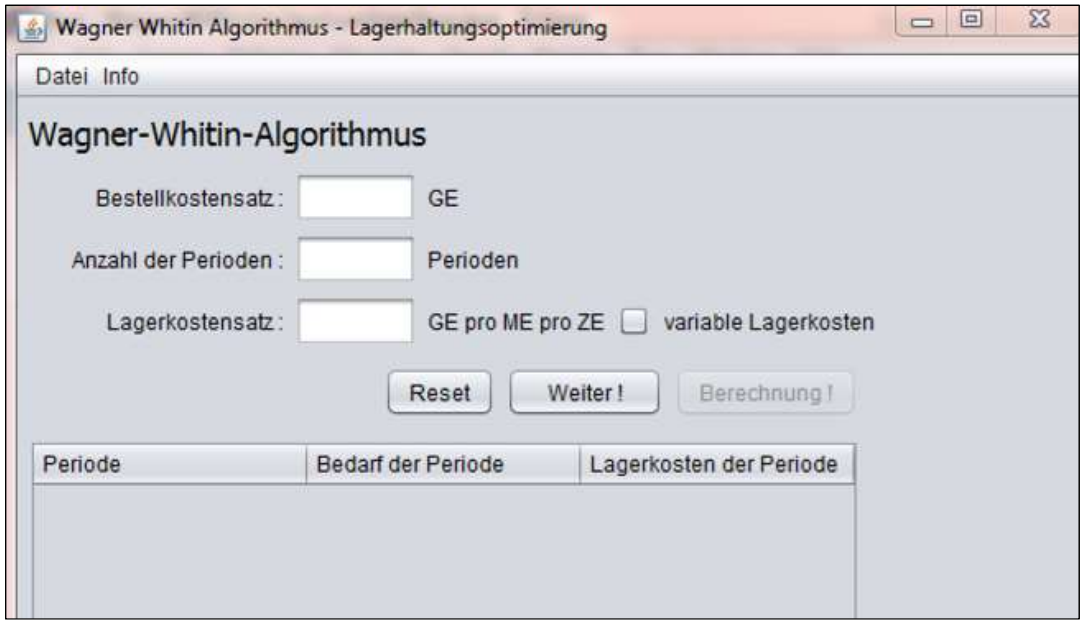
Agenda

- Vorstellung des Tools
- Funktionsumfang der Methode
- Optimierung
  - Anforderungen
  - Umsetzung
- Wagner Whitin-Verfahren
- Beispiel Wagner-Whitin-Verfahren
- Live-Demo
- Verbesserungsvorschlag / Fazit

