DOKUMENTATION DAKIN 2003

Migration des Dakin auf Windows7

Dieses Dokument protokolliert die Arbeit zur Migration des Dakin 2003 auf Windows 7

Stephan Keßler – 284899

Björn Burandt – 284928

HTWG Konstanz – WIN6 – Anwendung der linearen Optimierung

Inhalt

1. Grundlage der Migration

Dieses Dokument dokumentiert die Arbeit von Björn Burandt (284928) und Stephan Keßler(284899) zur Migration des Dakin 2003 Tools auf das Betriebssystem Windows 7. Dies wurde im Rahmen der Veranstaltung, Anwendung der linearen Optimierung, bei Herrn Prof. Dr. Grütz durchgeführt.

Als Ausgangslagen lagen uns ein Tool und eine Problemstellung zur Hand. Das Ziel war es, das Tool „Dakin 2003“ auf Windows 7 und 8 lauffähig zu machen. Hierzu hatten wir leider nur das Tool zur Hand, da das Vorgängerteam ihre Arbeit leider nicht protokolliert hatte.

1.1 Zielsetzung

Das Ziel der Migration wurde in einem Commitment festgehalten, das im Anhang beiliegt. Kurz zusammengefasst waren die primären Ziele:

1. Überprüfen des Dakin 2003 Tools auf seine Lauffähigkeit auf Windows 7 und das Beheben der Fehler.
2. Verbesserungen in der Benutzeroberfläche.
3. Verbesserung, der Angabe des Solverpfads.
4. Verbesserung der Usability.
5. Aufbau einer Dokumentation

Diese Ziele sollten im Rahmen des Selbststudiums zu der Vorlesung erreicht und in den Übungen präsentiert werden. Eine klare Abgrenzung muss allerdings bei dem Aufbau einer Dokumentation gezogen werden. Diese Dokumentation soll nur die von uns veränderten und angepassten Teile umfassen und nicht die versäumte Gesamtdokumentation ersetzen.

1.2 Ausgangslage des Tools

Zuerst einmal ist zu sagen, dass dieses Tool auf Windows 7 im alten Zustand nicht lauffähig war. Dies sollte im Rahmen der Veranstaltung behoben werden.

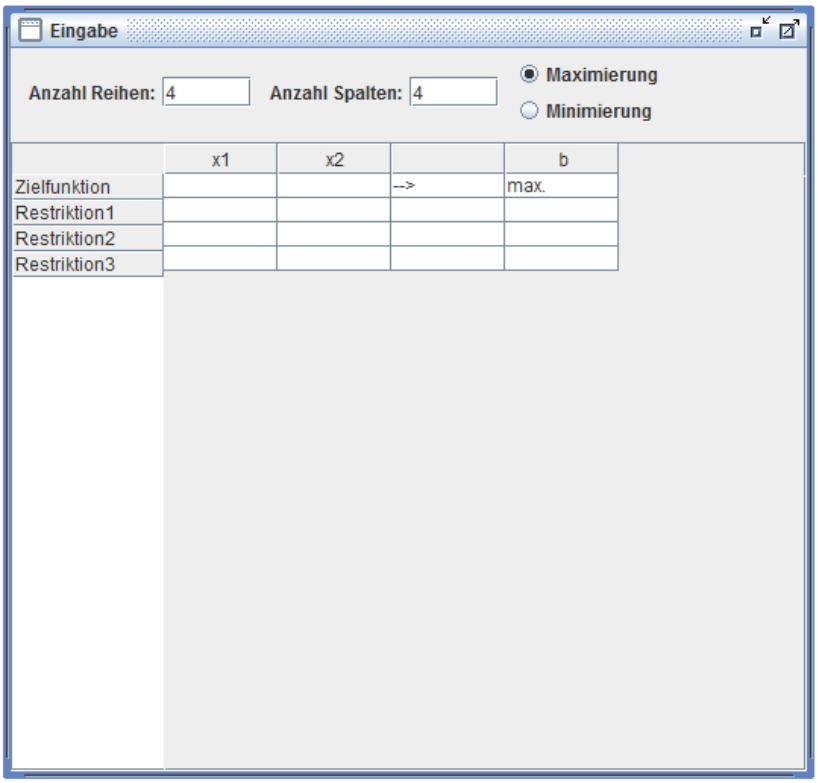
Das Tool wurde im Jahr 2003 von vier Studenten in der Sprache „Java“ programmiert. Es soll anhand von Restriktionen und einer Zielfunktion das Dakinsche Verfahren für ganzzahlige Lösungen von Gleichungssystemen abbilden. Dazu nutzt es den schon gegebenen Solver „LP Solve“. Das Dakinsche Verfahren ist so aufgebaut, dass das System anhand von variabel vielen Restriktionen, eine Zielfunktion maximieren oder minimieren kann. Dies kann dann entweder als Lösungsbaum dargestellt werden, oder auch als grafische Lösung, in dem die Lösungsräume und die Restriktionen gezeichnet werden. (Die Darstellung in einem Diagramm funktioniert aber nur sinnvoll mit maximal 2 Restriktionen) Es soll allerdings auch möglich sein, in der grafischen Lösung, den Lösungspunkt auf einen Blick abzulesen.

