Hochschule Konstanz Technik, Wirtschaft und Gestaltung
Fakultät Informatik/Wirtschaftsinformatik

GlpkGui

Bericht zur Erstellung der Software GlpkGui im Fach Anwendungen des OR

6. Semester Wirtschaftsinformatik

von

Sebastian Krüger

sekruege@htwg-konstanz.de

Wintersemester 2009/2010

Inhaltsverzeichnis

1 Motivation

Für das Wahlfach *Anwendungen des OR* sollte eine bestehende Applikation aus der *Methodendatenbank* verbessert, oder eine neue Software zum Thema *Operations Research und Linear Programming (LP)* erstellt werden. Da die häufig verwendeten *Solverprogramme* zum Lösen eines LP-Problems wie *XA oder LPSolve* bei Einbettung in eine Software relativ umständlich zu steuern sind, sollte die neu zu erstellende Anwendung das Solverprogramm nicht in einem externen Prozess aufrufen, sondern dessen Funktionen *nativ* einbinden. Außerdem sollten die bestehenden LP-Programme mit einem bisher unbenutzten Solverprogramm verglichen werden können.

Nach Internetrecherche zu Quell-offenen, bzw. freien Solverprogrammen wurde das *GNU Linear Programming Kit (GLPK)* ausgewählt. Es ist für verschiedene Betriebssysteme verfügbar und in *ANSI C* verfasst. Da es unter der *GNU Public License (GPL)* veröffentlicht wird, kann es angepasst werden, bzw. es können für die verschiedensten Programmiersprachen native Unterstützungen implementiert werden.

Um das GLPK mit anderen Solverprogrammen vergleichen zu können, sollte mit der zu erstellende Anwendung, ähnlich wie mit dem Programm *Power-LP* aus der Methodendatenbank, beliebige LP-Probleme erstellt, bearbeitet und gelöst werden können. Die Anwendung sollte also eine graphische Oberfläche bieten und LP-Probleme mit Hilfe des GLPK lösen können.

2 GNU Linear Programming Kit

Das GLPK wurde zum Lösen von großen LP-, MIP- *(Mixed Integer Programming)* und verwandten Problemen entwickelt. Die Entwickler legen Wert darauf, dass es sich als Programmbibliothek nutzen lässt. In der Distribution wird aber auch ein Standalone-Solverprogramm mitgeliefert. Es existieren Komponenten um ein LP-Problem mit verschiedene Methoden zu lösen:

- *Primal und Dual Simplex:*

  Die Simplex-Methode ist ein weit verbreiteter und häufig benutzter Algorithmus zum Lösen von LP-Problemen.

- *Primal-Dual Interior-Point:*

  Die Interior-Point Methode kann für große LP-Probleme effizienter uns schneller als die Simplex-Methode sein.

- *Branch-And-Cut*

  Der Algorithmus wird verwendet, um LP-Probleme mit Ganzzahligkeitsbedingungen (MIP) lösen zu können.

Probleme in folgenden Dateiformaten können geöffnet werden:

- *Fixed MPS Format (s. 3.1)*
- *Free MPS Format*
- *CPLEX LP Format*

Außerdem lassen sich die *Karush-Kuhn-Tucker-Bedingungen (KKT)* zur Optimalität einer Lösung ausgeben, sowie eine *Sensibilitätsanalyse* durchführen.

3 GlpkGui

Genau wie das Programm Power-LP sollte die neue Applikation einen graphischen Editor zur Erstellung und Bearbeitung eines beliebigen LP-Problems bieten und unter *Windows*, d.h. auf den Laborcomputern der *HTWG Konstanz* lauffähig sein. Als Programmiersprache wurde deshalb *C#* und dessen *.NET* Laufzeitumgebung gewählt. C# ist eine einfach zu erlernende objektorientierte Sprache, mit der unter der Windows Entwicklungsumgebung *Microsoft Visual Studio* komfortabel gearbeitet werden kann. Vor allem für die Entwicklung von graphischen Benutzeroberflächen bietet die Umgebung gute Unterstützung.

