Anwendung der linearen Optimierung   
Portierung / Neuprogrammierung „Wagner Whitin –

Lagerhaltungsoptimierung“

Doreen Brunner, Mark Deppe und Marcel Prügel

Inhalt

[Programm: Wagner-Whitin 3](#_Toc358809569)

[1 Ausgangs–Zustand 3](#_Toc358809570)

[1.1 Wagner-Whitin auf Windows XP 3](#_Toc358809571)

[1.2 Wagner-Whitin auf Windows 7 4](#_Toc358809572)

[2 Zu lösende Aufgabe 4](#_Toc358809573)

[3 Mockups 5](#_Toc358809574)

[4 Umsetzung 6](#_Toc358809575)

[4.1 Wagner-Whitin Algorithmus 6](#_Toc358809576)

[4.2 Vorgehensweise des Wagner-Whitin Verfahrens: 7](#_Toc358809577)

[4.3 Konkrete Umsetzung 9](#_Toc358809578)

[5 Das fertige Programm 12](#_Toc358809579)

[6 Quellen 14](#_Toc358809580)

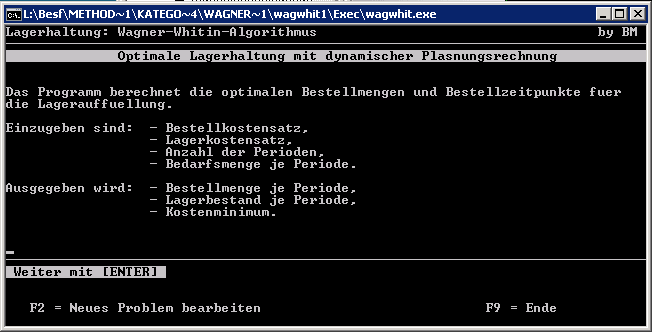
## Programm: Wagner-Whitin

## Ausgangs–Zustand

### Wagner-Whitin auf Windows XP

* Lauffähiges Programm (mit kleinen Fehlern der Programmlogik)
* 1993 entwickelt mit der Programmiersprache Pascal
* Veraltete Oberfläche (ähnlich Dos-Programmen)
* Auf Basis des Wagner-Whitin Algorithmus
* Berechnung optimaler Bestellmengen und Bestellzeitpunkte für die Lagerverwaltung

So sieht das alte Programm aus:



Eingegeben werden konnte:

* der Bestellkostensatz (DM/Bestellung)
* der Lagerkostensatz (DM/ME\*Zeit)
* die Anzahl der Perioden
* die Bedarfsmenge je Periode

Als ersten Schritt gibt man zunächst den Bestell- und Lagerkostensatz, sowie die Anzahl der Perioden ein. Wird dies bestätigt kann man - je nach Anzahl der Perioden – die Bedarfsmenge pro Periode eingeben.

Ausgegeben wurde:

* die Bestellmenge je Periode
* der Lagerbestand je Periode
* das Kostenminimum



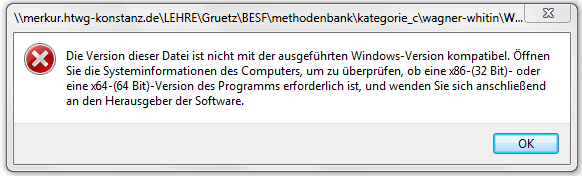
Das Ergebnis wird übersichtlich in einer Tabelle ausgegeben. Man erkennt, dass die Bestellmenge und der Lagerbestand immer größer/gleich der Bedarfsmenge sind. Steht bei „Anzahl Perioden“ eine 0, so wird in dieser Periode gar nicht produziert. Steht dort eine 1, so wird in dieser Periode auch nur für den Bedarf dieser Periode produziert. Ist die Zahl größer 1, so wird für weitere Perioden (je nach Anzahl) vorproduziert

Nach Ausgabe der Tabelle und des Kostenminimums kann man nun ein neues Problem berechnen oder das Programm beenden.

**Bei genauerer Betrachtung und Testen das Programm ist uns aufgefallen, dass das Programm bei einer Bedarfsmenge von 0 in einer Periode falsch rechnet und dann das Ergebnis nicht mehr stimmt. Somit war klar, dass ein einfaches Umschreiben des alten Pascalcodes in Java nicht möglich war und die Logik verändert oder neu programmiert werden musste!**

### Wagner-Whitin auf Windows 7

Auf Windows 7 war das Programm leider nicht lauffähig. Es erschien die nachfolgende Fehlermeldung beim Aufruf des Programms.