In welcher Java Version das Tool damals programmiert wurde, blieb abzuschätzen, da leider nichts dokumentiert wurde. Die Funktionalitäten waren zuletzt auf dem Betriebssystem Windows XP und Windows Server 2003 gegeben. Dort funktioniert das alte Tool auch immer noch. Nur auf Windows 7 bringt es eine Fehlermeldung, die eine Berechnung unterbindet und das Tool somit nutzlos macht.

1.3 Detailanalyse des Tools

Beim Öffnen des Tools, zeigt das Fenster zuerst eine leere Anzeige an. Dies ist der erste Punkt der suboptimal gelöst ist, da der User nicht weiß, was er jetzt machen soll oder wie die Funktionalität des Tools jetzt benutzt werden soll. Des Weiteren gibt es in der GUI eine Menüleiste, in der „Datei“, „Fenster“, „Pfad“ und „Info“ aufgerufen werden können. Die Datei Funktion beinhaltet die Möglichkeiten, Dateien zu laden oder zu speichern und das Tool zu beenden. Was hierbei auch aufgefallen ist, dass beim Laden oder Speichern der Daten alle Dateitypen akzeptiert werden. Dies kann allerdings zu dem Problem führen, dass ein Benutzer Dateien, wie .exe Dateien, laden kann, die das Tool nicht verarbeiten kann. Was ein weiteres Problem darstellt. Über die „Fenster“ Funktion haben wir die Möglichkeit, ein „Eingabefenster“, „Ausgabefenster“ und ein „Visualisierungsfenster“ zu öffnen. Bevor wir näher auf die Fenster eingehen, dass die beiden Lösungsfenster ohne vorgehende Berechnung leer geöffnet werden können. Da diese dann keinen Zweck erfüllen ist diese Funktion auch eher hinderlich in der Usability. Außerdem gibt es keine Möglichkeit die Fenster nach dem Öffnen wieder zu schließen was die Oberfläche schnell unübersichtlich machen kann.

Das erste Fenster, das Eingabefenster, wurde so visualisiert:

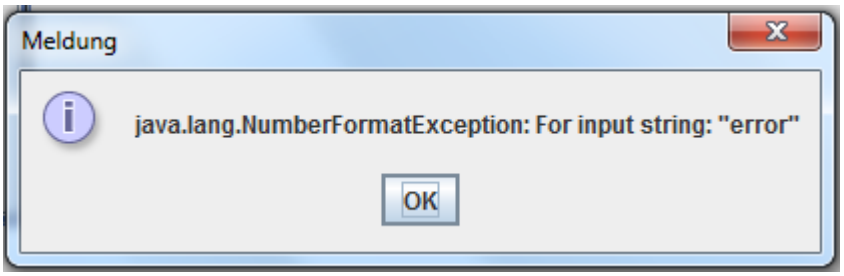


Bei der Anpassung der Anzahl der Reihen und Spalten muss man neue Eingaben mit der Enter Taste bestätigen, da ein „herausklicken“ aus dem Fenster hier nicht ausreicht. Dies sollte auch anders gelöst werden. Das Hauptproblem an diesem Fenster ist allerdings, dass wir keinen „Berechnen“ Button sehen können, dieser ist auf Grund der automatischen Zentrierung des Fensters nicht

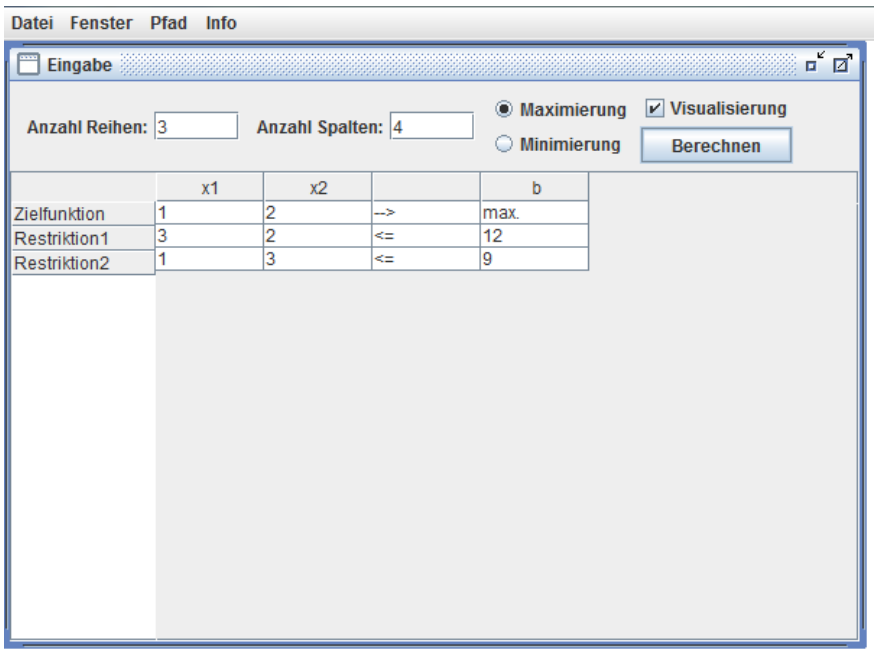
sichtbar. Das Fenster muss neu skaliert werden um den Button rechts neben „Maximierung“ sichtbar zu machen, was sehr verwirrend ist.