Ein wesentliches Merkmal für den Entwurf der Software war das Wegfallen der Solverkonfiguration im Vergleich zu Power-LP. Da das Programm mit GLPK als nativem Solver arbeitet sollte, würde das Setzen von Arbeits- oder Solverpfad entfallen. Dies sollte ebenfalls die Installation der Software auf verschiedenen Computer erleichtern. Um einen Vergleich der Solver von Power-LP mit dem GLPK zu ermöglichen, sollten die benutzbaren Dateiformate der neuen Anwendung kompatibel zu Power-LP sein. Außerdem sollte nur *ein* LP-Problem mit einer Programminstanz editierbar sein. In der folgenden Abbildung sind die GUI-Hauptelemente erläutert:

- **1:** Toolbar zum Öffnen und Speichern von Dateien, Lösen des LP-Problems, Skalieren des LP-Problems, Konfiguration des Lösungsalgorithmus
- **2:** Editierung von Matrix, Zielfunktion und B-Vektor
- **3:** Editierung von oberer Grenze, unterer Grenze und Ganzzahligkeit der Variablen
- **4:** Darstellung von LP-Lösung, MIP-Lösung, Interior-Point Lösung und Sensibilitätsanalyse

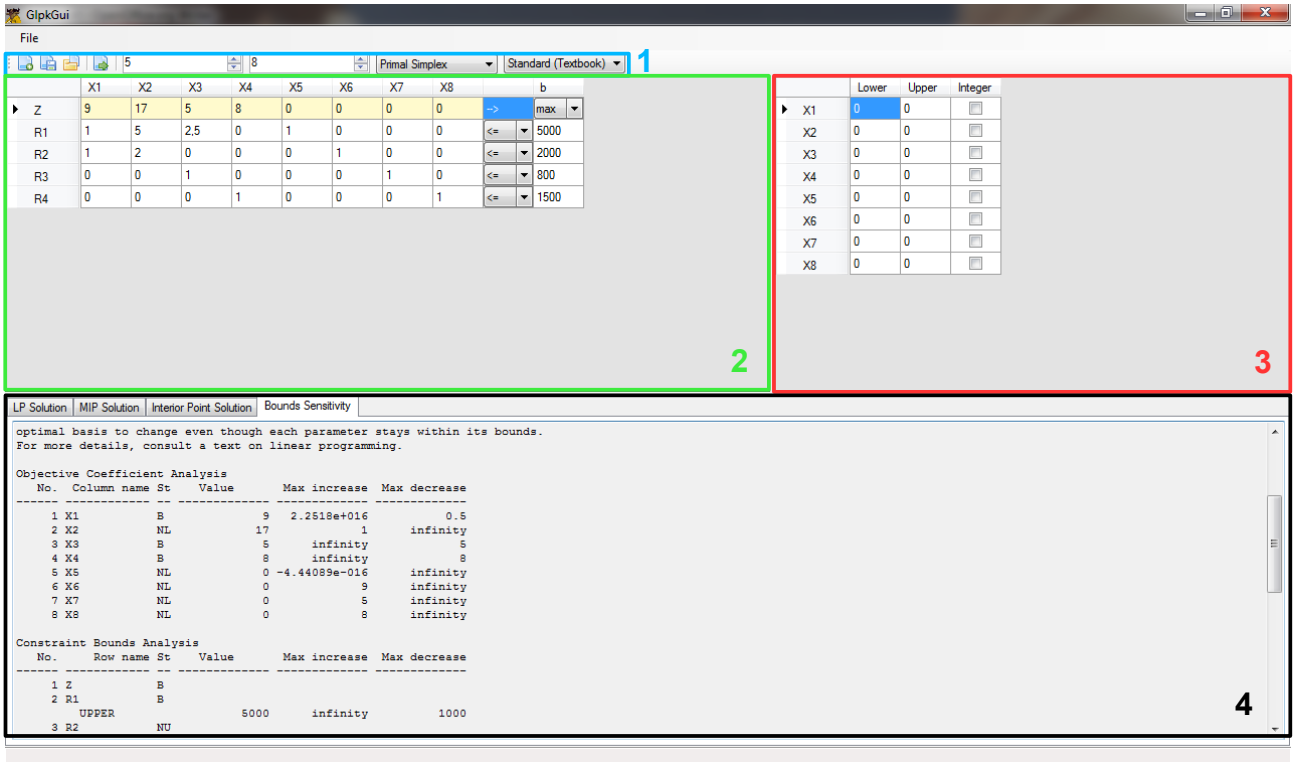


Abbildung 1: Screenshot der GlpkGui

Insgesamt wurden etwa *56* Arbeitsstunden für Studium des GLPK, Erlernen von C# und Implementierung, bzw. Entwurf der Software aufgewendet. Für die native Unterstützung des GLPK wurde eine bereits existierende Anbindung an C# *(GlpkSharp)* erweitert. Bei der Entwicklung des LP-Problem-Editors wurde darauf geachtet, dass keine falschen Eingaben gemacht werden können, wie z.B. die Eingabe von Buchstabenketten. Außerdem wurde das Setzen der Vergleichsoperatoren der Restriktionen durch eine *Combo-Box,* im Vergleich zu