Ganzzahlige lineare Programmierung
Anwendungen und Lösungsansätze

Jens Pickenpack - Studiengang Master Informatik

3. Juli 2013

Seminar Algorithmik im SS 2013 - FH-Wedel

Inhaltsverzeichnis

1 Einleitung

1.1 Anwendungsfälle

Es gibt zahlreiche praktische und ökonomische Anwendungen für ganzzahlige lineare Optimierung (GLP). Dazu gehören die Bereiche:

- Produktionsplanung
- Tourenplanung
- Planung von Telekommunikationsnetzen
- Personalplanung

Typische Fragestellungen aus diesen Bereichen sind:

- Welches Produktionsprogramm liefert einen maximalen Deckungsbeitrag unter gegebenen Restriktionen?
- Welche Route ist optimal?
- Wie kann ein Kommunikationsnetz zu minimalen Kosten dimensioniert werden?
- Wieviele Personen müssen mindestens beschäftigt werden, um die Restriktionen bzgl. Arbeitszeiten erfüllen zu können.

Bei GLP handelt es sich um eines der Probleme, über die erst durch die Entwicklung von Großrechnern in größerem Maßstab nachgedacht wurde. Die Basisalgorithmen stammen daher auch aus den 1950er und 1960er Jahren.

1.2 steigende Praxisrelevanz

Lineare Programmierung ist von steigender Bedeutung, weil der globale Wettbewerbsdruck immer mehr zunimmt. Es wird immer wichtiger Ressourcen effizient zu nutzen, um Ressourcenverschwendung zu minimieren. Außerdem sind die Probleme die gelöst werden müssen oftmals mit Ganzzahligkeitsbedigungen versehen. Beispiele:

- Die Stückzahl von herzustellenden Produkten ist ganzzahlig
- Ein LKW/Bus fährt oder nicht
- Maschinenkapazitäten werden geschaffen oder nicht

Auch Probleme der theoretischen Informatik können mit GLP gelöst werden. Dazu gehören das Rucksackproblem, das Travelling Salesman Problem (TSP) und das Erfüllbarkeitsproblem.
Das Rucksackproblem beschreibt ein Problem der Kombinatorik, bei dem es gilt einen begrenzten Platz in einem Rucksack mit Gegenständen mit unterschiedlichem Platzbedarf und Gebrauchsnutzen maximal zu füllen.
Das TSP besteht darin, die kürzeste Route in einem Graph zu finden bei dem jeder Knoten nur genau ein mal besucht wird.
Das Erfüllbarkeitsproblem besteht darin herauszufinden, ob eine aussagenlogische Formel eine gültige Belegung hat. (Eine gültige Belegung macht den Ausdruck wahr).
Bekanntlich ist das TSP und das Erfüllbarkeitsproblem NP-Vollständig. NP-Vollständigkeit bedeutet, dass kein Algorithmus bekannt ist, der Probleme dieser Art in polynomieller Zeit lösen kann. Es wird allgemein angenommen, dass ein solcher Algorithmus nicht gefunden werden kann, also die Menge der in polynomieller Zeit lösbaren Probleme ungleich der Menge der NP-Vollständigen Probleme ist.
Diese Probleme haben also im allgemeinen eine Exponentielle Laufzeit. Man kann also nicht davon ausgehen große Probleme in vernünftiger Zeit optimal zu lösen. Daher werden oftmals Heuristiken eingesetzt, die eine gültige aber nicht unbedingt optimale Problemlösung liefern. Es ist möglich abzuschätzen wie weit die Lösung vom Optimum entfernt ist (die optimale Lösung der Zielfunktion des LP kann nie übertroffen werden).
In der Praxis werden lineare Programme mit mehreren 10000 Variablen und Restriktionsgleichungen problemlos optimal gelöst. Dies ist nicht für alle ganzzahligen Probleme möglich und ob es ohne Heuristiken möglich ist oder nicht kann nicht eindeutig abgeschätzt werden.
Zum Beispiel wird in einem Artikel [1] die Lösung eines kombinatorischen linearen Problems mit 1000 Variablen in zwei Sekunden beschrieben, was die Autoren überraschte.

2 Algorithmen

2.1 Begriffe und Grundlagen

Nun einige einführende Definitionen und Beispiele

Definition 1. Die Ganzzahligkeitsbedingungen eines GLP wegzulassen und es als LP zu lösen nennt man Relaxierung oder Relaxation.