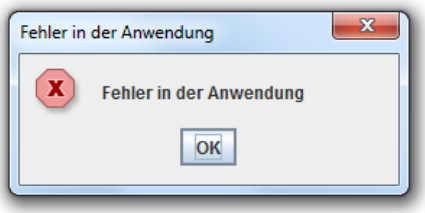
Nach der Berechnung, erfolgt die Ausgabe in einem Lösungsbaum und, nach Wunsch, in einem Lösungsraum Diagramm. Diese Oberflächen haben außer der Problematik mit des Schließens des Fensters keine weiteren Probleme gezeigt. Der Algorithmus funktioniert auch und liefert die richtigen Ergebnisse die auch korrekt dargestellt werden. Eine weitere Funktion die noch möglich ist, ist die Funktion, des setzten des Solverpfades. Dies ist über den „Pfad“ Reiter in der Menüleiste möglich.

Die Funktionalität des Tools war auf Windows XP und Windows Server 2003 gegeben, liefert allerdings bei der Ausführung auf Windows 7 folgende Fehlermeldung:



Dies erscheint wenn man eine Berechnung ausführen möchte. Ist also nicht lauffähig. Als Basis für diese Aussage wurde die Berechnung des Standard Beispiels gewählt. Ebenso erscheint folgende Fehlermeldung:





2. Fehleranalyse

Die Fehleranalyse war zu Beginn etwas schwierig, da man nicht genau wusste wo man ansetzen soll. Der einzige Unterschied von Lauffähig zu nicht lauffähig war das Betriebssystem. Dies sollte allerdings kein K.O. Kriterium sein, wodurch wir ausgegangen sind, dass das Tool auf jedem System laufen könnte. Da sonst das ganze Unterfangen zum Scheitern verurteilt gewesen wäre. Weitere potentielle Faktoren waren noch:

- ➔ Bit Version des Betriebssystems
- ➔ Hintergrundprozesse
- ➔ Entwicklungsumgebung -> Neukompilierung
- ➔ JFreeChart Version (Plug-In für Visualisierung)
- ➔ Java Version

Um nach dem Ausschlussverfahren vorzugehen, hat man beim Betriebssystem schon einmal die Möglichkeit einen Kompatibilitätsmodus zu verwenden. Dies ermöglicht uns, das Tool einmal auf dem Server auf dem es läuft und auf unseren Rechnern in verschiedenen Einstellungen zu starten. Dies umfasst mehrere Versionen von Windows und verschiedene Bit Versionen. Dabei zeigte sich, dass es nicht an der Version des Systems liegt. Das war daraus zu schließen, dass das Tool immer korrekt ausgeführt wurde wenn es auf dem Server lief, auf unseren Rechnern mit denselben Einstellungen allerdings nicht.

Der nächste Schritt war, die Hintergrundprozesse die stören könnten zu beenden. Da auf dem Server nur die nötigsten Prozesse laufen. Dies haben wir versucht, indem wir Windows bei uns im Abgesicherten Modus gestartet haben. Darin sind alle Prozesse deaktiviert außer die essentiellen. Durch das starten von Java haben wir so dieselbe Umgebung wie auf dem Server geschaffen. Dies führte allerdings zum selben Ergebnis, dass das Tool bei uns immer noch nicht korrekt ausgeführt wurde.

Eine Neukompilierung des Tools, bzw. das direkte Ausführen des Tools aus der Entwicklungsumgebung, brachte allerdings auch keine Verbesserung.

Ein weiterer Schritt den wir noch versucht haben, war das Einbinden eines aktualisierten Plug-Ins. Das verwendete jFreeChart Plug-In wurde eingebunden, um die grafische Lösung des Dakin zu visualisieren. Beim Einbinden einer neuen Version fiel schon auf, dass die Entwicklungsumgebung Fehler warf. Beim genaueren Hinsehen, wurden veraltete Teile des jFreeChart benutzt, die heute nicht mehr unterstützt werden. Hier war eine komplette Neuentwicklung notwendig, was allerdings später behandelt werden soll. Dies war nun allerdings als Fehlerquelle auch auszuschließen, da es mit der alten Version laufen sollte.

Als letztes blieb nur noch ein Unterschied der Java Umgebung. Das Tool wurde unter der Umgebung Java 1.4 programmiert. Wir programmieren in 1.7. Ein Vergleich der beiden Versionen mit Filterung auf die verwendeten Komponenten wäre allerdings zu intensiv in der Arbeit gewesen, da wir hier von tausenden Seiten Dokumentationen reden. Die einzige Möglichkeit die noch blieb, war der Debugg-Modus von Eclipse.

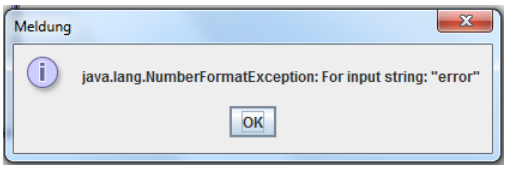
➔ Dieser brachte dann auch die Lösung. Das Problem war, dass eine Änderung in der Java Version ein ganz bestimmtes Attribut Case sensitive gemacht hatte. Genauer gesagt ist dies in der alten Version die Fehler-Nachricht „ERROR“, die in einem Objekt des Typs Vector() in Großbuchstaben abgelegt wird und später die Ursache für den Fehler ist. Diese Fehler-Nachricht muss jetzt allerdings in der neuen Version komplett in Kleinbuchstaben geschrieben werden, da sonst die Überprüfung mit dem „==“-Operator nicht funktioniert und das Tool so den falschen Weg im Algorithmus geht.