Power-LP, vereinfacht. Das Lösen des aktuellen LP-Problems erfolgt durch Druck auf den entsprechenden Button in der Toolbar. Dabei werden immer LP-, MIP- und Interior-Point-Lösung, sowie die Sensiblitätsanalyse neu erzeugt. Zwar werden die Ergebnisse in separate Dateien geschrieben und wieder eingelesen, weil dies durch das GLPK nicht anders möglich ist, jedoch bemerkt der Benutzer davon nichts und muss kein beschreibbares Verzeichnis angeben, da stattdessen einfach das temporäres Windows-Verzeichnis verwendet wird. Dies funktioniert auch mit eingeschränkten Benutzerrechten. In der Toolbar können außerdem die GLPK-Einstellungen für die Simplex-Methode geändert werden, das Ändern hatte sich in Tests aber nicht bemerkbar gemacht.

Um eine Kompatibilität zu Power-LP zu erreichen war es zusätzlich notwendig einen Parser für das LPI-Format zu implementieren. GLPK und Power-LP unterstützen zwar beide das Fixed MPS Format, jedoch können mit Power-LP diese Dateien teilweise nicht richtig interpretiert werden.

4 Fixed MPS Format

Das Format wurde in den 60er Jahren von *IBM* zur Dateneingabe für das mathematische Programmiersystem *MPS/360* entwickelt. Heutzutage findet es in Form von Textdateien weite Verbreitung bei LP-Software. Da eine MPS-Datei ursprünglich aus mehreren Lochkarten bestand, werden die einzelnen Zeilen einer MPS-Textdatei als *cards* bezeichnet. Dabei gibt es zwei Typen von cards:

- *indicator cards,* welche die Art der folgenden Daten festlegen
- *data cards,* welche Problemdaten beinhalten und in 6 Felder eingeteilt wird.

In einer MPS-Datei sollten folgende cards in Reihenfolge enthalten sein:

- NAME indicator card zur Definition des Problemtitels
- ROWS indicator card
  (data cards definieren die Zielfunktion, sowie die Restriktionen mit Vergleichsoperator als Konstanten)
- COLUMNS indicator card
  (data cards bestimmen die Spalten des LP-Problems mit Koeffizienten)
- RHS indicator card;
  (data cards bestimmen die Werte des B-Vektors)
- RANGES indicator card;
  (data cards können Grenzbereiche des B-Vektors bestimmen)
- BOUNDS indicator card
  (data cards bestimmen Grenzbereiche von Entscheidungsvariablen)
- ENDATA indicator card, zur Markierung des Endes der LP-Problem-Definition

```
* Problem:
* Class:      LP
* Rows:       3
* Columns:    2
* Non-zeros:  6
* Format:     Fixed MPS
*
NAME
ROWS
 N  Z
 G  R1
 G  R2
COLUMNS
    X1        R2                  25   R1                1000
    X1        Z          0.6000000238
    X2        R2                 100   R1                2000
    X2        Z          2.0999999046
RHS
    RHS1      R1                3000   R2                 100
ENDATA
```

Abbildung 2: Darstellung eines LP-Problems im Fixed MPS Format

5 LPI Format

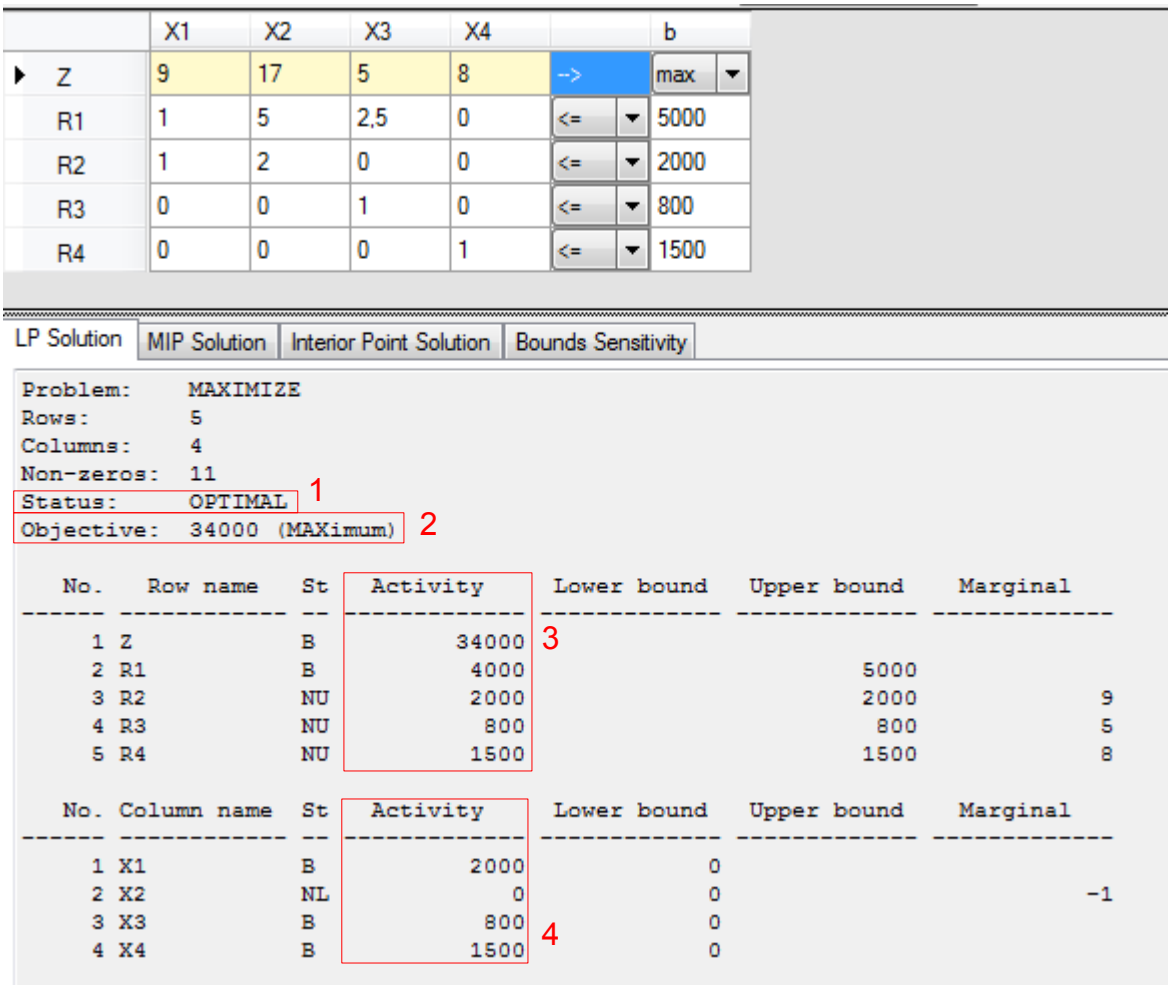
Das Format wird von Power-LP zum Speichern eines LP-Problems verwendet und wird als Binärdatei geschrieben. Die Größe eines Problems ist dabei beschränkt, es können maximal *75* Entscheidungsvariablen und maximal *100* Restriktionen gesichert werden. Eine LPI-Datei hat auf dem Datenträger immer die selbe Speichergröße (ca. 34 Kilobyte), da Speicherbereiche auch bei z.B. geringer Entscheidungsvariablenanzahl mit Null-Bytes aufgefüllt werden.

```
struct Tlpdata {
    char d1[2];
    bool maximize;      // 4
    char d2[1];
    short restr;        // 6
    short vars;         // 8
    char d3[408];       // 416
    float nb[75][101]; // 30716 (0x77F8)
    float zfb[100];     // 31116 (0x798C)
    char d7[308];       // 31424 (0x7AC0)
    float lbound[76];   // 31728 (0x7BF0)
    float hbound[76];   // 32032 (0x7D20)
    float zf[75];       // 32332
    char d5[1142];      // 33474 (0x82C2)
    short cc[100];      // 33674
    char d6[154];       // 33828 (0x8424)
    short integer[75];  // 33978
    // size 33978
};
```