## Zu lösende Aufgabe

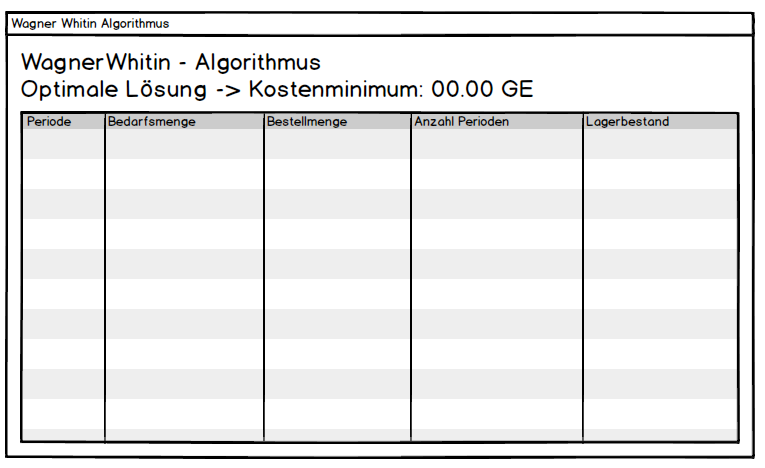
Da der Code in der veralteten Programmiersprache Pascal verfasst wurde, somit schwer portierbar ist und die GUI sowieso sehr veraltet erscheint, haben wir uns entschieden das Programm in Java neu zu entwickeln. Nachdem wir uns eine Weile mit dem alten Sourcecode befasst haben ist uns aufgefallen, dass dieser nicht völlig fehlerfrei programmiert wurde, weshalb wir diesen nicht als Vorlage verwenden konnten, sondern eine neue Logik entwickeln mussten.

Unsere Aufgabe war:

* Ein Logikverständnis für den Wagner-Whitin Algorithmus bekommen
* **Neuprogrammierung des Programms in der Programmiersprache Java**  
  🡪 **Programm soll Windows7 lauffähig gemacht werden**
* Vorbereitung für variablen Lagerkosten schaffen (auf Wunsch von Herrn Grütz)  
  🡪 für ein zukünftiges Projekt
* Entwickeln einer passenden GUI
* Testen der Lauffähigkeit auf Windows7
* Dokumentation der Durchführung und Ergebnisse
* Übergabe des Programms an die Gruppe des Teamprojektes

## Mockups

### 



## Umsetzung

### Wagner-Whitin Algorithmus

Zu aller erst einmal mussten wir ein Verständnis des Wagner-Whitin Algoritmus bekommen, um die Logik unseres Tools programmieren zu können.

Dieser 1958 von den zwei Professoren Harvey M. Wagner und Thomson M. Whitin entwickelte und benannte Algorithmus basiert auf der Basis dynamischer Optimierung. Er besteht nicht allein aus Formeln, sondern ist vielmehr ein komplettes Verfahren zur Bestimmung optimaler Losgrößen für Produkte, mit einer immer wieder ändernden Nachfrage, bei einstufiger Fertigung und ohne das Berücksichtigen vorhandener Kapazitäten. Das heißt es kann von einer unendlich hohen Produktionsgeschwindigkeit ausgegangen werden, wobei der Zeitpunkt benötigter Produkte nur auf die Periode definiert ist und nicht auf einen genaueren Zeitpunkt in der Periode. „Wagner und Whitin haben ihr Verfahren so aufgebaut, dass es in der Lage ist, eine Folge von Losen mit unterschiedlicher Größe und unterschiedlichen Zeitabständen zu ermitteln, sodass die Gesamtkosten minimiert werden können.“[[1]](#footnote-1)  
Zunächst einmal wird bei dem Wagner-Whitin Algorithmus errechnet welche Bestellungs-/Rüstkosen und Lagerkosten pro Periode aufgrund von Bedarf anfallen könnten, bis anschließend im Rückwärtsverfahren entschieden wird, welche Kombination von Bestellungen/Produktion und Lagerungen die Günstigste ist. Hierbei ist wichtig zu beachten, dass eine Produktion/Bestellung nur stattfindet, wenn das Lager leer ist – d.h. auch, dass ein Bedarf immer entweder komplett aus Lagerbestand oder komplett aus einer Neuproduktion/Neubestellung gedeckt wird.