3. Fehlersuche und Bugs – Technische Ebene

In diesem Kapitel werden die technischen Maßnahmen der Fehlersuche und Bug-Behebungen näher erläutert und dargestellt. Um die Verbesserungen und wieder Inbetriebnahme des Tools möglich zu machen, haben wir ein neues Projekt in „Eclipse Indigo (Service Release 2)“ mit den Klassen des Dakin2003, welche auf dem Merkur-Laufwerk unter BESF vorhanden waren, angelegt. So konnten wir zusätzlich die nötige „JFreeChart-0.9.4“-Bibliothek (welche auch auf dem Laufwerk abgelegt war) einbinden und die Testversuche beginnen. Nachdem auf diese Weise das ganze Projekt zum Bearbeiten wierderhergestellt wurde, konnte das Programm zum Testen ausgeführt werden. Die Oberflächenfunktionen (mit den verschiedenen Oberflächenproblemen und Bugs) waren soweit alle funktionsfähig. Allerdings ließ sich die Berechnung über den LP-Solve nicht ausführen und es wurde eine Fehlermeldung geworfen.

3.1 Fehlersuche – Exception-Meldung

Die Fehlersuche war anfangs sehr ungeplant da man mit der Fehlermeldung bei Drücken des „Berechnen“- Buttons leider nicht viel anfangen konnte.



Der Code in den wichtigen Algorithmen des Tools ist sehr komplex. Deshalb war dazu eine Lange Zeit des Einlesens nötig, um zu verstehen warum genau an dieser Stelle die Exception geworfen wird. Wie man der Meldung entnehmen kann, ist es eine „NumberFormatExeption“. Durch den Debug-Modus in Eclipse stellte sich heraus, dass das Problem in der Klasse „TreeDakin.java“ anfällt. Hier sollte der String „error“ fälschlicherweise in ein

```
87              // das erste Element des Ergebnisvektors (der Zielfunktionswert)
88              // als float-Wert in das tmpArray eintragen
89      tmpArray[0]= new Float((String) ergebnisVektor.get(0));
```

Float-Array gerschrieben werden. In der blau markierten Zeile ist der „Wurf“ der Exception markiert. Jetzt musste herausgefunden werden, warum dies in den früheren Versionen gar nicht bis zu dieser fälschlichen Zuweisung eines Strings in ein Float-Array gelangen konnte. Nach dem verschiedenen Setzen von Systemausgaben zum Testen bis wo hin das Programm ab dem Drücken des „Berechnen“-Buttons läuft und wo eventuell eine Abfrage auf den String nicht abgefangen wird, kam man auf die ersten Ergebnisse. In der Klasse „LPsolve.java“ wird an der Stelle an der eine Exception gefangen wird, die Fehler-Nachricht „error“ (in kleingeschriebenen Buchstaben) in ein Objekt des Typs „Vectort()“

```
149         } catch (IOException e) {
150             // System.err.println("exeption-error:" + e);
151             _ERROR_MSG += "\nIO error: " + e;
152             myStringVector.add(0,"error");
153             myStringVector.add(1,_ERROR_MSG);
```

geschrieben. Dies war das einzige kleingeschriebene „error“, das sich in den Klassen befunden hat, also war es genau der „input string“, den wir der Fehlermeldung entnehmen können. Aufgrund dieser Information haben wir die verschiedenen Klassen noch weiter nach Hinweisen auf die Error-Nachricht und den Gebrauch des Vectors untersucht.

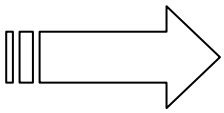
```
77          if (ergebnisVektor.get(0) == "ERROR")
78          {
79              optimal=false;
80              return;  // abbrechen
81          }
```

Genau dies stellte sich schlussendlich auch als richtige Fehlerquelle heraus. Kurz vor der missglückten Zuweisung eines Strings in ein Float-Array (in der schon oben genannten Klasse „TreeDakin.java“), sollte der Wert des Vectors an der Stelle „0“ mit dem String „ERROR“ (in GROßBUCHSTABEN) verglichen werden. Dort sollte die Methode verlassen und return zurückgegeben werden, wenn in dem „ergebnisVektor“ an der Stelle „0“ sich die Fehler-Nachricht „ERROR“ befindet. Der Vergleich von „error“ mit „ERROR“ mit dem „==“-Operatror wird an dieser Stelle aber immer als false gekennzeichnet, was die Methode weiter fortgesetzt hat und somit die Zuweisung eines Strings in ein Float-Array zum Abbrechen brachte.

3.1.1 Fehlerbehebung – Exception-Meldung

Nach dem Ersetzen des kleingeschriebenen „error“ in „ERROR“ mit Großbuchstaben (in der Klasse LPsolve.java), wurde diese Abfrage „true“ und der Programmablauf wurde korrekt fortgesetzt was zur richtigen Berechnung der Eingabe führt.

```
149         } catch (IOException e) {
150             // System.err.println("exeption-error:" + e);
151             _ERROR_MSG += "\nIO error: " + e;
152             myStringVector.add(0,"error");
153             myStringVector.add(1, _ERROR_MSG);
```

