



**HOCHSCHULE
KONSTANZ**
TECHNIK, WIRTSCHAFT
UND GESTALTUNG

Anwendung der linearen Optimierung

Ausarbeitung

Optimierung der Ergebnisdarstellung vom ILOG-Solver im Power-LP

Verfasser:

Renate Bondar-Erni	286507
Vildan Özgür	286106

Betreuer:

Prof. Dr. Michael Grütz
Mamadou Kane

Wintersemester 2014/2015

Konstanz, 25.01.2015

Inhaltsverzeichnis

Inhaltsverzeichnis	2
Abbildungsverzeichnis	3
1. Einleitung	4
2. Projektbeschreibung	4
3. Ergebnisdarstellung der Solver MOPS, Weidenauer, LP-Solve und CPLEX	5
3.1 Ergebnisdarstellung: MOPS	5
3.2 Ergebnisdarstellung: Weidenauer	6
3.3 Ergebnisdarstellung: LP-Solve	6
3.4 Ergebnisdarstellung: CPLEX	6
4. Problemstellung	7
5. Vorgehensweise	8
5.1 CPLEX-Ausgabe ändern	9
5.2 Qualitätsmanagement	12
5.2.1 Beispiel 1	10
5.2.2 Beispiel 2	11
5.2.3 Beispiel 3	12
6. Einstellung: Versionsmanagement	12
7. Technische Umsetzung	13
8. Fazit	15

Abbildungsverzeichnis

Abbildung 1: Standardbeispiel	4
Abbildung 2: Ergebnisausgabe bei ILOG	4
Abbildung 3: Standardbeispiel für die verschiedenen Solver	5
Abbildung 4: Ergebnisausgabe bei MOPS	5
Abbildung 5: Ergebnisausgabe beim Weidenauer	6
Abbildung 6: Ergebnisausgabe beim LP-Solve	6
Abbildung 7: Ergebnisausgabe bei CPLEX	6
Abbildung 8: Solver-Code: Generiert Stappelerarbeitungsdatei für den ILOG-Solver	7
Abbildung 9: Ausführung "cplex.exe"	9
Abbildung 10: Optimierte Darstellungsansicht vom ILOG	9
Abbildung 11: Beispiel 1	10
Abbildung 12: Ergebnis LP-Solve	11
Abbildung 13: Ergebnis ILOG	11
Abbildung 14: Beispiel 2	11
Abbildung 15: ganzzahliges Ergebnis LP-Solve	11
Abbildung 16: ganzzahliges Ergebnis ILOG	11
Abbildung 17: Beispiel 2	12
Abbildung 18: unlösbares Ergebnis LP-Solve	12
Abbildung 19: unlösbares Ergebnis ILOG	12
Abbildung 20: Info Übersicht	13
Abbildung 21: Beschreibungstext für den Power-LP in OR-Alpha	14
Abbildung 22: Bild für die Beschreibung des Power-LPs in OR-Alpha	15

1. Einleitung

In der Veranstaltung *Anwendung der linearen Optimierung* bei Prof. Dr. Michael Grütz sollte innerhalb eines kleinen Projekts die Ergebnisdarstellung des ILOG-Solvers optimiert werden, da die bisherige Ausgabe unzureichend war. Dieses Projekt war für Herrn Prof. Grütz wichtig, sodass wir diese Aufgabe annahmen und es lösen wollten. Er fungierte für uns während des Semesters als Betreuer für dieses Projekt. Des Weiteren wurde Herr Mamadou Kane als technischer Betreuer hinzugezogen, da er für das Einbinden des Power-LPs auf der OR-Web-Seite, sowie für den OR-Alpha zuständig ist.

2. Projektbeschreibung

Die Ergebnisdarstellung vom ILOG im Power-LP ist sehr unübersichtlich und enthält viele unnötige Informationen. Wir hatten die Aufgabenstellung, die Ergebnisdarstellung vom ILOG im Power-LP zu optimieren. Wir hatten die Möglichkeit die Ergebnisse ähnlich wie in der Ergebnismaske vom MOPS, Weidenauer, LP-Solve oder der CPLEX-Ausgabe darzustellen. In Abbildung 1 ist das Standardbeispiel im Power-LP ersichtlich.

The screenshot shows the 'Solve' dialog box in the Power-LP application. The 'Solve' tab is active, showing a list of solvers: XA, MOPS, LPSolve, ILOG, and Weidenauer. The 'ILOG' solver is selected. The problem is set to 'Maximierung' (Maximization). The problem data is as follows:

	x1	x2	b
Zielfunktion	1	2	-> max 1
Restriktion 1	3	2	<= 12
Restriktion 2	1	3	<= 9

On the right, there is a table for integer constraints:

	Untere Grez	Obere Grez	Ganzzahl
x1	0	0	Nein
x2	0	0	Nein

Abb. 1: Standardbeispiel

```
Lösung Primal
XA | MOPS | Weidenauer | LPSolve | ILOG
<?xml version = "1.0" encoding="UTF-8" standalone="yes"?>
<CPLEXSolution version="1.2">
<header
  <problemName="ILOG.lp"
  objectiveValue="6.85714285714286"
  solutionTypeValue="1"
  solutionTypeString="basic"
  solutionStatusValue="1"
  solutionStatusString="optimal"
  solutionMethodString="dual"
  primalFeasible="1"
  dualFeasible="1"
  simplexIterations="2"
  writeLevel="1"/>
<quality
  epRHS="1e-06"
  epOpt="1e-06"
  maxPrimalInfeas="0"
  maxDualInfeas="0"
  maxPrimalResidual="8.88178419700125e-16"
  maxDualResidual="0"
  maxX="2.57142857142857"
  maxPi="0.571428571428571"
  maxSlack="0"
  maxRedCost="0"
  kappa="3.14285714285714"/>
<linearConstraints>
  <constraint name="c1" index="0" status="LL" slack="0" dual="0.142857142857143"/>
  <constraint name="c2" index="1" status="LL" slack="0" dual="0.571428571428571"/>
</linearConstraints>
<variables>
  <variable name="x1" index="0" status="BS" value="2.57142857142857" reducedCost="-0"/>
  <variable name="x2" index="1" status="BS" value="2.14285714285714" reducedCost="-0"/>
</variables>
```

Gibt die Zielfunktion aus.

