Anwendungen der linearen Optimierung

PROJEKTDOKUMENTATION

Iterator 2.0

Autoren:

Tobias Stübler, 290911
Lucas Herre, 290723

Vorzulegen bei: *Prof. Dr. Grütz*
Vorzulegen am: *29.06.2017*

Inhalt

I. Abbildungsverzeichnis

II. Tabellenverzeichnis

1 Hinweis

Die nachfolgend beschriebene Projektdokumentation bildet eine Erweiterung zu der bereits bestehenden Projektdokumentation zum Iterator2013. Diese Projektdokumentation beinhaltet alle Optimierungen des Iterators, die im ALO – Projekt „Iterator 2.0“ im Sommersemester 2017 umgesetzt wurden.

Des Weiteren bleibt das aktuelle Benutzerhandbuch gültig, da an den grundlegenden Funktionen des Iterators nichts verändert wurde.

2 Einleitung

Im Folgenden wird die Optimierung des „*Iterator 2.0“* dokumentiert, welche im Rahmen der Lehrveranstaltung Anwendungen der linearen Optimierung an der HTWG Konstanz erfolgte.

2.1 Projektziele

Im Lastenheft wurden zu Beginn des Projektes folgende Ziele festgelegt:

- Den Button „Optimieren“ wieder ersichtlich machen und die Funktionalität gewährleisten.
- Iterieren auf Doppelklick ermöglichen.
- Funktion einrichten, dass mit der Tab – Taste das nächste Feld erreicht wird.
- Eingabe der Zahlen optimieren.
- CSS Datei überarbeiten und anpassen.

Zusätzlich zu diesen Anforderungen wurden weitere Verbesserungen am Iterator 2.0 durchgeführt:

- Pop-up Meldung sobald das Optimum erreicht wird
- Pop-up Meldung wenn der Lösungsraum unbeschränkt ist
- Pop-up Meldung bevor Fenster geschlossen wird
- Die Footer-Zeile des Tableaus anpassen
- Die Funktion „Über Iterator“ im Menüpunkt „Hilfe“ lauffähig machen
- Die Dokumentation im Menüpunkt „Hilfe“ anpassen
- Corporate Design einbinden
- Sprachänderung Deutsch-Englisch optimieren
- Text anzeigen wenn mit der Maus über die Symbole New, Save und Open gefahren wird
- Speicherpfad anpassen

2.2 Rahmenbedingungen

- Das Tool soll in Java implementiert werden
- Die Programmierung soll Objektorientiert erfolgen
- Die GUI wird mit JavaFx implementiert
- Das Tool muss unter Windows 7/8.1/10 lauffähig sein

2.3 Projektorganisation

2.3.1 Projektverantwortlicher / Auftraggeber:

Als Auftraggeber und Abnahmeverantwortlicher wird Herr Professor Dr. Grütz der HTWG Konstanz benannt.

2.3.2 Projektmitglieder

Tabelle 1: Projektmitglieder

Name	Matr.Nr.	Studiengang	Rolle
Stübler, Tobias	290911	Wirtschaftsinformatik B.Sc.	Entwicklung
Herre, Lucas	290723	Wirtschaftsinformatik B.Sc	Entwicklung

3 Durchführung des Projektes

In diesem Abschnitt sollen die einzelnen Arbeitspakete zur Optimierung des Iterators 2.0 aufgezeigt werden.

3.1 Button „Optimieren“ ersichtlich machen

In der ursprünglichen Version des Iterators wurde der Button „Optimieren“ innerhalb der grafischen Benutzeroberfläche nicht angezeigt.

Zur Lösung des Problems musste der Parameter „prefWidth“ in der XML – Datei auf den Wert 750.0 abgeändert werden.

```
<TitledPane fx:id="x1" animated="false" prefHeight="62.0" prefWidth="750.0">
```

Abbildung 1: Button „Optimieren“

3.2 Einbinden der CSS Dateien

In der Iterator Version 1.0 waren zwei CSS Dateien vorhanden. Diese wurden aber nicht erkannt. Der Compiler konnte die Dateien nicht lesen. Durch das Entfernen der ersten Codezeilen „@Charset ISO-8859-1“ wurden die Dateien wieder erkannt.