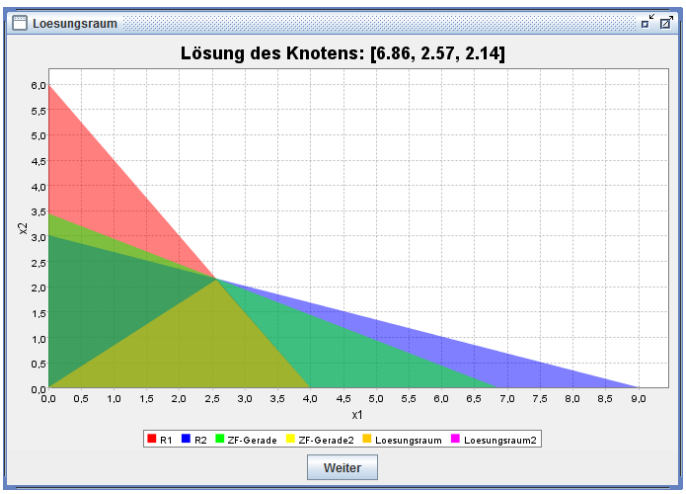


```
149         } catch (IOException e) {
150             // System.err.println("exeption-error:" + e);
151             _ERROR_MSG += "\nIO error: " + e;
152             myStringVector.add(0,"ERROR");
153             myStringVector.add(1, _ERROR_MSG);
```

Nach einiger Recherche über den Vergleichsoperator „==“ in den früheren Java-Versionen, konnte auf die Case-Sensitivtät leider keine eindeutige Antwort gefunden werden.

3.2 Fehlersuche - Visualisierung

Nachdem dieser Fehler behoben wurde, der LP-Solve-Solver richtig angesteuert und die Darstellung des Baumes funktionierte, stellte sich das nächste Problem heraus. Die Visualisierung wurde nicht richtig dargestellt. Die Linien wurden im Diagramm zwar eingezeichnet, allerdings wurde der ganze Raum unterhalb dieser bis zur X-Achse ausgefüllt.



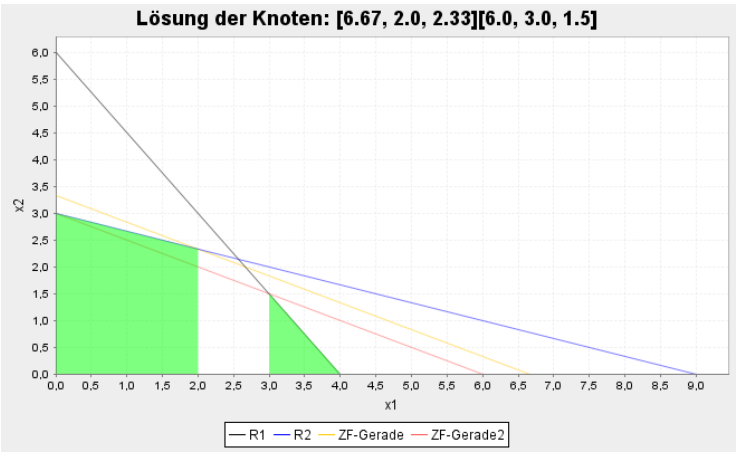
Somit konnten sich die Restriktionen und die Zielfunktion (als Geraden) nicht mehr von den Lösungsräumen abheben.

Die über JFreeChart dargestellten Diagramme funktionierten nicht richtig.

Nach dem die Funktionsweise des Aufbaus der Restriktionen und der sog. Polygone (der Lösungsräume) überschaubar in der Klasse „Visualisiere.java“ nachvollzogen wurde, hat sich das Problem herausgestellt. In den alten JFreeChart-Bibliotheken war es möglich, verschiedene Restriktionen in „Series“ aufzubauen und zusätzlich Polygone, also Flächen, auch über „Series“ darzustellen. Dabei wurden die einzelnen „Series“ automatisch als Flächen dargestellt, wenn mehr als 2 Koordinaten darin angefügt wurden. Wenn nur 2 Punkte hinzugefügt wurden, wurden diese als Geraden gezeichnet. Diese Funktion scheiterte aber in der Rekonstruierung unseres Projektes.

3.2.1 Fehlebehebung - Visualisierung

Aus diesem Grund entschieden wir uns erst einmal für das Einbinden einer neueren Version der JFreeChart-Bibliothek ➔ Version 1.0.14 und zusätzlich zur Nachbildung dieser Geraden und Flächen mit der Grundlage der neuen Bibliothek. Alleine nach dem Einbinden dieser neueren jar-Datei konnten die Importe der verwendeten Klassen nicht mehr gefunden werden. Diese befinden sich nach der Erneuerung in anders angeordneten Pfaden der Bibliothek. Als die verschiedenen Imports nach der Suche nach den richtigen Klassen umgestellt wurden, stellte sich auch heraus, dass die Darstellung rein über eine „AreaXYChart“ nicht mehr funktionieren könnte. Nach langer Recherche mit Hilfe von verschiedenen Dokumentationen rund um die JFreeChart Verwendung und dem gescheiterten aufzeichnen der Flächen über „XYAreas“, haben wir uns für die Darstellung in zwei Schritten entschieden. Restriktionen und Polygone in einem Plot über eine bestimmte Chart aufzuzeichnen ist in den neuen Versionen leider nicht mehr so einfach möglich. Darum haben wir die Restriktionen, welche in der „SeriesCollection“ gesammelt werden, genau mit dem alten Schema aus der Original-Klasse übernommen und stellen diese über einen sog. „XYPlot“ dar. Die Lösungsräume mussten aus diesen „Series“ entfernt werden. Die einzelnen Koordinaten dieser Flächen werden in ein double-Array zwischengespeichert und können so direkt an den Plot über ein Objekt der vordefinierten Klasse „XYPolygonAnnotation“ in der JFreeChart-Bibliothek angefügt werden. Was hierbei zu beachten ist, ist die Sortierung der Punkte im angefügten Array. Diese muss in einer Reihenfolge ausgehend von irgendeinem Punkt im oder entgegen dem Uhrzeigersinn erfolgen. Ansonsten kann die Darstellung fehlschlagen und es werden falsche Räume dargestellt, da die integrierten JFreeChart Bibliothek immer den Raum innerhalb der außenherum liegenden Punkte füllt.