### Vorgehensweise des Wagner-Whitin Verfahrens:

Das erste Ziel des Wagner-Whitin Verfahrens ist eine Matrix mit Feldern zu erstellen, in der die Kosten enthalten sind, die anfallen würden, falls man in einer bestimmten Periode produzieren würde.   
Das heißt die Felder werden grundsätzlich, falls sie befüllt werden, mit den Rüst-/Bestellkosten befüllt. Dazu kommen Lagerkosten, sowie die minimalsten Kosten die zuvor entstanden sind um den Bedarf der Perioden zu decken.

Zur Veranschaulichung der Felderbefüllung erstellen wir beispielhaft eine Matrix.  
Die Rüstkosten betragen in diesem Fall 100GE. Die Lagerkosten 1GE/ME\*Periode.  
Der Bedarf der jeweiligen Periode ist (in ME):

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Periode | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Bedarf pro Periode | 30 | 30 | 20 | 30 | 50 | 10 | 20 |

**Die Beispielsmatrix:**   
Die Spalte steht für den Periodenzeitpunkt der Produktion und die Zeile für die Produktion bis zur x. Periode

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Produzierende Periode | Produktion bis x | | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | 100 | 130 | 170 | 260 | 460 |  |  |
| 2 |  | 200 | 220 | 280 | 430 |  |  |
| 3 |  |  | 230 | 260 | 360 |  |  |
| 4 |  |  |  | 270 | 320 | 340 | 400 |
| 5 |  |  |  |  | 360 | 370 | 410 |
| 6 |  |  |  |  |  | 420 | 430 |
| 7 |  |  |  |  |  |  | 440 |
|  |  |  |  |  |  |  |  |

**Doch wie kam diese Matrix zustande?**

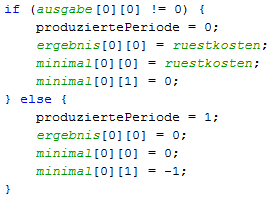
Man betrachtet immer erst die Spalten und dann die Zeilen.  
**1.** Im Feld [1][1] sind die Rüstkosten = 100GE enthalten. Das heißt im Periode 1 wird nur für den Bedarf von Periode 1 produziert.   
**2.** Was wäre nun, wenn man in Periode 1 für Periode 1 und 2 produzierten würde?  
Im Feld [1][2] fallen nun die Rüstkosten 100GE an und die Lagerkosten 1\*30 an. 🡪130GE.  
**3.** Nun betrachten wir das Feld[2][2]. D.h. in der ersten Periode wird für Periode 1 produziert (Rüstkosten1=100GE) und in der zweiten Periode für Periode 2 (Rüstkosten2=100GE). 🡪200GE  
**4.** Im Feld[1][3] werden die Kosten dargestellt, die entstehen bei Produktion in der 1. Periode für die 1. (100GE), 2. (1GE\*30) Und 3. Periode(2\*20\*1GE). 🡪170GE  
**5.** Nun betrachten wir Feld[2][3]. Für die Produktion in Periode 2 entstehen die Rüstkosten 100GE und die Lagerkosten für den Bedarf für Periode 3 (20\*1GE). Zusätzlich müssen noch die Kosten die in Periode 1 angefallen sind betrachtet werden, d.h. 100GE. 🡪220GE  
**6.** Feld [3][3].Bei der Produktion in Periode 3 fallen die Kosten für den Bedarf der Periode 3 (100GE) an und die Kosten, die ich in der ersten Periode benötigte, um für Periode 1 und 2 zu produzieren(130GE). Hier nehme ich den Wert von Periode 1 und keinen aus Periode 2, da es ersichtlich ist, dass in Periode 1 für Periode 2 mit zu produzieren billiger ist als in 1 und 2 extra. 🡪230GE  
**7.** Nun gehen wir wieder in die erste Zeile und betrachten die Produktion in der ersten Periode für Bedarf bis zur vierten Periode. Im Feld[1][4] entstehen die Kosten aus Feld [1][3] und die Lagerkosten für den Bedarf der vierten Periode (3\*1GE\*30). 🡪260GE  
**8.** Feld[2][4]. Hier entstehen die Kosten für das Rüsten in Periode 1 und 2 (200GE) und die Lagerkosten für Periode 3 (20\*1GE) und Periode 4(2\*30\*1GE). 🡪280GE  
**9.** Feld[3][4]. Hier betrachtet man das Minimum von der Produktion bis zum Bedarf der Periode 2 (130GE), die Rüstkosten in Periode3(100GE) und die Lagerkosten für Periode 4(30\*1GE). 🡪260  
**10.** Feld[4][4]. Für die Produktion in Periode 4 fallen Rüstkosten (100GE) an, auf die man noch die kostengünstigste Möglichkeit addieren muss, um den Bedarf von Periode 1 -3 decken zu können (170GE). 🡪270GE  
*Wir müssen so lange überlegen noch in der ersten Periode zu produzieren bis der optimale Preis nicht mehr in der ersten Periode liegt. Da es aber mit 260GE immer noch am günstigsten ist in Periode 1 für bis Periode 4 zu produzieren schauen wir ob es auch noch am günstigsten ist Periode 1 für bis Periode 5 zu produzieren.*  
**11.** Feld [1][5] fallen die Kosten für Feld [1][4] an plus die Lagerkosten für den Bedarf aus Periode 5 (4\*50\*1GE). 🡪460GE  
**12**. In Feld [2][5] fallen die Kosten aus Feld [2][4] (280GE) an plus die Lagerkosten für den Bedarf aus Periode 5 (3\*50\*1GE). 🡪 430GE  
**13.** In Feld [3][5] fallen die Kosten aus Feld [3][4] (260GE) an plus die Lagerkosten für den Bedarf aus Periode 5 (2\*50\*1GE). 🡪360GE  
**14.** In Feld [4][5] fallen die Kosten aus Feld [4][4] (270GE) an plus die Lagerkosten für den Bedarf aus Periode 5 (1\*50\*1GE). 🡪320GE  
**15.** Produziert man in der 5.Periode fallen die minimalen Kosten an, die nötig sind um den Bedarf aus der Periode 1-4 zu decken + die Rüstkosten. (100GE)🡪360GE  
*Nun ergibt sich zum ersten Mal ein Feld in dem weniger Kosten entstehen als wenn man in Periode 1 produzieren würde. Deshalb macht man nun mit dem Feld [4][6] neben den minimalen Kosten (Feld[4][5] 🡪320GE) weiter.*Weitere Erklärung scheint nun unnötig, da die Vorgehensweise bis her nachvollziehbar beschrieben wurde und das weitere Verfahren nach demselben Prinzip erfolgt.  
  