3.3 Tableaufenster optimiert

Der Footer Bereich in der Version 1.0 war nicht ersichtlich. Die Schriftzüge „Reihen, Spalten, neues Fenster und die Bezeichnung der jeweiligen Position“ wurden in der Schriftfarbe weis dargestellt. Durch die Änderung des Parameters „textFill = „black“;“ im FXML Dokument wurden die Beschriftungen wieder ersichtlich.

3.4 Änderungen im Menüpunkt „Hilfe“

Der Menüpunkt „Hilfe“ besteht aus den drei Unterpunkten Über Iterator, Dokumentation und Debug. Im Unterpunkt Debug mussten keine Änderungen vorgenommen werden. Für den Unterpunkt Dokumentation wurde eine neue HTML-Datei erstellt. Die HTML-Datei wurde hinsichtlich ihres Aussehens komplett überarbeitet. Des Weiteren wurde die HTML Seite responsive gemacht und das neue HTWG Logo wurde hinzugefügt. Die Seite ist im nachfolgenden Bild zusehen.

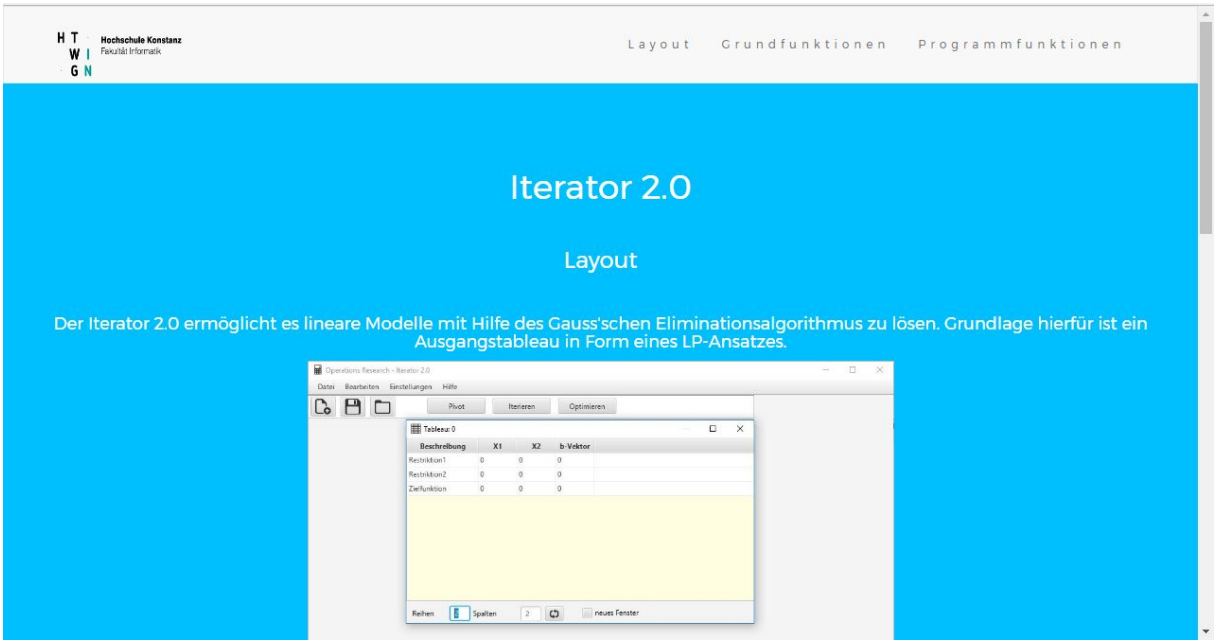

Abbildung 2: Screenshot HTML-Dokumentation

Der Unterpunkt „Über Iterator“ hatte in der Version 1.0 keine Funktion. Hier wurde ein Pop-Up Fenster implementiert, das eine allgemeine Information über den Iterator anzeigt.