Gibt die Variablen x1 und x2 aus.

Abb. 2:
Ergebnisausgabe bei ILOG

Anwendung der linearen Optimierung: Optimierung der Ergebnisdarstellung vom ILOG-Solver im Power-LP

In Abbildung 2 sieht man die Ergebnisdarstellung des ILOG-Solvers am Beispiel des Standardbeispiels. Um die Ergebnisse von der Zielfunktion und der Variablen zu bekommen, muss man sie erst suchen, da dort viele weitere zusätzliche Informationen stehen, die für unsere Bedürfnisse uninteressant sind. Ziel ist es nun, diese Darstellung so zu verändern, dass die benötigten Ergebnisse sofort ersichtlich sind.

3. Ergebnisdarstellung der Solver MOPS, Weidenauer, LP-Solve und CPLEX

In diesem Abschnitt zeigen wir jeweils die Darstellung der einzelnen Solver, die für unsere Aufgabe verlangt war. Die ILOG-Darstellung ist so zu optimieren, dass sie die Ergebnisse ähnlich wie in der 3.1 Ergebnisdarstellung für den MOPS, in der 3.2 Ergebnisdarstellung für den Weidenauer, in der 3.3 Ergebnisdarstellung für den LP-Solve oder für die 3.4 Ergebnisdarstellung des CPLEX darstellt. Für die Ausgabe nehmen wir unser Standardbeispiel.

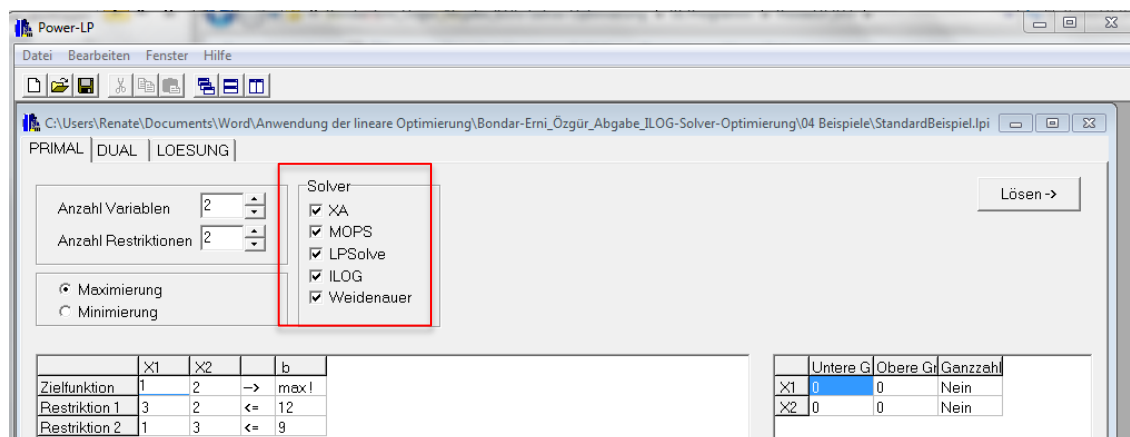


Abb. 3: Standardbeispiel für die verschiedenen Solver

3.1 Ergebnisdarstellung: MOPS

Lösung Primal

XA **MOPS** Weidenauer LPSolve ILOG

MOPS V7.06, (C) Uwe H. Suhl, 14.06.2004 30.10.2014 20.16

solution: (optimal)

iteration number = 2

Zielfunktion

...name... defined as
functional ...activity... 6.85714 ZF
restraints MYRHS

Variablen x1 und x2

SECTION 1 - ROWS

NUMBER	..ROW..	AT	..ACTIVITY...	SLACK	ACTIVITY	..LOWER LIMIT.	..UPPER LIMIT.	..DUAL ACTIVITY
3	R1	UL	12.00000	0.00000	0.00000	none	12.00000	-0.14286
4	R2	UL	9.00000	0.00000	0.00000	none	9.00000	-0.57143

SECTION 2 - COLUMNS

NUMBER	..COLUMNS	AT	..ACTIVITY...	..INPUT COST..	..LOWER LIMIT.	..UPPER LIMIT.	..REDUCED COST.
1	X1	BS	2.57143	1.00000	0.00000	none	0.00000
2	X2	BS	2.14286	2.00000	0.00000	none	0.00000

Abb. 4: Ergebnisausgabe bei MOPS

3.2 Ergebnisdarstellung: Weidenauer

Lösung Primal

XA | MOPS | **Weidenauer** | LPSolve | ILOG

OPTIMAL SOLUTION
 Weidenauer Optimizer Ver. 3.0 vom 29.1.2000 20:16:04

Section 1 - ROWS

...ROW...	AT	...ACTIVITY...	SLACK ACTIVITY	..LOWER LIMIT.	..UPPER LIMIT.	..DUAL ACTIVITY
ZF	\$\$	-6.85714	NONE	NONE		
R1	UL	12.00000	0.00000	NONE	12.00000	0.14286
R2	UL	9.00000	0.00000	NONE	9.00000	0.57143

Section 2 - COLUMNS

.COLUMNS	AT	...ACTIVITY...	..INPUT COST..	..LOWER LIMIT.	..UPPER LIMIT.	..REDUCED COST.
X1	BS	2.57143	1.00000		4.00000	
X2	BS	2.14286	2.00000		3.00000	