Somit mussten hier noch 2 Funktionen hinzugefügt werden, damit diese Punkte richtig geordnet werden. Dabei bestand die Herausforderung darin, einen solchen Sortieralgorithmus möglich zu machen und zu finden. Nach sehr langer Internetrecherche

sind wir auf eine Möglichkeit der Uni Paderborn gestoßen. In unserem Fall handelt es sich um die Implementierung von „einfachen Polygonen“.

http://www2.cs.uni-paderborn.de/cs/ag-monien/PERSONAL/OBELIX/Lehre/EffAlg/WS02/0502a.pdf

In diesem Skript konnten wir einen Algorithmus in C++ finden, der ausgehend von einem Ausgangspunkt, einer Reihe von weiteren Punkten die Steigung und Distanz zum Ausgangspunkt berechnet. Dieses Codefragment haben wir für unser Tool in Java umgeschrieben und angepasst, damit im folgenden Schritt mit einem Bubblesort nach dem Kriterium der Steigung sortiert werden kann. Nach diesen 2 Durchführungen bekommt man ein Array mit den verschiedenen Punkten, gegen den Uhrzeigersinn sortiert, zurück und das Plotten funktioniert erfolgreich.

3.2.2 Fehlerbehebung – Visualisierung - Darstellungsbug

Nach verschiedenen parallelen Testläufen des neu aufgesetzten Dakin2003 und dem „alten“ Dakin2003 auf dem Hochschulserver stellte sich ein zusätzliches Problem heraus. Nach mehrfachem Lösung von gleichen oder auch verschiedenen Restriktionen/ Zielfunktionen, werden wenn im Visualisierungsfenster nicht mit dem Button „weiter“ alle verschiedenen Lösungen der Knoten angeschaut wurden, beim erneuten Lösen „alte“ Knoten noch aus der vorherigen Berechnung angezeigt. Dies wurde mit 2 weiteren Methoden mit dem Test auf die erste Visualisierung nach dem Start des Tools und der Methode mit dem Test auf eine neue Berechnung/ Visualisierung behoben. Dabei werden bei jeder neuen Berechnung die vorherigen Werte der Knoten und Lösungsräume „geleert“ und neu gefüllt. So werden immer nur genau die Lösungen der Knoten der aktuellen Berechnung im Visualisierungsfenster dargestellt.

Fazit zur Fehlersuche und –behebung:

Einige Fehler konnten leider sogar nach Rücksprache mit Herr Professor Johner und Herr Professor Schimkat nicht genau geklärt werden. Die Case-Sensitivität variiert (aus unerklärlicher Weise) beim Abgleich mit den „==“-Operator in den verschiedenen Java-Entwicklungsversionen.

Das Einbinden der neuen JFreeChart-Bibliothek ermöglicht noch weitere Funktionen zur besseren Darstellung der Visualisierung und trägt zum „Update“ unseres Tools bei. Die einzigen Mängel, die auf ein eventuelles Folgeteam zur Optimierung noch anfällt ist die neue Darstellung des Lösungsraumes bei der Minimierung. Hierbei müsste in den Algorithmus der Punktberechnung eingegriffen werden. Es gibt Probleme beim Visualisieren des Lösungsraumes der oberhalb der Restriktionen liegt und ins unendliche geht. Das Plotten mit der Funktion der „XYPolygonAnnotation“ füllt NUR die Räume der angegebenen Punkte und nicht bis „ins Unendliche“ der X- und Y-Achse.

Diesen Fall haben wir in unserem Commitment als eine Funktion abgegrenzt, die als Sonderfall gilt und somit für das Nachfolgeteam vollends ausdokumentiert und als To Do angegeben wird. Die Hauptaufgabe lag dabei auf der Lauffähigkeit des Tools auf Windows 7, was mit der Fehlerbehebung der „error"-s vollständig erfolgte und der Neuimplementierung der aktuellen Version des JFreeCharts sogar übertroffen wurde.

4. Optimierungen

4.1 Benutzeroberfläche und -anwendung

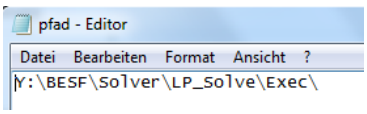
Nach den schon oben genannten vielen Tests, die es benötigt hat das Tool lauffähig zu machen, haben wir uns dafür entschieden die Oberfläche neu aufzusetzen. Die internen Fenster führen immer wieder zu Unübersichtlichkeiten und Verwirrung, darum haben wir für die neue Darstellung 3 Tabs ausgewählt. Hierbei sind oben drei Reiter zu sehen:

- ⇨ Die Eingabetabelle,
- ⇨ Die Ausgabe des Lösungsbaumes &
- ⇨ Die Ausgabe für die Visualisierung.