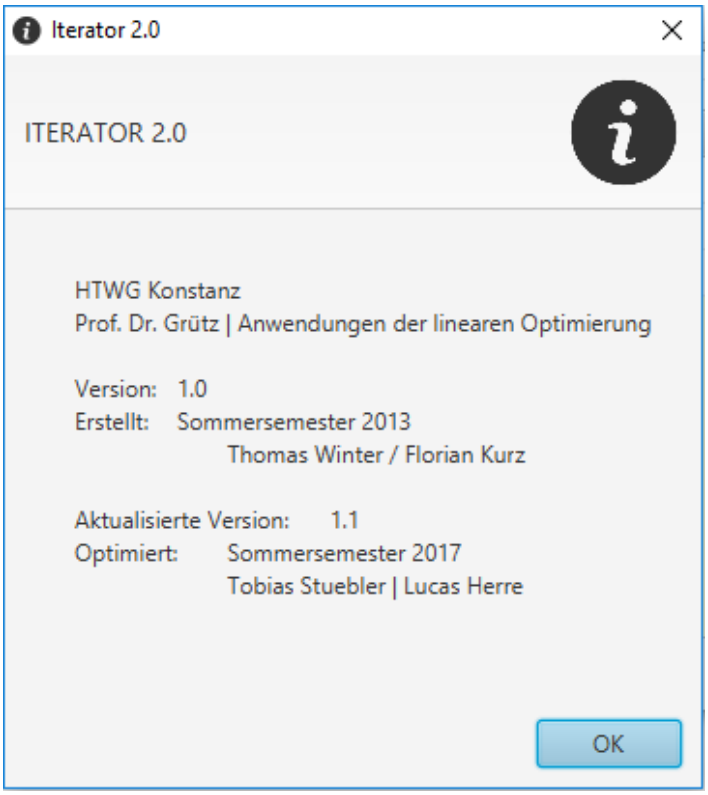

Abbildung 3: Screenshot „über Iterator“

```
// EVENT -> MenuItem ABOUT
itemAbout.setOnAction(new EventHandler<ActionEvent>() {
    // Pop-up with short informations about the Application ITERATOR
    @Override
    public void handle(ActionEvent event) {


        String title = "Iterator 2.0";
        String head = "ITERATOR 2.0";
        String message = "\r\n"+
                "    HTWG Konstanz       \r\n" +
                "    Prof. Dr. Grütz | Anwendungen der linearen Optimierung\r\n\r\n" +
                "    Version:    1.0 \r\n" +
                "    Erstellt:   Sommersemester 2013 \r\n" +
                "                Thomas Winter / Florian Kurz\r\n\r\n" +
                "    Aktualisierte Version:  1.1 \r\n" +
                "    Optimiert:  Sommersemester 2017 \r\n" +
                "                Tobias Stuebler | Lucas Herre\r\n\r\n\r\n";
        ImageView image = new ImageView(this.getClass().getResource("img_about.png").toString());

         Alert alert = new Alert(AlertType.INFORMATION);
            alert.setTitle(title);
            alert.setHeaderText(head);
            alert.setGraphic(image);
            image.setFitHeight(60);
            image.setFitWidth(60);
            alert.setContentText(message);
            Stage stage = (Stage) alert.getDialogPane().getScene().getWindow();
            stage.getIcons().add(new Image(this.getClass().getResource("img_about.png").toString()));

            alert.showAndWait();
```

Abbildung 4: Java-Code „über Iterator“

3.5 Corporate Design

Das neue Corporate Design der HTWG Fakultät Informatik wurde in die Benutzeroberfläche des Iterators 2.0 hinzugefügt. Aus technischer Sicht wurde das Logo in die FXML-Datei implementiert. Der Programmcode dazu wird im nach folgendem Screenshot dargestellt.

```
<bottom>
      <AnchorPane prefHeight="121.0" prefWidth="749.0" BorderPane.alignment="CENTER">
         <children>
            <ImageView fitHeight="300.0" fitWidth="400.0" layoutY="100.0" pickOnBounds="true"
            preserveRatio="true" AnchorPane.bottomAnchor="9.303192138671875" AnchorPane.leftAnchor="0.0"
            AnchorPane.rightAnchor="460.0" AnchorPane.topAnchor="100.0">
               <image>
                  <Image url="@img_htwg_new.png" />
               </image>
            </ImageView>
         </children>
      </AnchorPane>
   </bottom>
```