Abb. 5: Ergebnisausgabe beim Weidenauer

3.3 Ergebnisdarstellung: LP-Solve

Lösung Primal

XA | MOPS | Weidenauer | **LPSolve** | ILOG

Model name: lp

	x1	x2		
Maximize	1	2		
r_1	3	2	<=	12
r_2	1	3	<=	9
Type	Real	Real		
upbo	Infinite	Infinite		
lowbo	0	0		

Value of objective function: 6.85714

Actual values of the variables:

	x1	x2
	2.57143	2.14286

Actual values of the constraints:

	r_1	r_2
	12	9

Dual values with from - till limits:

	r_1	r_2	x1	x2
	0.142857	0.571429	0	0
			-1e+024	-1e+024

Abb. 6: Ergebnisausgabe beim LP-Solve

3.4 Ergebnisdarstellung: CPLEX

Iteration log . . .

Iteration:	1	Dual infeasibility =	0.000000
Iteration:	2	Dual objective =	6.857143

Dual simplex - Optimal: Objective = 6.8571428571e+000
 Solution time = 0.02 sec. Iterations = 2 (1)
 Deterministic time = 0.00 ticks (0.25 ticks/sec)

CPLEX> display solution variables -

Variable Name	Solution Value
x	2.571429
y	2.142857

CPLEX>

Abb. 7: Ergebnisausgabe bei CPLEX

4. Problemstellung

Da der gesamte Code des Power-LPs in der Programmiersprache C++ geschrieben worden war, mussten wir uns erst darin einarbeiten, da wir diese Sprache bisher noch nicht angewendet hatten. Innerhalb des Codes vom ILOG-Solver wird die Klasse „solver.cpp“ in der Methode *PutILOG_BAT(char* dir)* die „cplex.exe“ aufgerufen (siehe Abbildung 8). Sie ruft sie auf, aber die Berechnung wird nur im CPLEX durchgeführt und das Ergebnis aus der „cplex.exe“ wird dann nur in den Power-LP zurückgegeben und anschließend als Lösung ausgegeben. Das heißt, innerhalb des Codes vom Power-LP gab es zu diesem Zeitpunkt aus unserer Sicht keine Möglichkeit die Ausgabe zu beeinflussen.

```
/**
 * Generiert die Stapelverarbeitungsdatei fuer den ILOG Solver
 */
void __fastcall TSolver::PutILOG_BAT(char* dir) {

    // cplex.exe in Temp Ordner kopieren
    char sourcefile[1000];
    char destfile[1000];
    strncpy(sourcefile, reg_getillogdir(), sizeof(sourcefile));
    strcat(sourcefile, "cplex.exe");
    strncpy(destfile, reg_getworkdir(), sizeof(destfile));
    strcat(destfile, "cplex.exe");
    CopyFileA(sourcefile, destfile, 1);

    //Entfernen des letzten Backslash aus dem Work Directory für Aufbau der .bat Datei
    char tempdir[1000]="";
    strcpy(tempdir, reg_getworkdir());
    int index = strlen(tempdir);
    if (--index >= 0)
    {
        tempdir[index] = '\\0';
    }

    //Erstellen der der Stapelverarbeitungsdatei
    char filename[1000];
    char tempStr[1000];
    strncpy(filename, dir, sizeof(filename));
    strcat(filename, "ILOG.bat");
    int f = open(filename, O_RDWR | O_BINARY | O_CREAT | O_TRUNC, S_IREAD | S_IWRITE);
    writeLF(f, "@echo off");
    writeLF(f, "");
    writeLF(f, "cplex.exe -c \\read ILOG.lp \\ \"optimize\\\" \\\"write ILOG.out sol\\\" \\\"quit\\\" \");
    close(f);
}
```

Ab hier wird die „cplex.exe“ ausgeführt.



Abb. 8: Solver-Code: Generiert Stapelverarbeitungsdatei für den ILOG-Solver

Als nächstes schauten wir uns den CPLEX genauer an, um eventuell dort eine Möglichkeit zu finden, die Ausgabe anders zu definieren. Leider fanden wir außer der Anwendungsdatei nur dll-Dateien, die verschlüsselt sind, da sie in Maschinencode geschrieben wurden und wir keine Rechte von IBM besitzen diese zu ändern. Auch innerhalb der „cplex.exe“ ist es für uns nicht möglich, an der Ausgabe etwas zu ändern. Nachdem wir dies alles ausprobiert hatten, recherchierten wir im Internet nach weiteren Lösungsmöglichkeiten, fanden aber leider nichts Nützliches für unseren Fall. Danach hatten wir keine Ideen mehr, was wir noch ausprobieren könnten und haben uns deswegen an Herrn Prof. Grütz gewendet. Er hat uns den Tipp gegeben, dass wir einen Parser einbauen soll, der dann das Solution File mit den Ergebnissen einliest und

Anwendung der linearen Optimierung: Optimierung der Ergebnisdarstellung vom ILOG-Solver im Power-LP

wir darin definieren können, welche Variablen wir brauchen und uns nur die in ein neues File ausgeben lassen, welches dann im Power-LP eingelesen wird. Da wir aber nicht über genügend Programmiererfahrung in C++ verfügen, müssten wir uns zunächst einmal besser in diese Sprache einarbeiten, um dies mit dem Parser lösen zu können, was aber viel Zeit in Anspruch genommen hätte. Als Team haben wir uns daher nach einer alternativen Lösung umgeschaut. Wir haben den Kontakt zum vorherigen Team um Christian Damerau, Melanie Friedrich und Özlem Karahan aufgebaut, die im WS 13/14 den ILOG-Solver eingebunden hatten und gefragt, wie wir noch anders vorgehen könnten. Leider konnten sie uns diesbezüglich auch nicht helfen, da sie sich auch nicht gut genug mit C++ und dem IBM ILOG auskennen.