Abbildung 3: LPI-Format als C-Struct

6 Beispiel: Aufgabe 3.42

In der Aufgabe wird ein einfaches LP-Modell mit 4 Entscheidungsvariablen und 4 Restriktionen aufgestellt, die sich aus den Kostenstellen für verschiedene Produkte ergeben. In der Zielfunktion soll der maximale Deckungsbeitrag ermittelt werden.



	X1	X2	X3	X4		b
Z	9	17	5	8	-->	max
R1	1	5	2,5	0	<=	5000
R2	1	2	0	0	<=	2000
R3	0	0	1	0	<=	800
R4	0	0	0	1	<=	1500

LP Solution | MIP Solution | Interior Point Solution | Bounds Sensitivity

```
Problem:    MAXIMIZE
Rows:       5
Columns:    4
Non-zeros:  11
Status:     OPTIMAL
Objective:  34000 (MAXimum)
```

1 2

No.	Row name	St	Activity	Lower bound	Upper bound	Marginal
1	Z	B	34000			
2	R1	B	4000		5000	
3	R2	NU	2000		2000	9
4	R3	NU	800		800	5
5	R4	NU	1500		1500	8

3

No.	Column name	St	Activity	Lower bound	Upper bound	Marginal
1	X1	B	2000	0		
2	X2	NL	0	0		-1
3	X3	B	800	0		
4	X4	B	1500	0		

4

Abbildung 4: Lösung von Aufgabe 3.42

Die Abbildung soll veranschaulichen, wie die Lösung von einem LP-Problem durch das GLPK zu Lesen ist. Im Gegensatz zu anderen Solvern wie z.B. LPSolve werden mehr Informationen ausgegeben:

- **1:** Status der Lösung (hier *OPTIMAL*, könnte aber z.B. auch *UNBOUNDED* sein)
- **2:** Wert der Zielfunktion
- **3:** Werte des B-Vektors
- **4:** Werte der Entscheidungsvariablen

Die folgende Abbildung zeigt die Sensibilitätsanalyse. Sie besagt wie die Grenzen der Variablen, die Restriktionsgrenzen sowie die Koeffizienten der Zielfunktion, bei gleich bleibender Basis geändert erden können.