Da bei mehr als 2 Restriktionen die Visualisierung nicht durchgeführt werden kann, können wir das Tool so steuern, dass immer nur die Registerkartenreiter aktiv und auswählbar sind, die Ergebnisse darstellen.

Im selben Zug gibt es auch keine Probleme mehr mit der Suche nach dem „Berechnen"-Button, der in der alten Version aufgrund der Frame-Formatierung leider ausgeblendet wurde und nur schwer zu finden war wenn man das Tool noch nie benutzt hat.

Außerdem wird nun bei Programmstart sofort das Eingabefenster angezeigt um gleich Werte eingeben zu können. In der alten Version mussten die verschiedenen Fenster (Eingabe, Ausgabe und Visualisierung) erst ausgewählt werden, damit die internen Fenster auf der Oberfläche angezeigt wurden. ➔ Diese Auswahlmöglichkeiten für die Fenster wurden aus der Menübar entfernt, da diese nun in den Registerkartenreiter direkt im Hauptfenster erreichbar sind.



Zusätzlich haben wir die Funktion „Pfad" in der Menü Bar angepasst, damit nur noch .exe Dateien ausgewählt werden können (!!!Das Tool funktioniert AUSSCHLIEßLICH mit der „lp_solve.exe"-Datei!!!). Damit dieser Pfad direkt im Programm auch ohne neue Kompilation des Tools geändert werden kann und nicht jedes Mal ausgewählt werden muss, wird dieser nun aus einer direkt „neben" (auf der selben Pfadebene) des Tools liegenden .txt-Datei jedes Mal bei Programmstart ausgelesen und kann somit bei einem Umzug an andere Orte immer wieder beliebig geändert werden. (Realisiert wurde dies mit einer FileReader-Funktion, welche die .txt-Datei ausliest und den Pfad an der richtigen Stelle im Tool einsetzt.) Der aktuelle Pfad wird in der Hauptansicht zur Kontrolle angezeigt.

In der Eingabesicht wurde die Eingabetabelle mit den Ansichtsbugs nun erneuert und korrigiert. Der weiße freie Raum wurde entfernt und die Tabellenzeilen auf dieselbe Größe und Höhe angepasst. Hinzu kommt die Behebung des Bugs auf den in einer Meldung beim Programmstart hingewiesen wird. Nach dem Bearbeiten der Werte in der Eingabetabelle musste jeweils „enter“ gedrückt werden, damit diese auch korrekt übernommen wurden. Wenn dies vom Benutzer vor dem Starten der Berechnung nicht durchgeführt wurde, wurde in der letzten bearbeiteten Zelle der Wert übernommen, der zuvor darin vermerkt war. In der neuen Version werden die Daten der Tabelle vor der Ausführung der Berechnung gespeichert und übernommen, damit die korrekten Werte an den Solver übergeben werden können. Außerdem ist es jetzt möglich eine Zelle anzuklicken und beim Eingeben der gewünschten Zahl den bisherigen Wert direkt zu überschreiben. In der alten Handhabung musste dieser Wert erst (mit Backspace oder Entfernen) gelöscht werden, damit ein neuer eingegeben werden konnte und die Zahlen nicht an den alten Wert angehängt wurden.

4.2 Erneuerungen und Verbesserungen bei der Visualisierung

Wie oben schon beschrieben ist, wurde die Visualisierung mit der neuen JFreeChart-Version-1.0.14 komplett neu realisiert. Diese Funktionen begrenzen sich alleine auf die Klasse „Visualisiere.java“. Die Wiederholungsbugs von vorherigen Berechnung wurden behoben und die Darstellung der Diagramme wurde auf nur den positiven Bereich eingestellt (minimaler Wert der x-Achse = 0, maximal der maximale Wert der Restriktionen) → In der alten Version startete diese Ansicht zum Teil im negativen Bereich.

Manche Einstellungen konnten zum Teil übernommen werden, allerdings mussten die Lösungsraum-Koordinaten komplett neu aus den Funktionen ausgelesen und in den benötigten Objekten zur späteren Darstellung abgelegt werden.

Zudem kann man nun über den „XYLineAndShapeRenderer“ (im Code „renderer“) verschiedenste Einstellungen für die Darstellung der Geraden vornehmen. Zum Beispiel können so die Farbe angepasst oder die Linienpunkte zur Übersichtlichkeit ausgeblendet werden.

4.3 Sonstige Erweiter- und Verbesserungen

Zu den oben genannten Punkten wurden noch verschiedene Fehlermeldungen beim Auslesen und setzen des Pfades hinzugefügt um die Probleme schneller erkennen zu können. Außerdem wurde die Eingabemöglichkeit der Operatoren auf „<“, „<=“, „>=“ und „>“ beschränkt. Das „=“ wurde entfernt damit der Nutzer nur „sinnvolle“ Eingaben für dieses Tool tätigen kann.

Außerdem zur Information: Die Eingabe der Restriktionen und Parameter kann prinzipiell im beliebiger Anzahl variieren. Die Visualisierung funktioniert allerdings (standardmäßig seit der Entwicklung des Tools im Jahre 2003) nur mit maximal 2 Restriktionen und jeweils 2 Parameter.

Zusätzlich wurden die „unnötigen“ und unbenutzten Klassen aus dem Projekt entfernt, damit nicht fehlerhafte Klassen mitkompiliert werden müssen für die ausführbare Jar-Datei.