Abbildung 5: XML-Code „Corporate Design“

3.6 Sprache deutsch-englisch anpassen / vollfunktionsfähig machen

Im Menüpunkt Einstellungen steht die Sprache Englisch und Deutsch zur Verfügung. In der Vorgängerversion 1.0 war die Sprache standardmäßig auf Deutsch eingestellt, allerdings war optisch durch einen Hacken die Sprache Englisch markiert. Des Weiteren änderte sich nur die Sprache im Mainframe und nicht im Tableau. Durch eine Modifizierung der Methode in JAVAFX, öffnet sich nun in der aktuellen Version nach Auswahl der Sprache ein neues Tableau mit der gewünschten Sprache. Beim Start des Solvers wird die Sprache Deutsch ausgewählt und der Hacken unter den Einstellungen markiert auch die Sprache Deutsch.

3.7 Text anzeigen wenn mit Maus über Buttons gefahren wird

Innerhalb der GUI gibt es die Möglichkeit durch betätigen der Buttons „New“, „Save“ und „Open“ die entsprechende Funktion auszuführen. Da die Buttons allerdings nur durch ein Symbol abgebildet werden, kann dies schnell zu Verwechslungen führen.

Um diesem Fall vorzubeugen wurde eine Hover-Funktion implementiert, die die Beschriftung des Buttons anzeigt, sobald mit der Maus über diesen gefahren wird. Aus folgender Abbildung ist der Java-Code ersichtlich, welcher zur Implementierung dieser Funktion verwendet wurde:

```
btnNew.setTooltip(
            new Tooltip(MessageHandler.getInstance().getMessage("gui.item.new"))
            );

btnOpen.setTooltip(
        new Tooltip(MessageHandler.getInstance().getMessage("gui.item.open"))
        );


btnSave.setTooltip(

            new Tooltip(MessageHandler.getInstance().getMessage("gui.item.save"))


);
```

Abbildung 6: Java-Code Hover über Buttons

3.8 Pop-up Meldung wenn das Optimum erreicht wird

Bei Version 1.0 des Solvers gab es keinen Hinweis darauf, egal ob über den Button „Optimieren“ oder durch manuelles durchlaufen des Simplexalgorithmus mittels der Buttons „Pivot“ und „Iterieren“, wann das Optimum erreicht wurde.

Daher wurde eine Meldung implementiert, die immer aufgerufen wird, sobald nach dem Simplexalgorithmus kein weiteres Pivotelement gefunden wird.

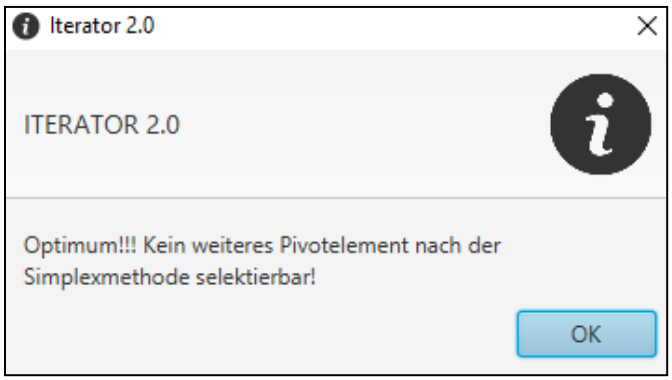

Abbildung 7: Screenshot „Optimum erreicht“

Der Java-Code hierzu sieht folgendermaßen aus:

```
ImageView image = new ImageView(this.getClass().getResource("img_about.png").toString());
 Alert alert = new Alert(AlertType.INFORMATION);
    alert.setTitle("Iterator 2.0");
    alert.setHeaderText("ITERATOR 2.0");
    alert.setGraphic(image);
    image.setFitHeight(60);
    image.setFitWidth(60);
    alert.setContentText(MessageHandler.getInstance().getMessage("gui.action.iterate.optimized"));
    Stage stage = (Stage) alert.getDialogPane().getScene().getWindow();
    stage.getIcons().add(new Image(this.getClass().getResource("img_about.png").toString()));
    alert.showAndWait();
```

Abbildung 8: Java-Code Alert-Befehl

3.9 Pop-up Meldung wenn der Lösungsraum unbeschränkt ist

Sobald alle Elemente der Pivotspalte gegen die Nichtnegativitätsbedingung verstoßen, liegt eine unbeschränkte Lösung vor. Hierfür wurde ein Alert implementiert, welcher auf diese Situation hinweist.