Objective Coefficient Analysis

No.	Column name	St	Value	Max increase	Max decrease
1	X1	B	9	2.2518e+016	0.5
2	X2	NL	17	1	infinity
3	X3	B	5	infinity	5
4	X4	B	8	infinity	8

Constraint Bounds Analysis

No.	Row name	St	Value	Max increase	Max decrease
1	Z	B			
2	R1	B			
	UPPER		5000	infinity	1000
3	R2	NU			
	UPPER		2000	1000	2000
4	R3	NU			
	UPPER		800	400	800
5	R4	NU			
	UPPER		1500	infinity	1500

Variable Bounds Analysis

No.	Column name	St	Value	Max increase	Max decrease
1	X1	B			
	LOWER		0	2000	infinity
2	X2	NL			
	LOWER		0	333.333	infinity
3	X3	B			
	LOWER		0	800	infinity

Abbildung 5: Sensibilitätsanalyse zu Aufgabe 3.42

7 Beispiel: Aufgabe 3.29b

In der Aufgabe wird einfaches LP-Problem vorgegeben, dass aber keine optimale Lösung hat und z.B durch LPSolve nicht gelöst werden kann. GLPK bietet ähnlich wie der XA-Solver eine Näherungslösung an.

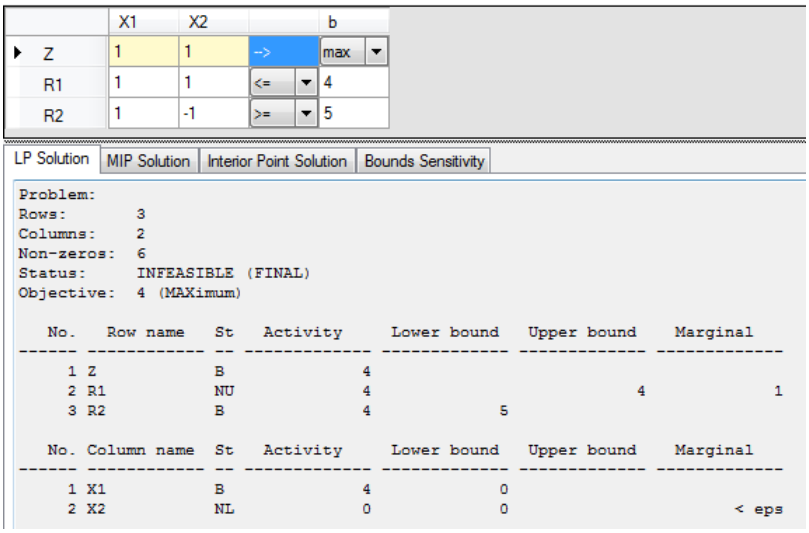


Abbildung 6: Näherungslösung für Aufgabe 3.29b

8 Bemerkungen

Die Anwendung wurde mit GLPK in Version *4.41* sowie für das .NET Framework *3.5* kompiliert. Die Distribution besteht aus folgenden zwei notwendigen Dateien:

- *GlpkSharp.dll* (die native Anbindung des GLPK an C#)
- *GlpkGui.exe* (die ausführbare Startdatei)

Zwar wurde das Programm aus Zeitgründen nicht sehr ausführlich getestet, jedoch sollten keine größeren Fehler beim Benutzen auftreten. Auch konnte die Präsentation der graphischen Lösung für LP-Probleme mit nur zwei Entscheidungsvariablen wegen Zeitmangel nicht mehr implementiert werden. Das C#-Framework *ZedGraph* wäre dafür aber sehr gut geeignet.

9 Internetverweise

GLPK	http://www.gnu.org/software/glpk/
GlpkSharp	http://yoyovicks.blog.free.fr/
ZedGraph	http://zedgraph.org/wiki/index.php?title=Main_Page