Definition 2. Gemischt ganzzahlige lineare Programme enthalten auch Variablen ohne Ganzzahligkeitsbedingung

[1] [2]IPCO

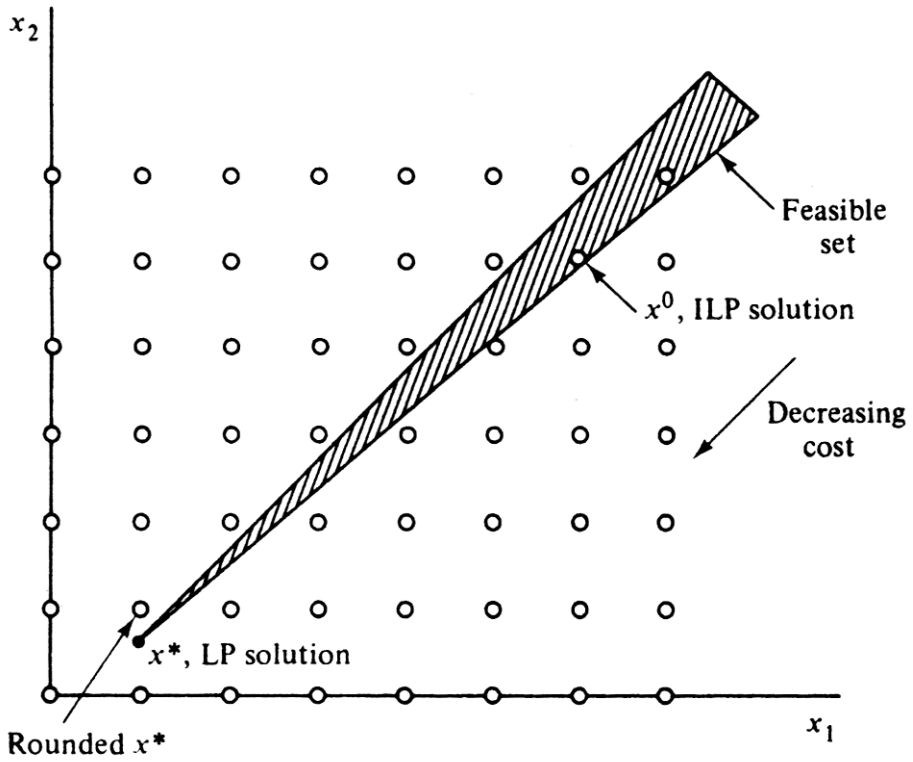
Definition 3. Binäre lineare Programme (BLP) enthalten nur Variablen mit den Werten Null oder Eins

Definition 4. Ein GLP ist ein Problem folgender Art: $\max; min \left\{c^T x | Ax \leq; \geqslant b, x \geq 0, x \in \mathbb{Z}^{\mathbb{N}}\right\}$

Beispiel. $\max \{x_0 = 2x_1 + 5x_2\}$
$2x_1 - x_2 \leq 9$
$2x_1 + 8x_2 \leq 31$
$x_1 x_2 \in \mathbb{N}$

2.2 Betrachtungen zur Komplexität von GLP

Man könnte zunächst auf die Idee kommen, dass der Forderung nach Ganzzahligkeit durch Runden entsprochen werden könnte. Dies ist nicht der Fall. Die diskrete Optimallösung kann beliebig weit von der linearen entfernt sein. Das Problem kann auch keine diskrete Lösung haben. Folgende Abbildung zeigt anschaulich im zweidimensionalen, warum dies so ist:



Der Punkt X* ist LP optimal. Ihn zu runden ergibt einen Wert der nicht zur Lösungsmenge des GLP gehört. Es ist auch nicht der Fall, dass alle Lösungswerte des GLP im Fall einer Minimierung größer/gleich der Lösungswerte der LP sein müssen.

Die Mächtigkeit der GLP kann am Beispiel des NP-Vollständigen Erfüllbarkeitsproblems, relativ leicht deutlich gemacht werden. Bei diesem Problem geht es um die Entscheidung, ob eine aussagenlogische Formel erfüllbar ist, d.h. ob es eine Variablenbelegung gibt, die sie wahr macht.