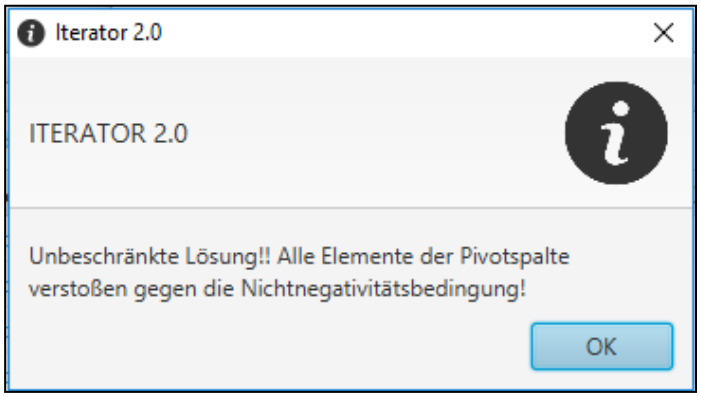

Abbildung 9: Screenshot „unbeschränkte Lösung“

3.10 Pop-up Meldung „Fenster schließen“

In der aktualisierten Version 1.1 öffnet sich ein Fenster, wenn der User das Programm beenden will. So besteht die Möglichkeit das Programm doch nicht zu beenden und die Speicherung durchzuführen. Der nachfolgende Screenshot zeigt die Meldung grafisch.

```
public static boolean display(String title, String message){

    Stage window = new Stage();
    window.setTitle(title);
    window.setMinWidth(300);
    window.setMinHeight(100);

    Label label = new Label();
    label.setText(message);

    Button yesButton = new Button("     Ja      ");
    Button noButton = new Button("     Nein    ");

    yesButton.setMaxWidth(500.0);
    noButton.setMaxWidth(80.0);

    yesButton.setOnAction(e -> {
        answer = true;
        window.close(); });

    noButton.setOnAction(e -> {
        answer = false;
        window.close(); });

    HBox hb = new HBox(yesButton, noButton);
    hb.setMaxWidth(200);
    hb.setSpacing(10);
    hb.setPadding(new Insets(20));

    VBox vb = new VBox(label);
    vb.setSpacing(10);
    //vb.setPadding(new Insets(20));

    FlowPane root = new FlowPane();
    root.setVgap(10);
    root.setHgap(10);
    root.setPrefWrapLength(300);
    root.getChildren().add(label);
    root.getChildren().add(vb);
    root.getChildren().add(hb);
    root.setAlignment(Pos.CENTER);

    Scene scene = new Scene(root,300,100);
    window.setScene(scene);
    window.showAndWait();
    return answer;
```

Abbildung 11: Java-Code „Fenster schließen“

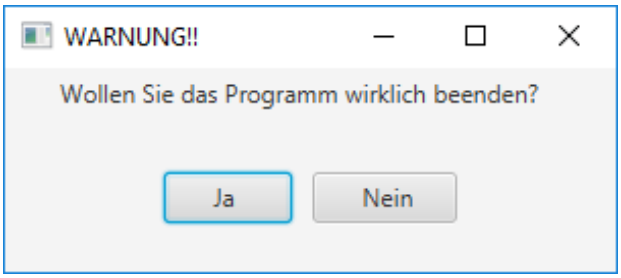

Abbildung 10: Screenshot „Fenster schließen“

3.11 Tableau mit Tab-Taste durchlaufen

Bei der Eingabe der Zahlen in das Tableau war es bisher nicht möglich durch betätigen der Tab-Taste in das nächstfolgende Feld zu springen. Die Koordination der Eingabe musste somit immer mithilfe der Maus erfolgen.