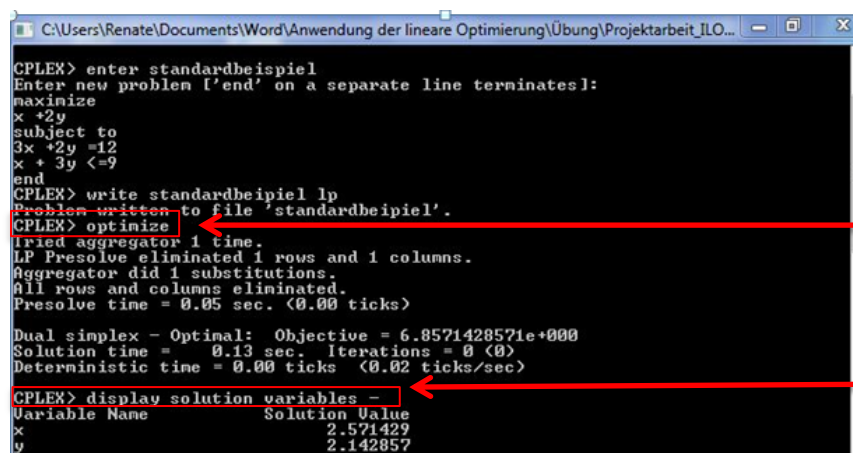
Frau Friedrich hat uns dann schließlich auf die Idee gebracht, Herrn Schlenker von der IBM zu kontaktieren, da er ihnen damals bei der ILOG-Anbindung geholfen hatte. Wir haben dann von Herrn Prof. Grütz die Mailadresse von Herrn Schlenker erhalten und sofort mit ihm Kontakt aufgenommen. Wir haben ihm vom unseren Problem erzählt und er gab uns den Vorschlag, dass wir die Solution Datei extrahieren und in XSLT umformatieren sollen. Er meinte, dass es nicht ganz trivial ist und Informatik-Kenntnisse erfordert. Zudem hat er uns für die XSLT-Prozessor einen Link geschickt, den wir kostenlos herunterladen und installieren könnten. Damit wir Zeit einsparen und effektiver sein konnten, haben wir uns die Aufgaben aufgeteilt: Vildan Özgür wollte den Ansatz mit dem Parsen voranbringen und Renate Bondar-Erni hat sich mit dem Vorschlag von Herrn Schlenker beschäftigt. Nebenbei hat uns auch Herr Kane immer wieder versucht zu helfen und hat uns Ideen bzw. Tipps vorgeschlagen, die ähnlich mit denen von Herrn Prof. Grütz waren. Dafür trafen wir uns ab zu mit ihm in seinem Büro.

5. Vorgehensweise

Wie in Abschnitt 4 beschrieben wurde, hatten wir uns die Aufgaben im Team aufgeteilt. Bevor wir mit den Ansätzen angefangen hatten, haben wir uns in der Klasse „solver.cpp“ die Methode *PutILOG_BAT(char* dir)* nochmals angeschaut (siehe Abbildung 8), da wir durch Herrn Schlenkers Vorschlag eine Idee hatten, die wir ausprobieren wollten. In der Methode, in der zweit letzten Codezeile (rot markiert) ist uns aufgefallen, dass die eingegebenen Befehle sich der “cplex.exe“-Anweisungen ähneln. Der “optimize“-Befehl gibt die Zielfunktion aus und die “display solution variables -“-Befehl gibt die Variablen bei der Ausführung von “cplex.exe“ aus. Somit kamen wir auf die Idee, die Codezeile

Anwendung der linearen Optimierung: Optimierung der Ergebnisdarstellung vom ILOG-Solver im Power-LP

entsprechend wie bei der "cplex.exe"-Ausführung zu verändern.



```
CPLEX> enter standardbeispiel
Enter new problem ['end' on a separate line terminates]:
maximize
x + 2y
subject to
3x + 2y = 12
x + 3y <= 9
end
CPLEX> write standardbeispiel lp
Problem written to file 'standardbeispiel'.
CPLEX> optimize
Tried aggregator 1 time.
LP Presolve eliminated 1 rows and 1 columns.
Aggregator did 1 substitutions.
All rows and columns eliminated.
Presolve time = 0.05 sec. (0.00 ticks)

Dual simplex - Optimal: Objective = 6.8571428571e+000
Solution time = 0.13 sec. Iterations = 0 (0)
Deterministic time = 0.00 ticks (0.02 ticks/sec)

CPLEX> display solution variables -
Variable Name      Solution Value
x                  2.571429
y                  2.142857
```

Befehl für Zielfunktionausgabe.

Befehl für Variablenausgabe.

Abb. 9: Ausführung „cplex.exe“

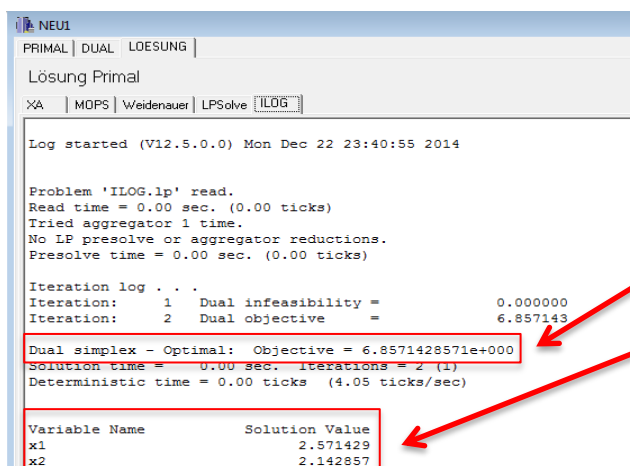
5.1 CPLEX-Ausgabe ändern

In der Klasse „solver.cpp“ der Methode *PutILOG_BAT(char* dir)* (siehe Abbildung 8) wurde folgendes verändert. Hierbei ist nur der relevante Codeteil aufgelistet:

```
[...]
//Erstellen der der Stappelverarbeitungsdatei
char filename[1000];
char tempStr[1000];
strncpy(filename, dir, sizeof(filename));
strcat(filename, "ILOG.bat");
int f = open(filename, O_RDWR | O_BINARY | O_CREAT | O_TRUNC, S_IREAD | S_IWRITE);
writeLF(f, "@echo off");
writeLF(f, "");
writeLF(f, "cplex.exe -c \"read ILOG.lp \" \"optimize\" \" \"display solution variables -\" \"write cplex.exe\" \"quit\" ");
close(f);
}
```