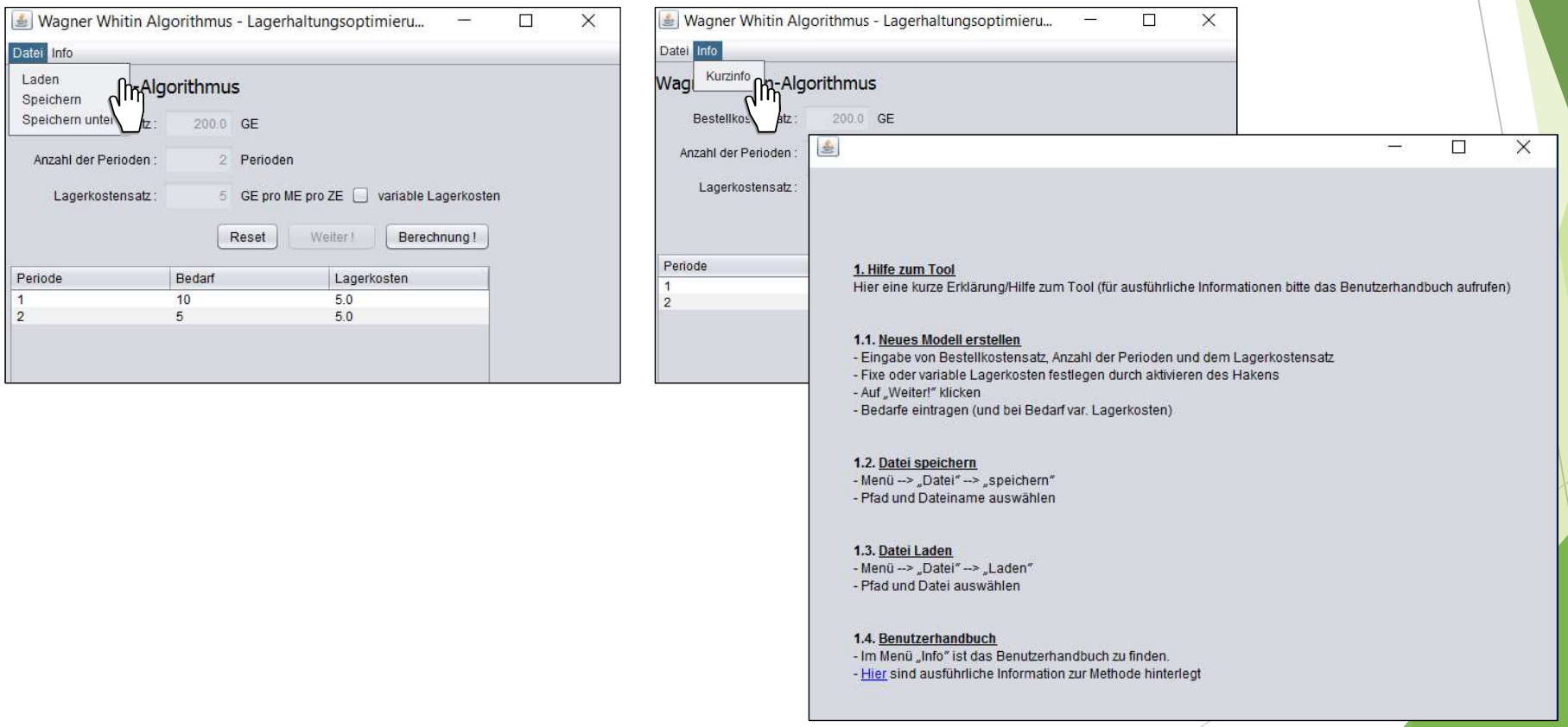
Vorstellung Tool: Wagner-Whitin

- Wagner-Whitin-Tool basiert auf Wagner-Whitin Verfahren
- Dieser errechnet die optimale Bestellmenge und den optimalen Bestellzeitpunkt unter der Berücksichtigung sich immer wieder ändernden Nachfrage
- Die vorhandene Kapazität wird hierbei nicht berücksichtigt