Ist nun die Matrix aufgestellt, wird geschaut, welche Kombinationsmöglichkeiten die günstigste ist.   
Hierzu beginnt man die Matrix von hinten zu betrachten und schaut sich die letzte Spalte an. Nun nimmt man den Wert der am kleinsten ist, der in unserem Fall sich in Feld[4][7] befindet. Das heißt wir suchen die Periode, in der es am günstigsten ist den Bedarf bis zu unserer letzten Periode 7 zu decken. In unserem Beispiel wird also in Periode 4 der Bedarf für die Periode 4-7 produziert. Da in Periode 4 der Bedarf ab Periode 4 produziert wird brauchen wir also nur noch die Spalten der Perioden 3 und niedriger betrachten. Der niedrigste Wert in Spalte 3 ist der Wert 170GE, welcher durch die Produktion in Periode 1 anfällt.

**Zusammenfassend ist zu sagen:  
In unserem Beispiel entstehen Gesamtkosten von 400GE.  
In Periode 1 werden 80ME produziert (Bedarf Periode 1-3).  
In Periode 4 werden 110ME produziert (Bedarf Periode 4-7).**

### Konkrete Umsetzung

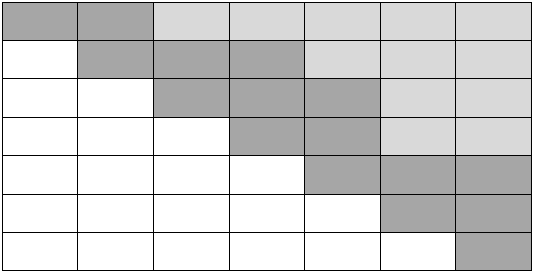
Die in Kapitel 1.3.1 beschriebene Matrix zur Berechnung der Kosten wird im Programm durch ein zweidimensionales Array umgesetzt. Der Umfang des Arrays beträgt jeweils die Anzahl der Perioden in der Länge sowie auch in der Tiefe. Dieses Array wird im Programm nur zur Berechnung verwendet.