Als Beispiel haben wir unser Standardbeispiel verwendet (siehe Abbildung 3).

Durch diese Veränderung der Codezeile hat sich unsere Darstellungsansicht des ILOGs (siehe Abbildung 10) an die CPLEX Ergebnisdarstellung (siehe Abbildung 9) angepasst.



```
Log started (V12.5.0.0) Mon Dec 22 23:40:55 2014

Problem 'ILOG.lp' read.
Read time = 0.00 sec. (0.00 ticks)
Tried aggregator 1 time.
No LP presolve or aggregator reductions.
Presolve time = 0.00 sec. (0.00 ticks)

Iteration log . . .
Iteration: 1 Dual infeasibility = 0.000000
Iteration: 2 Dual objective = 6.857143

Dual simplex - Optimal: Objective = 6.8571428571e+000
Solution time = 0.00 sec. Iterations = 2 (1)
Deterministic time = 0.00 ticks (4.05 ticks/sec)

Variable Name      Solution Value
x1                  2.571429
x2                  2.142857
```

Zielfunktion

Variablen x1 und x2

Abb. 10:
Optimierte Darstellungsansicht vom ILOG

Anwendung der linearen Optimierung: Optimierung der Ergebnisdarstellung vom ILOG-Solver im Power-LP

So konnten wir dann recht einfach die Ergebnisdarstellung des ILOG-Solvers ändern, ohne viel Zeit zu verlieren. Diese Ergebnisdarstellung wurde dann per Mail als Screenshort an Herrn Schlenker und Herrn Prof. Grütz gesendet. Prof. Grütz war zufrieden mit der Lösung und für ihn reicht diese Ergebnisdarstellung aus, die jetzt natürlich deutlich übersichtlicher ist und die Ergebnisse auch besser nun zusammenfasst. Die anderen Ideen wurden somit natürlich nicht mehr umgesetzt.

5.2 Qualitätsmanagement

Die neue Ergebnisdarstellung wurde nun mit folgenden drei größeren Beispielen zusätzlich getestet, um die fehlerfreie Funktionalität der Darstellung zu gewährleisten. Da wir nur die Ergebnisdarstellung geändert haben und nichts an dessen Funktionalität, reichen aus unserer Sicht diese Beispiele als Qualitätsmerkmal aus. Dabei wird die ILOG-Ausgabe mit der LP-Solve-Ausgabe verglichen.

5.2.1 Beispiel 1

Als erstes Beispiel haben wir eine sehr umfangreiche Aufgabe gewählt mit 14 Restriktionen und 23 Variablen. Hierbei sind keine Grenzen und keine Ganzzahligkeit eingegeben worden. Es soll dargestellt werden, dass der ILOG auch bei mehreren Variablen das Ergebnis sinnvoll auflistet.

	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15	x16	x17	x18	x19	x20	x21	x22	x23	b
Zielfunktion	2	9	9	12	2	9	9	12	2	8	9	13	2	10	2	6	13	2	5	8	2	9	2	→ min!
Restriktion 1	-1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	= 7
Restriktion 2	0	0	0	0	-1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	= 0
Restriktion 3	0	0	0	0	0	0	0	0	-1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	= 0
Restriktion 4	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	= 0
Restriktion 5	0	0	0	0	0	1	0	0	0	0	0	0	0	0	-1	1	1	0	0	0	0	0	0	= 3
Restriktion 6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	1	0	0	0	= 0
Restriktion 7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	= 0
Restriktion 8	0	1	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	= 4
Restriktion 9	0	0	1	0	0	1	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	= 0
Restriktion 10	0	0	0	1	0	0	0	0	0	1	0	0	-1	0	0	0	0	0	0	0	0	0	0	= 0
Restriktion 11	0	0	0	0	0	0	1	0	0	0	1	0	0	-1	0	0	0	0	0	0	0	0	0	= 0
Restriktion 12	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	-1	0	0	0	0	0	= 4
Restriktion 13	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	-1	0	0	= 0
Restriktion 14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	-1	= 2

	Untere Grenze	Obere Grenze	Ganzzahl
x1	0	0	Nein
x2	0	0	Nein
x3	0	0	Nein
x4	0	0	Nein
x5	0	0	Nein
x6	0	0	Nein
x7	0	0	Nein
x8	0	0	Nein
x9	0	0	Nein
x10	0	0	Nein
x11	0	0	Nein
x12	0	0	Nein
x13	0	0	Nein
x14	0	0	Nein
x15	0	0	Nein
x16	0	0	Nein
x17	0	0	Nein
x18	0	0	Nein
x19	0	0	Nein
x20	0	0	Nein
x21	0	0	Nein
x22	0	0	Nein
x23	0	0	Nein

Abb. 11: Beispiel 1

Nachfolgend die identischen Ergebnisse vom LP-Solve und ILOG:

Anwendung der linearen Optimierung: Optimierung der Ergebnisdarstellung vom ILOG-Solver im Power-LP