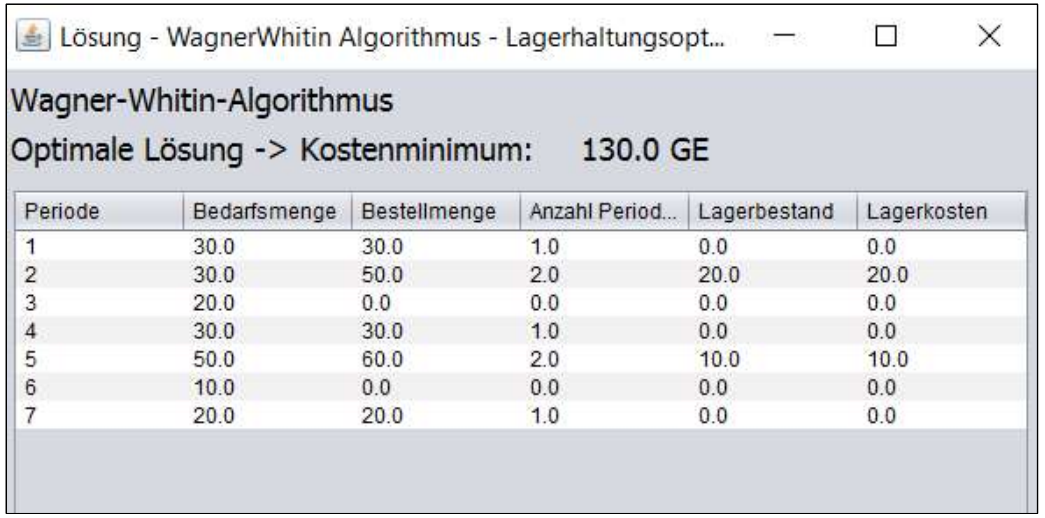
Vorstellung Tool: Wagner-Whitin



Vorstellung Tool: Wagner-Whitin



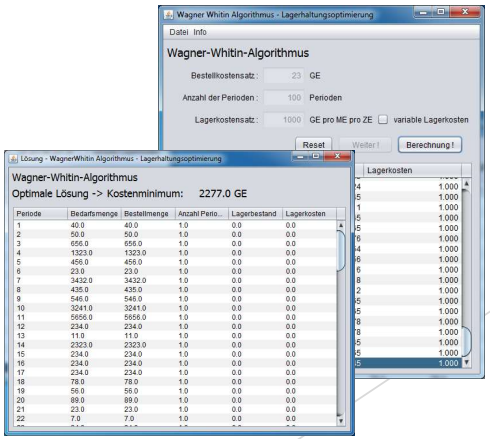
Vorstellung Tool: Wagner-Whitin



Periode	Bedarfsmenge	Bestellmenge	Anzahl Period...	Lagerbestand	Lagerkosten
1	30.0	30.0	1.0	0.0	0.0
2	30.0	50.0	2.0	20.0	20.0
3	20.0	0.0	0.0	0.0	0.0
4	30.0	30.0	1.0	0.0	0.0
5	50.0	60.0	2.0	10.0	10.0
6	10.0	0.0	0.0	0.0	0.0
7	20.0	20.0	1.0	0.0	0.0

Optimierung: Anforderungen

- Funktionale Anforderungen:
  - ✓ Implementierung der Speicher-Funktion durch einen Speicher-Button unter dem Menüpunkt „Datei“.
  - ✓ Ausschließlich WWA-Dateien auswählbar
  - ✓ Tests der Methode
- Nichtfunktionale Anforderungen:
  - ✓ Rechtschreibfehler beseitigen



Optimierung: Umsetzung – Tests

- Preview-Review-Test der Methode
  - Verschiedene Testszenario
    - Berechnung
    - Kurzinfo
    - Abspeichern und Laden
    - Reset-Funktion

Optimierung: Umsetzung – WWA-Dateien

- Ergänzung der Klasse SpeichernOeffnen.java

```
public class MyFilter extends FileFilter{
    private String endung =".wwa";

    public MyFilter(String endung){
        this.endung = endung;
    }

    @Override
    public boolean accept(File chooser){
        if(chooser == null){
            return false;
        }else
        //Ordner anzeigen
        if(chooser.isDirectory()){
            return true;
        }
        //true, wenn File gewuenschte Endung besitzt
        else{
            return chooser.getName().toLowerCase().endsWith(endung);
        }
    }

    @Override
    public String getDescription(){
        return endung;
    }
}
```

- innere Klasse „MyFilter“ erbt von Java-Klasse FileFilter
- FileFilter verwendet JFileChooser
- boolean-Methode prüft ob WWA-Datei true/false

Optimierung:
Umsetzung – Speichern-Funktion

- Ausgangspunkt:
  - Klasse userInterface.java
    - Menüpunkt: *Laden* mit der Methode actionPerformed(ActionEvent load)
    - Menüpunkt: *Speichern* mit der Methode actionPerformed(ActionEvent save)
- Code erweitert:
  - Klasse userInterface.java
    - Menüpunkt: Speichern unter mit der Methode actionPerformed(ActionEvent save_as)

- Interface: ActionListener
  Methode: void actionPerfromed(Action Event)
- Methode: addActionListener() mit dem Listener verbinden.