5. Weitere Potentiale

Weitere Potentiale, hat das Tool definitiv noch. Leider konnten wir viele Ideen die wir hatten, im Rahmen der drei ECTS leider nicht umsetzen. Einige Ideen die wir allerdings hier noch für ein eventuelles Nachfolger Team dokumentieren wollten:

1. Dynamische Visualisierung der grafischen Lösung. Das soll heißen, dass die Grafen dynamisch eingeblendet werden und danach erst der lösungsraum erscheint. Dies soll das Lösungsverfahren allgemein nachvollziehbarer machen.
2. Eine Hilfe Funktion zum Lösungsbaum Verfahren. Dies soll nicht unbedingt, kann aber, durch eine visuelle Hilfe geschehen. Eine Art dynamischen Aufbauens der Lösung. Eher aber durch eine textuelle Erklärung, die beschreiben soll, wie der Baum jetzt aufgebaut wurde und wie das Verfahren generell funktioniert.
3. Verweis auf die Schnittstelle des MightyLP → Mit dieser Neueinführung könnten später verschiedene Solver schnell und einfach mit nur einer (einheitlichen) Schnittstelle und Klasse angesteuert werden.
4. Genauere Darstellung der Lösungsraum-Visualisierung bei der Minimierung. Der bisherige Plot lässt noch keine Räume „ins Unendliche“ zu. Hierbei müssten die Punkte für die „XYPolygonAnnotation“ im Berechnungsalgorithmus soweit angepasst werden.

Es gibt sicher auch noch mehr Optimierungspotential, mit dem wir uns allerdings nicht mehr beschäftigen können, da es in dieser Veranstaltung primär um die Migration auf das Betriebssystem Windows 7 ging.

6. Migration

Die Migration des Tools unter Windows 7 erfolgt relativ einfach. Das Tool wird mit dem passenden Link über die Methodenbank aufgerufen. Der dort hinterlegte Pfad für den Solver sollte immer stimmen sonst muss er gesetzt werden. Wenn man nun das Tool auf den eigenen Rechner kopieren möchte, muss man entweder den Pfad so setzen, dass er auf das Merkur Laufwerk direkt auf den Solver referenziert oder den Solver gleich mit kopieren. Dann muss man nur den Pfad so anpassen, dass es den Solver auf der Festplatte ansteuert und kann auch offline arbeiten.

7. Fazit

Insgesamt haben wir in unserem Projekt alle Punkte aus dem Commitment abgearbeitet und erfolgreich fertiggestellt. Das Tool ist Lauffähig auf Windows 7 und 8 (32- wie auch 64Bit). Außerdem haben wir zusätzlich noch eine Tool-Hilfe im HTML-Format mit eingebaut, damit jeder Nutzer sich einlesen und die Funktionen mit Screenshots erklärt bekommen kann. Das genaue Kommentieren der Funktionen und Methoden unseres Codes und auch zum Teil des alten Codes erfolgt direkt in den verschiedenen Klassen des Projektes. Der ungefähre Arbeitsaufwand für beide Teammitglieder beläuft sich (exklusive der Anwesenheitsdauer in den Vorlesungen und den vorbereitenden Tool-Präsentationen zu Beginn des Semesters) in den aufgelisteten Aufgabengebieten auf insgesamt 55 bis 65h pro Person. Die verschiedenen Aufgabengebiete:

- Fehleranalyse
- Fehlerbehebung
- Einbinden neuer Bibliotheken
- Verbesserungen der Logikabläufe und Abfangen der Fehler bei der Visualisierung
- GUI-Oberflächen-Überarbeitung
- Sonstige zusätzliche Ablaufverbesserungen und –änderungen

8. Quellen- und Hilfeverzeichnis

Skript für den Sortieralgorythmus der Uni Paderborn:

http://www2.cs.uni-paderborn.de/cs/ag-monien/PERSONAL/OBELIX/Lehre/EffAlg/WS02/0502a.pdf

http://wwwdh.informatik.uni-erlangen.de/IMMD8/Lectures/Algo1/Skript/1-auf-1/algo1-18-1.pdf

Bubble-Sort-Beispiel der Uni Magdeburg:

http://wwwiti.cs.uni-magdeburg.de/iti_db/algoj/code/algoj/kap5/Sort.java

JFreeChart Hilfeseiten und Foren:

http://www.tutorials.de/java/317823-java-chart-jfreechart-groesse-anzeigebereich.html

http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/annotations/XYPolygonAnnotation.html

http://www.java2s.com/Code/Java/Chart/CatalogChart.htm

http://www.jfree.org/jfreechart/api/javadoc/org/jfree/data/xy/XYSeries.html

PolygonAnnotation-Demo:

http://read.pudn.com/downloads145/sourcecode/java/jsp/633794/JfreeChart/src/org/jfree/chart/annotations/XYPolygonAnnotation.java__.htm

Sonstige Hilfeseiten zur Programmierung:

http://openbook.galileodesign.de/javainsel5/javainsel15_018.htm#Rxx747java150180400054A1F025100

http://www.java-forum.org/awt-swing-javafx-swt/88880-jtable-edit-cell-edit-end-erzwingen.html

http://www.java-blog-buch.de/d-jtable-momentan-editierte-zelle-abfragen/

http://www.java-forum.org/java-basics-anfaenger-themen/105526-sortieren-polygons.html

http://www.java2s.com/Tutorial/Java/0240__Swing/SwingsClientPropertiesSummaryTable.htm