Lösung Primal

XA | MOPS | Weidenauer | **LPSolve** | ILOG

Value of objective function: 137

Actual values of the variables:

x10	0
x11	0
x12	0
x13	0
x14	0
x15	0
x16	1
x17	2
x18	0
x19	0
x20	0
x21	0
x22	0
x23	0
x1	0
x2	7
x3	0
x4	0
x5	3
x6	0
x7	0
x8	3
x9	0

Abb. 12: Ergebnis LP-Solve

Lösung Primal

XA | MOPS | Weidenauer | LPSolve | **ILOG**

Tried aggregator 1 time.
LP Presolve eliminated 0 rows and 2 columns.
Aggregator did 9 substitutions.
Reduced LP has 5 rows, 12 columns, and 23 nonzeros.
Presolve time = 0.00 sec. (0.01 ticks)

Iteration log . . .
Iteration: 1 Dual objective = 60.000000

Dual simplex - Optimal: Objective = 1.3700000000e+002
Solution time = 0.00 sec. Iterations = 4 (0)
Deterministic time = 0.02 ticks (24.44 ticks/sec)

Variable Name	Solution Value
x2	7.000000
x5	3.000000
x8	3.000000
x16	1.000000
x17	2.000000

All other variables in the range 1-23 are 0.

Abb. 13: Ergebnis ILOG

5.2.2 Beispiel 2

In dem zweiten Beispiel haben wir nun eine Aufgabe gewählt, die nicht nur 3 Restriktionen mit größeren Zahlen beinhaltet, sondern die auch jeweils untere und obere Grenzen sowie die Ganzzahligkeit für die Variablen enthält.

Power-LP

Datei Bearbeiten Fenster Hilfe

C:\Users\Renate\Documents\Word\Anwendung der linearen Optimierung\Bonder-Erni_Özgür_Abgabe_ILOG-Solver-Optimierung\04 Beispiele\Beispiel 2.lpi

PRIMAL DUAL LOESUNG

Anzahl Variablen: 2
Anzahl Restriktionen: 3

Maximierung
Minimierung

Solver:
☒ XA
☒ MOPS
☒ LPSolve
☒ ILOG
☒ Weidenauer

Lösen ->

	x1	x2	b
Zielfunktion	30	70	-> max!
Restriktion 1	2	4	<= 200
Restriktion 2	2.8	2.8	<= 170
Restriktion 3	1.7	4.5	<= 150

	Untere G	Obere G	Ganzzahl
x1	30	60	Ja
x2	20	42	Ja

Abb. 14: Beispiel 2

Folgend nun die Ergebnisse vom LP-Solve und ILOG, die identisch sind:

Lösung Primal

XA | MOPS | Weidenauer | **LPSolve** | ILOG

Model name: lp

	x1	x2	
Maximize	30	70	
r_1	2	4	<= 200
r_2	2.8	2.8	<= 170
r_3	1.7	4.5	<= 150
Type	Int	Int	
upbo	60	42	
lowbo	30	20	

Value of objective function: 2450

Actual values of the variables:

x1	35
x2	20

Abb. 15: ganzzahliges Ergebnis LP-Solve

Lösung Primal

XA | MOPS | Weidenauer | LPSolve | **ILOG**

Sync time (average) = 0.00 sec.
Wait time (average) = 0.00 sec.
Total (root+branch&cut) = 0.00 sec. (0.01 ticks)

Solution pool: 2 solutions saved.

MIP - Integer optimal solution: Objective = 2.4500000000e+003
Solution time = 0.02 sec. Iterations = 0 Nodes = 0
Deterministic time = 0.03 ticks (1.58 ticks/sec)

Variable Name	Solution Value
x1	35.000000
x2	20.000000

Abb. 16: ganzzahliges Ergebnis ILOG

5.2.3 Beispiel 3

In Beispiel 3 wollen wir nun eine unlösbare Aufgabe lösen, die in dieser Konstellation nicht lösbar ist. Dafür wurden zusätzlich Grenzen und Ganzzahligkeit eingegeben, um es noch komplexer zu machen.

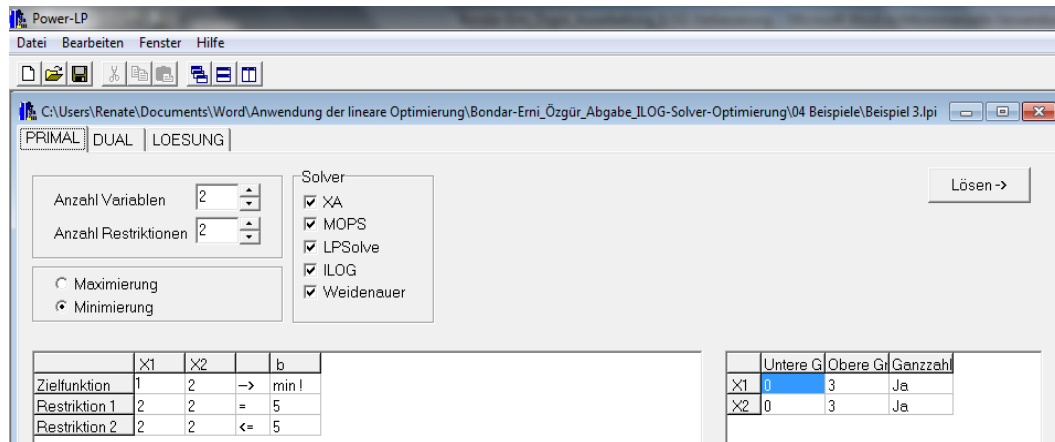


Abb. 17: Beispiel 3

Folgende identische Ergebnisse liefern LP-Solve und ILOG:

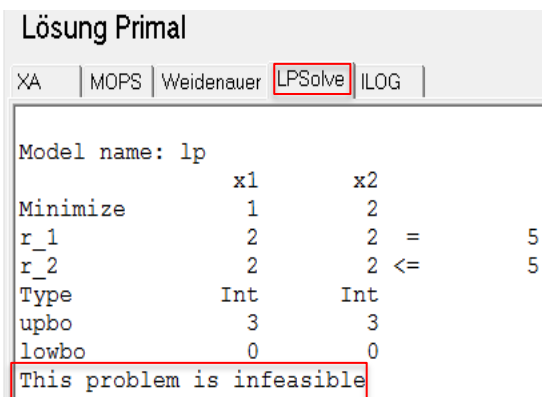


Abb. 18: unlösbares Ergebnis LP-Solve

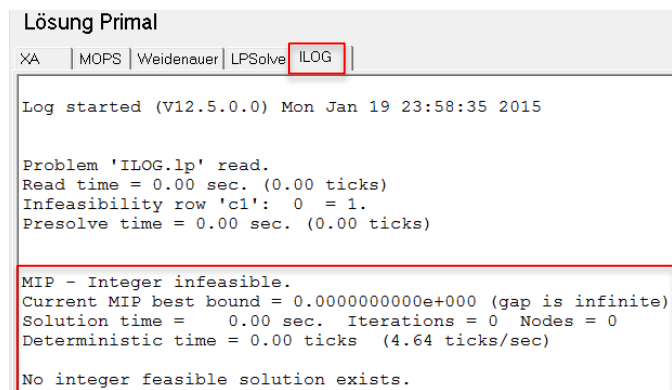


Abb. 19: unlösbares Ergebnis ILOG

Während der Bearbeitung des Projekts wurden immer wieder viele Beispiele mit der neuen Ausgabe im ILOG-Solver ausprobiert. Jedes Mal sah die Ergebnisdarstellung wie oben aus und hat sich nicht mehr verändert. Sie ist für jedes Beispiel gleich aufgebaut.

6. Einstellung: Versionsmanagement

In der Klasse „About.cpp“ wurden folgende Änderungen unter Design-Einstellungen für die neue Version 0.7.3 vorgenommen. Hierbei ist nur der relevante Codeteil aufgelistet:

[...]

object Version: TLabel

Left = 160

Top = 18

Width = 62

Height = 13

Caption = 'Version 0.7.3'

Anwendung der linearen Optimierung: Optimierung der Ergebnisdarstellung vom ILOG-Solver im Power-LP

```
IsControl = True
end
Lines.Strings = (
  'Copyright 2004 Andreas Burkhardt, Oliver Geiger, '
  'Maurice Lenz. '
  ''
  'AdBSF Fachhochschule Konstanz SS 2004 '
  'gemacht mit Borland C++ Builder 5 '
  ''
  'Version 0.6.1: Christian Gruhler mit Turbo C++ 2006 '
  ''
  'Version 0.7.0: Benedikt Woelfle mit Borland C++ Builder 5 '
  ''
  'Version 0.7.1: Isabel König und Timo Collmann '
  'mit Borland C++ Builder 6 '
  ''
  'Version 0.7.2: Melanie Friedrich, Christian Damerau '
  'und Özlem Karahan '
  'mit Borland C++ Builder 6 '
  ''
  'Version 0.7.3: Vildan Özgür, Renate Bondar-Erni '
  'mit Borland C++ Builder 6 '
  ')
ReadOnly = True
ScrollBars = ssVertical
TabOrder = 0
end
[...]
```

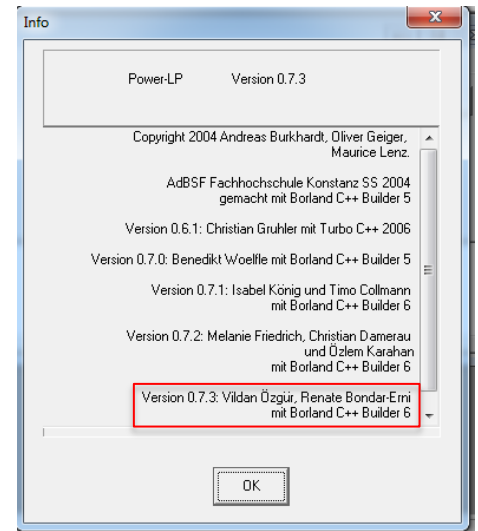


Abbildung 20: Info Übersicht

7. Technische Umsetzung

Zum Schluss wurde dann der Power-LP-Ordner gezippt und an Herrn Kane übergeben, der ihn dann in das OR-Web eingebettet hatte. Anschließend wurde dann der neu eingefügte Power-LP-Ordner aus dem OR-Web runtergeladen und getestet. Lief auf dem Hochschulsystem dann einwandfrei. Weitere Beschreibungs- bzw. Gestaltungsoptionen für OR-Web zum Power-LP wurden nicht vorgenommen, da das momentane Aussehen der Seite für uns in Ordnung ist. Nur leider war die Downloadzeit ziemlich lang und daher haben wir gemeinsam mit Herrn Kane besprochen, dass für den OR-Web eine verkürzte Version des Power-LPs genommen werden soll, damit man es schneller downloaden kann. Somit wurde innerhalb des Ordners die Dokumentation, der Source-Code und ein alter Power-LP-Ordner entfernt und damit war die Ordnergröße auf das halbe reduziert worden. Dies wurde dann Herr Kane übergeben und er wird es in nächster Zeit dann im OR-Web einbinden.

Momentan ist der vollständige Ordner von der neuen Version des Power-LPs im OR-Web drinnen, der nach dem Download aber einwandfrei funktioniert. Hier drinnen ist die Dokumentation und der Quellcode enthalten, d.h. es sind dort alle wichtigen

Anwendung der linearen Optimierung: Optimierung der Ergebnisdarstellung vom ILOG-Solver im Power-LP

Informationen abrufbar. Die Dokumentation zu dem Code wurde mit dem Tool DoxyS erstellt.