Deklaration des Berechnungs-Arrays im Code:

Die Berechnung beginnt in der ersten Zeile der ersten Spalte des Arrays (also in Periode 0). Dort wird mittels einer IF-Bedingung abgefragt, ob in dieser Periode überhaupt Nachfrage herrscht. Falls ja bleibt keine andere Wahl als in dieser Periode zu produzieren. Falls nein, macht es auf keinen Fall Sinn in dieser Periode zu produzieren, da die Produktionskosten in jeder Periode identisch sind und auf jeden Fall Lagerkosten anfallen würden. Der Beginn ist also recht einfach, da es nur diese beiden Alternativen gibt. Realisiert wird dies im Code durch so:

Das Array ausgabe[][] beinhaltet in der ersten Zeile die Nachfragemengen der jeweiligen Periode. Also wird in der IF Bedingung die Nachfragemenge in Periode 0 abgefragt. Ist diese ungleich 0, wird produziert. Das bedeutet, dass Rüstkosten anfallen. Diese werden in die Berechnungsmatrix auch direkt eingetragen, da es die einzigen Kosten dieser Periode sind. Es gibt auch keine Alternative, daher sind es gleichzeitig auch die Minimalen Kosten. Dies wird in die erste Zeile des Arrays minimal[][] eingetragen. In der zweiten Zeile befindet sich der Index der Periode, in der zu diesem Zeitpunkt zuletzt produziert wurde. Da es vorher keine Produktion gab, wird hier der Index 0 eingetragen.

Ab Periode 1 übernimmt das weitere Vorgehen dann eine FOR-Schleife. Sie durchläuft das gesamte Array von Periode 1 an bis Ende:

Verschachtelt in dieser ersten FOR-Schleife befindet sich eine zweite FOR-Schleife, die das Array pro Periode in die Tiefe abläuft und die Berechnungen durchführt. Hier ist zu beachten, dass das Array in der Tiefe erst ab der zuletzt produzierten Periode starten muss, da es vorher auf jeden Fall teurer wäre. So kann man sich einige Berechnungsschritte sparen. Außerdem muss das Array auch nur bis zur aktuellen Periode in der Tiefe berechnet werden.

Die Abbildung links soll das noch einmal verdeutlichen. Der graue Teil ist der, der für die Berechnung relevant ist. Jedoch kann nur im dunkelgrauen Bereich wirklich ein Minimum auftauchen, daher kann der hellgraue Teil vernachlässigt werden.

Im Code wird dies so realisiert:

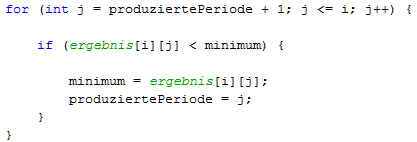
Falls der Index der ersten FOR-Schleife nun gleich des Index der zweiten FOR-Schleife ist (i = j), ist der Fall eingetreten, in dem produziert wird (siehe Kapitel 1.3.1). Hier müssen also nur die Produktionskosten zu den aktuellen Minimalkosten der letzten Periode addiert werden.

Alle anderen Felder des Arrays müssen aber anders berechnet werden. Dies betrifft den Fall, dass nicht produziert wird, sondern gelagert. Was wiederrum bedeutet, dass die gesamte Nachfragemenge der aktuellen Periode am letzten Produktionszeitpunkt produziert wurde und seitdem eingelagert war. Da die fixen Produktionskosten durch die Mehrproduktion nicht steigen und die variablen Produktionskosten vernachlässigt werden, da sie in jeder Periode anfallen würden, müssen hier aber nur die Lagerkosten berücksichtigt werden. Allerdings müssen diese für alle Perioden seit der letzten Produktion berechnet werden. Sind diese Lagerkosten außerdem noch variabel, wird diese Berechnung noch einmal erschwert. Im Code wurde die Berechnung der Lagerkosten so umgesetzt:

Die Lagerkosten (werden in einer externen Methode lagerkostenBerechnen() berechnet) werden zu den Kosten der letzten Periode addiert. Diese Kosten befinden sich in der Matrix direkt links vom aktuell bearbeiteten Feld (daher [i - 1]) .

Zur Berechnung der Lagerkosten benötigt man die aktuelle Nachfragemenge, sowie die Koordinaten des aktuellen Feldes. Innerhalb der Methode wird nun das Berechnungsarray ab der Periode, in der zuletzt produziert wurde bis zur aktuellen Periode durchlaufen und in jeder Periode die Lagerkosten mit der zusätzlichen Bedarfsmenge berechnet. Die Lagerkosten werden (da sie ja variabel sein können) aus einem zusätzlichen Array gezogen, das die jeweiligen Lagerkosten zur aktuellen Periode bereithält.