Da die Möglichkeit mit der Tab-Taste durch das Tableau navigieren zu können deutlich Benutzerfreundlicher ist wurde diese Funktion ebenfalls implementiert. Notwendig waren hierfür zwei zusätzliche Methoden, die aufgerufen werden sobald der Anwender die Tabulator-Taste betätigt. Der verwendete Code wird in Abbildung 12 dargestellt.

```
} else if(t.getCode() == KeyCode.TAB){

    anzColumns = tc.getAnzColumns();
    anzRows = tc.getAnzRows();
    System.out.println(anzColumns + " Anzahl");
    TableColumn nextColumn = getNextColumn(!t.isShiftDown());

    commit(textField.getText());
    if (nextColumn != null) {
        //tc.selectCell(1, 1);
        getTableView().edit(currentRow, nextColumn);
    }

}

 private List<TableColumn<S, ?>> getLeaves(
         TableColumn<S, ?> root) {
     List<TableColumn<S, ?>> columns = new ArrayList<>();
     if (root.getColumns().isEmpty()) {
         // We only want the leaves that are editable.
         if (root.isEditable()) {
             columns.add(root);
         }
         return columns;
     } else {
         for (TableColumn<S, ?> column : root.getColumns()) {
             columns.addAll(getLeaves(column));
         }
         return columns;
     }
 }
```

```
private TableColumn<S, ?> getNextColumn(boolean forward) {
    List<TableColumn<S, ?>> columns = new ArrayList<>();
    currentRow = getTableRow().getIndex();
    for (TableColumn<S, ?> column : getTableView().getColumns()) {
        columns.addAll(getLeaves(column));
    }
    // There is no other column that supports editing.
    if (columns.size() < 2) {
        return null;
    }
    int currentIndex = columns.indexOf(getTableColumn());
    int nextIndex = currentIndex;
    if (forward) {
        nextIndex++;
        if (nextIndex > columns.size() - 1) {
            nextIndex = 0;
        }
    } else {
        nextIndex--;
        if (nextIndex < 0) {
            nextIndex = columns.size() - 1;
        }
    }

    if (currentIndex == anzColumns+1) {
        if(currentRow < anzRows){
            currentRow = currentRow+1;
            nextIndex = nextIndex+1;
        }
        //System.out.println(currentRow);

    }
    return columns.get(nextIndex);
```

Abbildung 12: Java-Code Tableau mit Tab durchlaufen

4 GUI

Die Graphische Benutzeroberfläche lehnt sich, wie bereits erwähnt, stark an den vorhandenen Iterator (1996). In Abbildung 13 wird die aktuelle Version des Iterators 2.0 dargestellt.