Der vollständige Power-LP-Ordner mit der neuen Version wird dann noch später in OR-Alpha und im C-Laufwerk eingebunden. Zusätzlich wurde dann von uns noch die Beschreibung für den Power-LP im OR-Alpha aktualisiert, da dort noch der Strada-Solver beschrieben wurde und der ILOG überhaupt nicht vorkam. Die textuelle Beschreibung wurde als HTML erfasst und abgespeichert. Dann wurde noch ein Bild vom neuen Power-LP erstellt, welches dann noch neben der Beschreibung in das OR-Alpha kommen soll. Es gibt eine Gruppe, die den OR-Alpha als Projekt in der Veranstaltung neu strukturiert. Sie haben bei sich lokal den OR-Alpha bereits fertiggestellt, der auch funktioniert. Des Weiteren binden sie alle Anwendungen ein, sowie die, die in dieser Veranstaltung bearbeitet wurden, bei sich lokal in OR-Alpha ein. Dementsprechend wurde auch unser Power-LP bereits dort eingebunden und funktioniert einwandfrei. Die Beschreibung, wie auch das Bild wurden schon eingesetzt. Dieser neu strukturierte OR-Alpha wird dann Herr Kane für das Sommersemester 2015 übergeben, der es dann auf dem Hochschulserver einbinden wird. Zu Testzwecken hat sich Herr Prof. Grütz von der Gruppe den lokalen OR-Alpha mit allen optimierten Anwendungen auf seinen Rechner installieren lassen.

Power-LP

Allg. Solver-Frontend

Power-LP ist ein Frontend für die Batch-Solver XA, MOPS, LPSolve, ILOG und Weidenauer. Allerdings ist der XA durch die Migration auf Windows 7 leider nicht immer voll funktionsfähig und gibt unvollständige Ergebnisse aus. Der ILOG-Solver ist seit WS 13/14 als Ersatz für den STRADA eingeführt worden.

Hinweis:

- Es lassen sich mehrere Solver per Checkbox auf einmal ausführen und die Ergebnisse im Programm per Reiterauswahl einsehen.
- Durch die Aufspaltung großer Zahlen auf Hilfsvariablen mit 0,1 Ganzzahligkeit können LPSolve, MOPS und XA-Solver echte Ganzzahligkeit berechnen.
- Damit der XA-Solver echte Ganzzahligkeit unterstützen kann, ist die Eingabe einer oberen Grenze nötig.
- Damit der Weidenauer Optimizer funktioniert, müssen entweder alle Variablen ganzzahlig sein oder keine. Mixed-Mode ist nicht erlaubt.
- ILOG unterstützt Ganzzahligkeit.
- Das Datenverzeichnis von LP-Interaktiv wird von Power-LP mitgenutzt.

© 2004 - 2005 Fachhochschule Konstanz

Abb. 21: Beschreibungstext für den Power-LP in OR-Alpha

Anwendung der linearen Optimierung: Optimierung der Ergebnisdarstellung vom ILOG-Solver im Power-LP

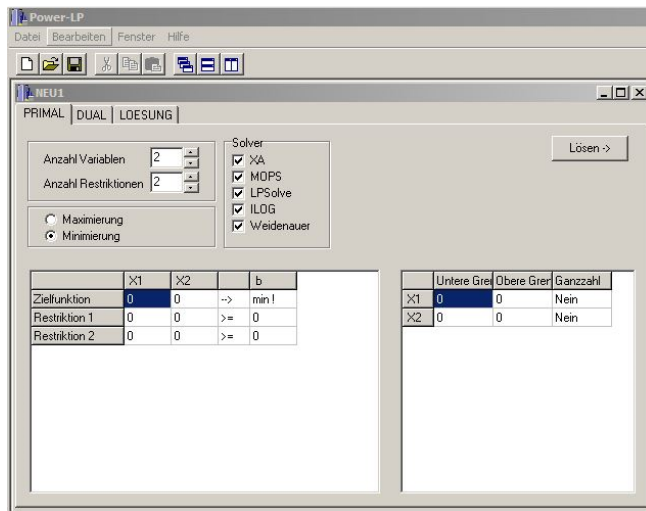


Abb. 22: Bild für die Beschreibung des Power-LPs in OR-Alpha

8. Fazit

Zusammenfassend können wir sagen, dass das Projekt sehr spannend und lehrreich war. Wir haben viel über die Funktionsweise des Power-LPs erfahren, also wie es aufgebaut ist und wie es funktioniert. Wir haben dadurch die Programmiersprache C++ intensiv kennen gelernt, welches für unsere weitere Zukunft sicherlich von Vorteil sein wird. Auch die Zusammenarbeit mit verschiedenen Leuten, wie Herrn Prof. Grütz, Herrn Kane, Herrn Schlenker oder das Altteam haben uns weiter voran gebracht, da man sich mit verschiedenen Persönlichkeiten auseinandersetzen musste und dies im Berufsleben nicht anders sein wird. Obwohl wir am Anfang einige Schwierigkeiten hatten, da wir nicht voran kamen oder den Code in der neuen Programmiersprache nicht genau verstanden, könnten wir dennoch durch die Vorschläge von Herrn Prof. Grütz, Herrn Kane und Herrn Schlenker einige Ideen sammeln, wie man solche Probleme in Zukunft lösen könnte und sind dadurch schlussendlich auf die Lösung unseres Problems gekommen. Am Ende war die Lösung eigentlich gar nicht komplex um zusetzen, wenn man weiß, was man tun muss. Aber aus solchen Fehlern lernt man und wächst daran weiter.

An dieser Stelle vielen Dank an alle beteiligten Personen, die uns geholfen haben, dieses Projekt erfolgreich zu beenden.