Nachdem die vertikale Berechnung der jeweiligen Periode abgeschlossen ist, muss das Array in der Tiefe noch einmal durchlaufen werden, da das Minimum der aktuellen Periode berechnet werden muss. Dies ist notwendig, da in der nächsten Periode die Produktionskosten im Fall (i = j) auf die Minimalkosten der letzten Periode addiert werden müssen (siehe oben). Außerdem wird für das Minumum auch die Periode gespeichert, in der zu diesem Zeitpunkt produziert wurde. Dies ist notwendig um so später nachzuvollziehen, welchen „Weg“ der Algorithmus durch die Matrix gewählt hat um auf das Kostenminimum zu kommen.

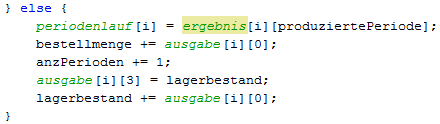


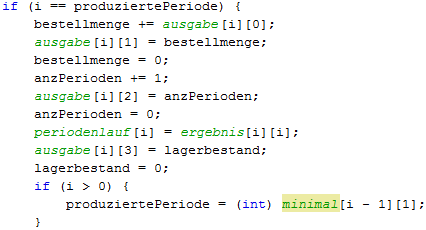
Nun ist das Berchnungsarray fertiggestellt und theroretisch die Lösung auch bereits fertig. Da der Benutzer aber damit nicht besonders viel anfangen kann, müssen die Daten noch aufbereitet werden. Hierzu ist eine weiter Methode notwendig. Diese Methode beinhaltet wieder eine FOR-Schleife, die das Berechnungsarray diesmal von rechts nach links durchläuft. In der letzten Periode wird zuallererst das Minimum gesucht. Zu diesem Minimum wird dann die zugehörige Periode ermittelt, in der produziert wurde. Diese beiden Werte wurden ja während der Berechnung (siehe oben) immer abgespeichert und befinden sich nun im Array minimal[][].

Die Periode, in der zuletzt produziert wurde, wird nun also ausgelesen: 

Anschließend wird das Array von rechts nach links durchlaufen bis zu dieser Periode: 

In jeder Periode, die dazwischen liegt, wird jeweils die Bestellmenge zu diesem Zeitpunkt auf eine Variable addiert (diese Variable wird später in der weiteren Berechnung gebraucht). Außerdem wird für jede Periode ein Zähler hochgezählt, der berechnen soll, für wieviele Perioden produziert wurde. Anschließend wird noch der aktuelle Lagerbestand zu diesem Zeitpunkt berechnet.