Dieses Problem kann nach folgendem Vorgehen nachgebildet werden:

1. Eine Formel wird in Konjunktive Normalform (KNF) gebracht und ist somit eine Konjunktion von Disjunktionen der Form: $\bigwedge_j \bigvee_i (\neg) X_{ji}$
   Beispielformel:$(A \vee B) \wedge (\neg B \vee C \vee D)$

2. Jede Disjunktion wird zu einer Restriktion mit Beschränkung >=1. Jede darin Vorkommende Variable wird an ihre Stelle im Tableau übertragen und wenn sie negiert ist von 1 subtrahiert. Beispiel:
   Klausel Zwei führt zu einer Restriktion:
   0*X1+(1-X2)+X3+X4 >=1

3. Die Zielfunktion besteht aus der Menge aller Variablen die minimiert oder maximiert werden kann und einen Zielwert >= der Anzahl der Klauseln haben muss.

Fakt. *Jede Formel kann in KNF gebracht werden und hat genau eine Darstellung dieser Art.*

Also kann jede Aussagenlogische Formel mit diesem Verfahren auf Gültigkeit geprüft werden. Somit gehört GLP ebenfalls zur Komplexitätsklasse der NP-Vollständigen Problemen.

2.3 Exakte Verfahren

Es gibt zwei Grundideen zur exakten Lösung von GLP. Das Branch and Bound Verfahren und Schnittebenenverfahren. Diese stammen aus der Anfangszeit der linearen Optimierung und benötigen ein Verfahren zur Lösung von linearen Problemen welches in der Regel häufig angewendet werden muss um ein GLP zu lösen. Auch wenn der Simplexalgorithmus keine bewiesene polynomielle Laufzeit besitzt (wie andere Algorithmen), so ist er doch im Einsatz, weil er gut warmstartet - Er findet beim Hinzufügen von weiteren Bedingungen zu einem Problem, schnell eine neue Optimallösung.

2.3.1 Branch and Bound Verfahren

Bei dem Branch and Bound Verfahren handelt es sich um ein in rekursives Verfahren, bei dem ein Entscheidungsbaum aufgebaut wird, der es ermöglicht Teilbäume nicht zu betrachten, weil dort keine besseren Lösungen vorkommen können als eine die schon gefunden wurde . Es stammt aus den 1950/1960er Jahren.

Der Algorithmus läuft folgendermaßen:

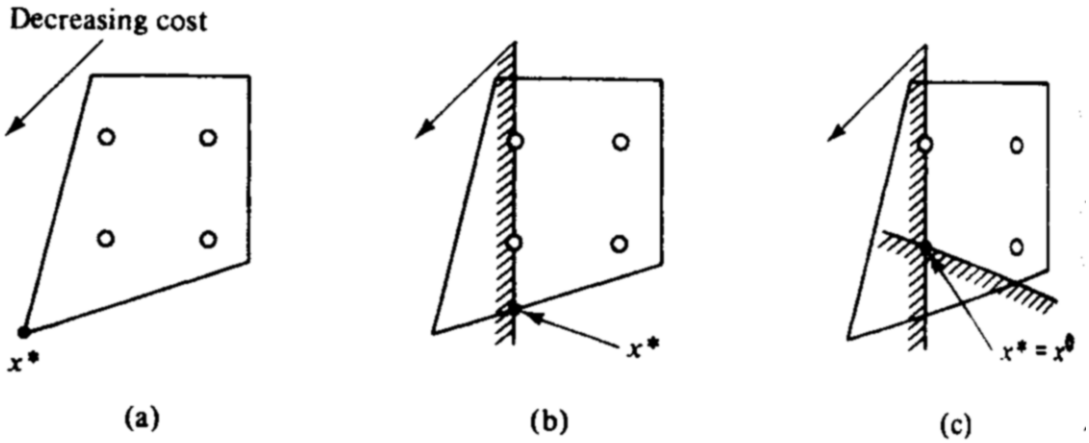
- Lösen der Relaxation. Wenn die Variablen ganzzahlig sind, sind wir fertig.
  - Zerlegen des Problems in Teilmengen (Branches), die optimale Lösungen enthalten können oder nicht
  - Berechnung von oberen/unteren Schranken (Bound)
  - eine Auswahlregel zur Wahl der nächsten betrachteten Branches (LIFO oder best bound)
  - Lösung mit dem Branch and Bound Verfahren

Ein Illustriertes Beispiel findet sich im Abschnitt 3.