Optimierung:
Umsetzung – Speichern-Funktion

- Erweiterung der GUI mit dem Button „Speichern unter“

```
mntmSpeichern_Unter = new JMenuItem("Speichern unter");
mntmSpeichern_Unter.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent save_as) {
```

Klasse:
userInterface.java
Methode:
actionPerformed(ActionEvent save_as)

Klasse:
SpeichernOeffnen.java
Methode:
fileChooserDialog

Optimierung:
Umsetzung - Speichern-Funktion

- Button „Speichern“

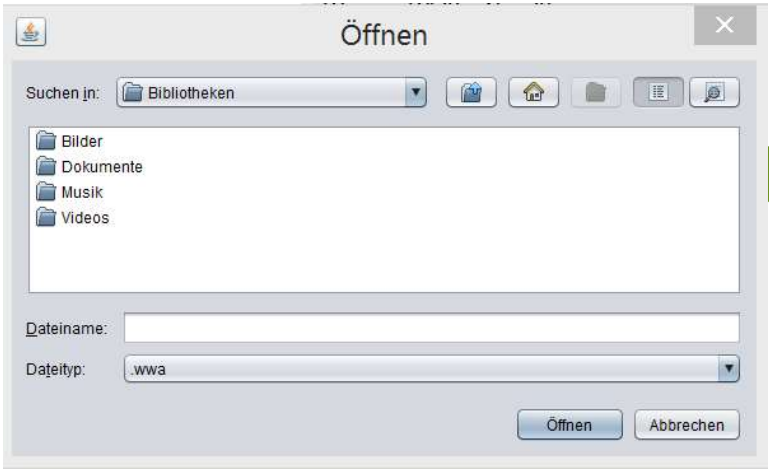
```
mntmSpeichern = new JMenuItem("Speichern");
mntmSpeichern.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent save) {
```

Klasse:
userInterface.java
Attribut:
private String speicherPfad

Klasse:
userInterface.java
Methode:
actionPerformed(ActionEvent save)

Optimierung: Umsetzung – Laden



- .APPROVE_OPTION gibt einen Wert zurück, wenn Ja/Ok Option ausgewählt wurde
- Nach Abbrechen ist der Pfad = null

```
if(chooser.showOpenDialog(null) == JFileChooser.APPROVE_OPTION){
    pfadAuswahl = chooser.getSelectedFile().getAbsolutePath();
    System.out.println("der gewählte Pfad ist:" + pfadAuswahl);
}
else{
    pfadAuswahl=null;
    System.out.println("File-Open-Funktion wurde abgebrochen (Cancel): " + pfadAuswahl);
}
```

Optimierung:
Umsetzung – Verlinkung zum Benutzerhandbuch

```
JLabel link = new JLabel("<html>"    // ONA Defintion und Text für Hyperlink
        + "- <a href=\"\">Hier</a> sind ausführliche Information zur Methode hinterlegt </html>"
        );
link.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR)); //  ONA Cursor Einstellungen für den Hyperlink-Text
link.addMouseListener(new MouseAdapter() {                      //  ONA Maus-Event Definieren für den Hyperlink-Text
    public void mouseClicked(MouseEvent e){                     //  ONA Maus Event-Click Beschreiben
        if(e.getClickCount() > 0){
            try { // ONA     zu öffnete PDF-Datei definieren
                File pdfFile = new File(System.getProperty("user.dir")+"\\"+"Benutzerhandbuch_Wagner_Whitin_1.1.pdf");
                if (pdfFile.exists()) { // ONA Abfrage ob die Datei existiert

                    if (Desktop.isDesktopSupported()) {
                        Desktop.getDesktop().open(pdfFile); // ONA PDF datei öffnen
                    } else {
                        System.out.println("Awt Desktop is not supported!");
                    }

                } else {
                    System.out.println("File is not exists!");
                }

                System.out.println("Done");

              } catch (Exception ex) {
                ex.printStackTrace();
              }
            }
    }
});
```