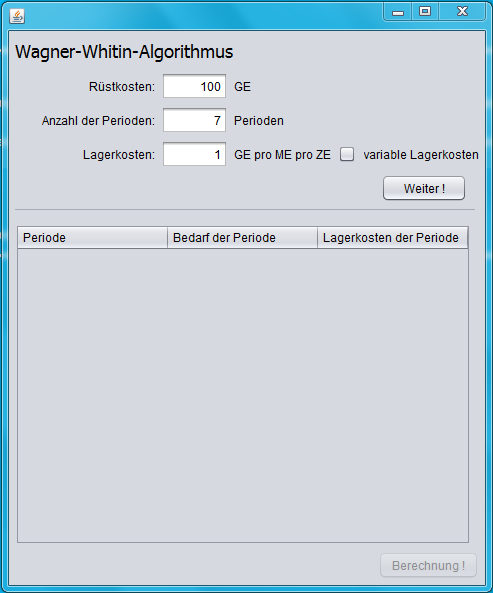


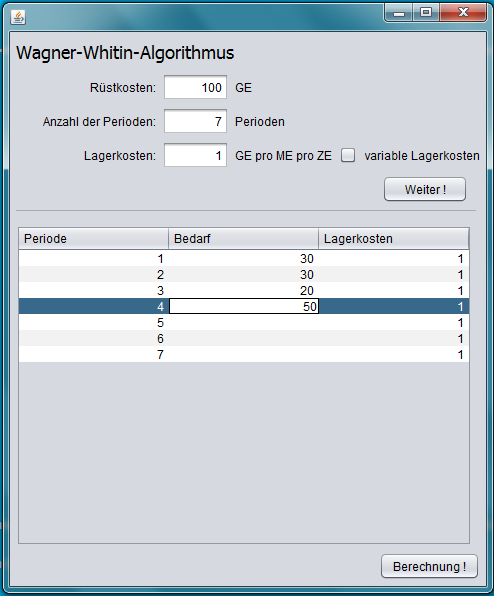
Die wichtige Berechnung geschieht aber zu dem Zeitpunkt, wenn die produzierte Periode von der FOR-Schleife erreicht wurde (also im IF-Teil der Bedingung). Dort wird die aktuelle Bestellmenge noch einmal addiert und dann in die Ausgabematrix gespeichert. Dann wird die Bestellmenge für die folgende Berechnung wieder auf 0 gesetzt. Nun weiß man, welche Menge hier produziert werden muss um alle nachfolgenden Perioden abdecken zu können. Genau das gleiche wird auch mit dem Zähler gemacht, der die Perioden zählen soll, für die Produziert wird. Auch dieser wird noch einmal um 1 erhöht, bevor er dann in die Ausgabematrix gespeichert und resettet wird. Auch der Lagerbestand wird nun gespeichert und auf 0 gesetzt. Allerdings wird dieser nicht noch einmal erhöht, da die Bestellmenge zu dieser Periode ja sofort benötigt wird und nicht eingelagert werden muss.

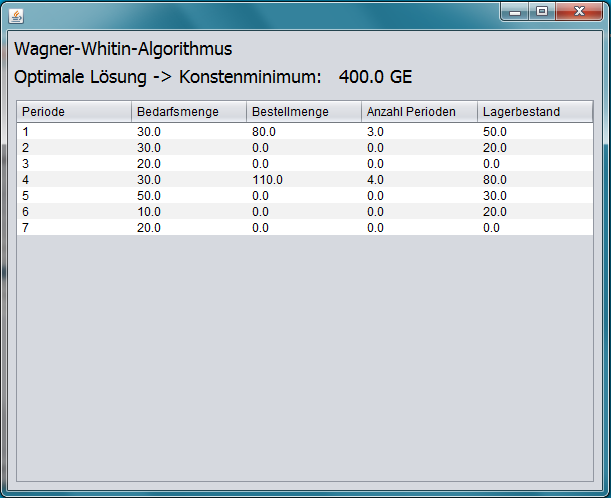
Nun ist die Berechnung der relevanten Werte für die Ausgabe fertiggestellt und befindet sich im Array ausgabe[][]. Dies betrifft die Nachfragemenge, die Bestellmenge, die Anzahl der Perioden für die produziert wurde sowie den Lagerbestand jeweils immer zum Zeitpunkt t.

## Das fertige Programm

Dies ist die Startansicht unseres neu entwickelten Wagner-Whitin Programms. Eingegeben werden können die Rüstkosten, die Anzahl der Perioden und die Lagerkosten, sowie, nach Klicken auf den „Weiter“-Button, den Bedarf pro Periode.





Klickt man nun auf „Berechnen“ so wird das Kostenminimum errechnet. Zusätzlich wird eine Tabelle mit den Perioden, deren Bedarfsmengen, deren Bestellmengen, für welche Anzahl von Perioden in einer Periode bestellt wird und der Lagerbestand der Periode angezeigt. 

## Ausblick

Ein nächster Schritt wäre einen Dateiexport bzw. Dateiimport möglich zu machen, damit man gerade bei vielen Perioden nicht die Arbeit hat viele Bedarfsmengen neu einzutippen.

Außerdem wäre für zukünftige Projekte wichtig, dass bei der Neuprogrammierung des Wagner-Whitin per linearer Methode die selbe GUI verwendet werden sollte.

## Quellen

- „Wagner Whitin Verfahren und dessen Anwendung“, Tim Florian Schucht, 2010 Göttingen, GRIN Vertrag

-Vorlesungsskript „Bestellmengenplanung“ von Prof. Dr. W. Toprowski (Professur für Handelsbetriebslehre) [*www.uni-goettingen.de/en/85867.html*](http://www.uni-goettingen.de/en/85867.html) *(Stand April 2013)*

1. „Wagner Whitin Verfahren und dessen Anwendung“, Tim Florian Schucht, 2010 Göttingen, GRIN Vertrag, Seite 5 [↑](#footnote-ref-1)