2.3.2 Schnittebenenverfahren

Es handelt sich um Verfahren, bei dem zusätzliche Restriktionen eingeführt werden, die den Lösungsbereich so lange einschränken bis alle Variablen ganzzahlig sind oder keine weiteren Schnitte möglich sind und ein Branch and Bound Prozess gestartet wird. Die Idee dazu stammt ebenfalls aus den 1950/1960er Jahren.
Der Algorithmus läuft folgendermaßen:

- Lösen der Relaxation. Wenn die Variablen ganzzahlig sind, sind wir fertig.
  - solange nicht alle Variablen ganzzahlig:
    * Einführung von zusätzlichen Bedingung(en) (Cut), die keine potentiell zulässigen Lösungen eliminieren, aber zumindest das vorherrige nicht ganzzahlige Optimum eleminieren.
    * Erneute Lösung



In Schritt a wurde das Problem Beispielsweise per Simplexverfahren gelöst und x* ermittelt. Da die Variablen nicht alle ganzzahlig waren wurde ein Cut eingeführt und erneut per Simplexverfahren gelöst und ein neues x* gefunden (Schritt b). Es wird ein weiterer Schnitt eingeführt und das Problem ein weiteres Mal gelöst (c). Nun sind alle Variablen ganzzahlig und somit die optimale Lösung gefunden.
Die Frage wie ein Schnitt (oder ob sogar mehrere) vor erneuter Lösung mittels Simplex eingeführt werden ist vom konkreten Verfahren abhängig.
Ein gängiges Verfahren ist der Gomory Cut. Es war das erste Verfahren für das gezeigt werden konnte, dass es in endlich vielen Schritten eine Lösung für ganzzahlige lineare Probleme liefert. Es galt lange als unpraktikabel, da es zwar endlich viele Schritte benötigt, aber doch viele.

Ein Gomory Cut erfolgt nach folgender Idee:
Füge eine Schnittgerade hinzu, die durch einen außen liegenden ganzzahligen Punkt geht, die keine weiteren ganzzahlig erlaubten Punkte eleminiert. Damit wird das Optimum des alten Problems im neuen Problem nicht mehr gelten.
Mathematisch funktioniert dies so:

- Idee: Aufteilung einer Zeile des Tableau in einen ganzzahligen und einen gebrochenen Teil.
- Beispiel für die Trennung:
  - 2,7 -> 2 und 0,7
  - -3,7 -> -4 und 0,3
- für den gebrochenen Teil gilt: $\sum f_{ij} * x_j \geqslant f_{i0}$
- Es wird zum Tableau folgende Zeile hinzugefügt:$-\sum f_{ij} * x_j + s = -f_{i0}$

i steht für die Zeile, f für den gebrochenen Teil der Faktoren von den Variablen x

Ausschnitt aus einem Tableau:

	K1	K2	S8	RS
K1	1	1/5	-1/20	3,5

1*K1 + 1/5*K2 - 1/20*S8 = 3,5

zusätzliche Schnittbedingung also:
-1/5*K2 - 19/20*S8 + C1 = -0,5
Es ergibt sich ein neues Tableau mit einer Zeile und einer Spalte mehr

	K1	K2	S8	C1	RS
K1	1	1/5	-1/20	0	3,5
C1	0	-1/5	-19/20	1	-0.5

In der neuen Zeile ist C1 als Schlupfvariable eingeführt worden, darum ist ihr Wert 1. Dies Problem wird erneut per Simplex gelöst.

2.3.3 Branch and Cut Verfahren

Bei dem Branch and Cut verfahren handelt es sich um eine Kombination aus Branch and Bound und Schnittebenen Verfahren. Es stammt aus den 1980-1990er Jahren. Vor dem Branch werden Cut Vorgänge durchgeführt, was sich als sehr effektiv herausgestellt hat. Es ist in unterschiedlichen Varianten State-of-the-Art für die Lösung von GLP. Es werden mehrere unterschiedliche Schnittebenenverfahren verwendet, um den Lösungsbereich möglichst gut vorher einzuschränken.

2.4 Heuristische Verfahren

Heuristische Verfahren werden in diesem Seminar nur kurz erwähnt, aber nicht weiter behandelt. Sie verzichten auf sichere Optimalität der Lösung, eine Abschätzung der Lösungsqualität aber möglich. Es handelt sich um problemspezifische Ansätze, daher sind die Algorithmen nicht generisch. Sie sind die einzige Möglichkeit, mit Sicherheit bei komplexen GLP-Problemen schnelle Lösungen zu erhalten. Daher sind sie von großer praktischer Relevanz.