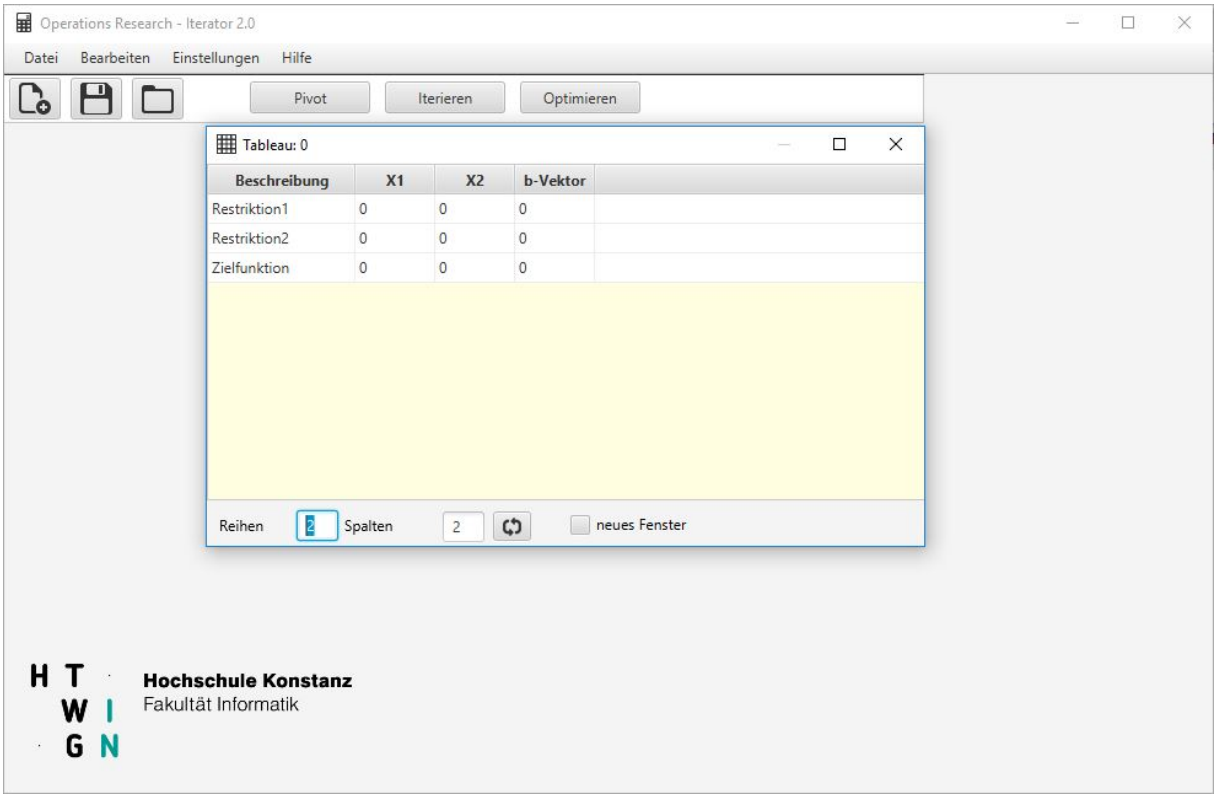

Abbildung 13: Screenshot GUI Iterator 2.0

4.1 Projektabschluss

Mit der Abgabe des Programms und des hier Dokumentierten Programmaufbaus zum 29.06.2017 wurden folgende Ziele aus dem Lastenheft erfolgreich abgehandelt:

- ✓ Den Button „Optimieren“ wieder ersichtlich machen und die Funktionalität gewährleisten.
- ✓ Funktion einrichten, dass mit der Tab – Taste das nächste Feld erreicht wird.
- ✓ Eingabe der Zahlen optimieren.
- ✓ CSS Datei überarbeiten und anpassen.

Weitere Funktionalität

Funktionalitäten, welche über die, im Lastenheft gesetzten Ziele, hinausgehen:

- ✓ Pop-up Meldung sobald das Optimum erreicht wird
- ✓ Pop-up Meldung wenn der Lösungsraum unbeschränkt ist
- ✓ Pop-up Meldung bevor Fenster geschlossen wird
- ✓ Die Footer-Zeile des Tableaus anpassen
- ✓ Die Funktion „Über Iterator“ im Menüpunkt „Hilfe“ lauffähig machen
- ✓ Die Dokumentation im Menüpunkt „Hilfe“ anpassen
- ✓ Corporate Design einbinden
- ✓ Sprachänderung Deutsch-Englisch optimieren
- ✓ Text anzeigen wenn mit der Maus über die Symbole New,Save und Open gefahren wird
- ✓ Speicherpfad anpassen

Nicht erreichtes Ziel

- X Funktion mit Doppelklick iterieren

5 Fazit

Die Einarbeitungszeit zu Beginn der Projektphase war aufgrund der komplexen Programmarchitektur sowie des uns noch unbekannten JavaFX Frameworks relativ hoch. Nachdem die Strukturen des Frameworks ersichtlich waren konnten auch recht schnell erste Resultate erzielt werden. Dabei fiel uns auf, dass für vermeintlich banale Funktionen ein sehr hoher Programmieraufwand notwendig war.

Trotzdem kann über einfache XML-Strukturen ein ansehnliches User-Interface erstellt werden. Abschließend ist zu sagen, dass der Iterator 2.0 keine wesentliche Verbesserung zum Iterator 1.0 darstellt. Der Iterator 2.0 hebt sich lediglich durch sein moderneres User-Interface von seinem Vorgänger ab.