- awt: abstract window toolkit – Teil der Java Foundation Classes
- Verwendet java.awt.event.MouseAdapter java.awt.event.MouseEvent

Wagner-Whitin Verfahren

Der *Wagner-Whitin*-Algorithmus reduziert die Anzahl der zu überprüfenden Alternativen mittels einer Rekursion. Für jede Periode wird die bis zu dieser Periode optimale Bestellpolitik bestimmt. Nimmt man an, dass bis zur Periode t-1 alle optimalen Bestellpolitiken ermittelt worden sind, so kann sich die Suche nach der optimalen Bestellpolitik bis zur Periode t darauf beschränken, Strategien zu vergleichen, die aus einer Kombination der optimalen Bestellpolitik bis zur Periode i-1, i = 1... t und einer Bestellung in i für die Perioden i bis t bestehen. Die Rekursionsgleichung hat die folgende Form:

$$F_t = \min_{1 \le i \le t} \left(F_{i-1} + k_F + k_L \sum_{j=i}^{t} (j-i) b_j p \right)$$

mit:

F_i = Kosten einer optimalen Bestellpolitik bis zur Periode i, i = 1,..., t
k_F = fixe Bestellkosten
k_L = Lagerkostensatz
b_i = Bedarf der Periode j
p = Stückpreis

Quelle: Toporowski, W., Logistik im Handel, 1996 S175ff

Wagner-Whitin-Verfahren

- Das Modell geht von folgenden Annahmen aus:
  - Die Höhe des Bedarfs ist für alle Perioden des Planungszeitraumes bekannt.
  - Die Ware geht jeweils zu Beginn einer Periode zu und ab.
  - Es kann in jeder Periode bestellt werden.
  - Es gibt keine finanziellen und räumlichen Grenzen. (d.h. keine Kapazitäten werden berücksichtigt)
  - Die Lieferzeit beträgt null Perioden
  - Der Lagerbestand zu Beginn und am Ende des Planungszeitraumes belaufen sich auf Null

Quelle: Toporowski, W., Logistik im Handel, 1996

Beispiel Wagner-Whitin-Verfahren

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

1. Periode (Ein-Perioden-Problem)

- Minimalkosten = Bestellkostensatz (Rüstkosten) und keine Lagerkosten

Beispiel Wagner-Whitin-Verfahren

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

2. Periode (Zwei-Perioden-Problem)

- Bestellkosten 1 + Lagerkosten * Bedarf 2
- Bestellkosten 1 + Bestellkosten 2

Beispiel Wagner-Whitin-Verfahren

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				

3. Periode (Drei-Perioden-Problem)

- Bestellkosten 1 + Bestellkosten 2 + Lagerkosten * Bedarf 3
- Bestellkosten 1 + Bestellkosten 2 + Bestellkosten 3

Beispiel Wagner-Whitin-Verfahren

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				

4. Periode

- **Bestellkosten 1 + Bestellkosten 2 + (Lagerkosten * Bedarf 3) + 2 * (Lagerkosten * Bedarf 4)**
- **Bestellkosten 1 + Bestellkosten 2 + Lagerkosten * Bedarf 3 + Bestellkosten 4**

Beispiel Wagner-Whitin-Verfahren

Optimale Lösung

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

- Kosten von 136 GE

Live Demo

- Bestellkosten: 20 GE
- Lagerkosten: 1 GE/ME/ZE
- Bedarfe:
  - 1. Periode: 30 ME
  - 2. Periode: 30 ME
  - 3. Periode: 20 ME
  - 4. Periode: 30 ME
  - 5. Periode: 50 ME
  - 6. Periode: 10 ME
  - 7. Periode: 20 ME

Live-DEMO

Verbesserungsvorschlag / Fazit

- Funktionalitäten sind vorhanden
- Benutzerfreundlich
- Läuft zuverlässig

- Verbesserung der Codestruktur -> Code-Refactoring