2.5 Bewertung und Einsatz der Algorithmen

2.5.1 Bewertung

Diese Tabelle ist eine eher subjektive Bewertung und die Einstufung ist insbesondere für Heuristiken eher schwammig.

Algorithmus	Prozessor	Speicher	Qualität	Zeit
Branch and Bound	o	-	++	-
Schnittebenen	-	o	++	-
Branch and Cut	o	o	++	o
Heuristiken	(+)	(+)	(o)	++

Branch and Bound Verfahren sind Speicherintensiver, weil viele Teilprobleme im Speicher bleiben müssen. Eine Parallelisierung der Berechnung der Lösung der unabhängiger Teilbäume ist möglich.
Schnittebenenverfahren sind weniger speicherintensiv, aber sie lassen sich allgemein schlechter parallelisieren - Gomory Cuts garnicht. In der Regel funktionieren Schnittebenenverfahren deutlich weniger gut als Branch and Bound. Zudem sind neigen sie zu numerischer Instabilität.
Branch and Cut Verfahren lassen sich ebenfalls parallelisieren.
Die Verbesserungen der Algorithmen ändern nichts an der NP-Vollständigkeit des Problems.

2.5.2 Einsatz

Varianten von Branch and Cut sind State-of-the-Art. Sie kommen auch in freien Solvern wie GLPK zum Einsatz und verwenden unterschiedliche Schnittebenenverfahren in Verbindung mit Branch and Bound sowie dem Simplexverfahren.
GLPK ist einfach konfigurierbar was die eingesetzten Cut und Branchverfahren betrifft. Es ist frei verfügbar und wird zum Beispiel von Sage (einer mathematischen Bibliothek/Präcompiler für Python) standardmäßig verwendet. Benchmarks zeigen jedoch, dass für große Probleme sind kommerzielle Lösungen wie CPLEX empfehlenswert sind.

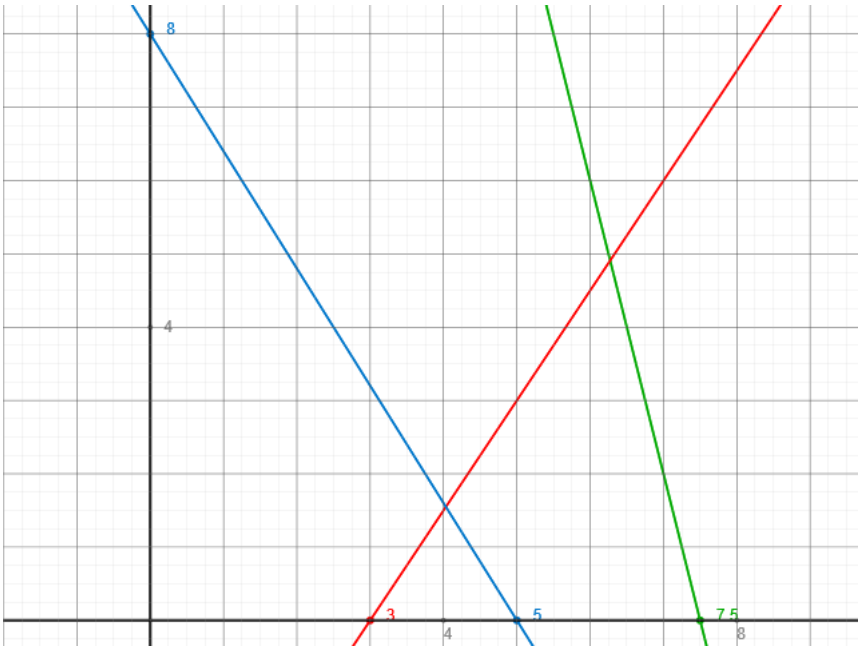
3 Beispiele

3.1 Vorstellung

Problemdefinition:

- zu maximierende Funktion: c = x1 + 4*x2
  - 5x1+8x2<=40

- -2x1+3x2<=9
- x1,x2>=0 und ganzzahlig



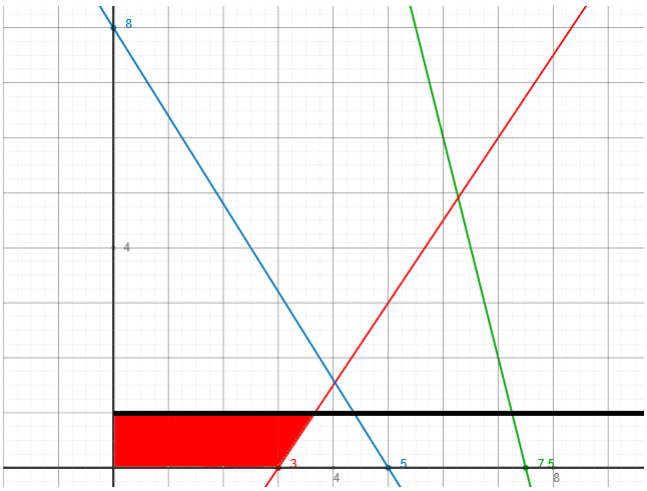
Blaue Linie erste Restriktion: f(x) = (40-8x2)/5
Rote Linie zweite Restriktion: f(x) = (9-3x2)/-2
Grüne Linie Zielfunktion für c = 30: f(x)= 30-4x2

3.2 Lösung per Branch and Bound nach Dakin

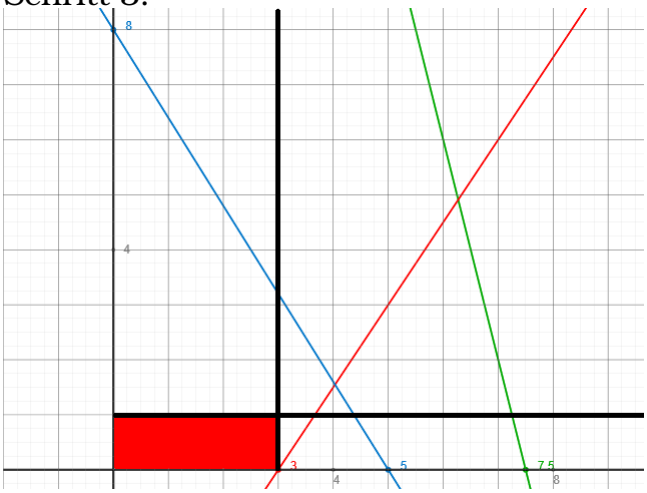
Ergebnis der Lösung per Simplex ergibt in jedem Schritt :

Schritt	x1	x2	c*	Restriktionen
1	48/31	125/31	548/31	
2	1	11/30-4*x2	47/3	x1<=1
3	1	3	13	x1<=1 ; x2<=3
4	NZL			x1<=1 ; x2>=4
5	2	15/4	17	x1>=2
6	16/5	3	76/5	x1>=2 ; x2<=3
7	3	3	15	x1>=2 ; x2<=3 ; x1<=3
8	4	5/2	14	x1>=2 ; x2<=3 ; x1>=4
9	NZL			x1>=2 ; x2>=4

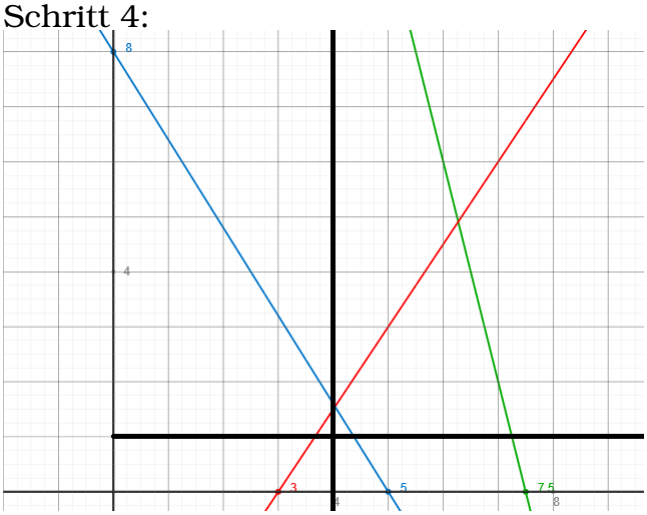
In Schritt 8 wird über x2 nicht weiter verzweigt, da die Zielfunktion in Branches nie größer als 14 werden kann und 15 schon besser war.
Zwischendurch wurde eine mögliche Lösung x1=1 , x2=3 bei einem Zielfunktionswert von 13 gefunden.
Das Maximum ist also für x1=3 , x2=3 erreicht und hat einen Wert von 15.
Schitt2:



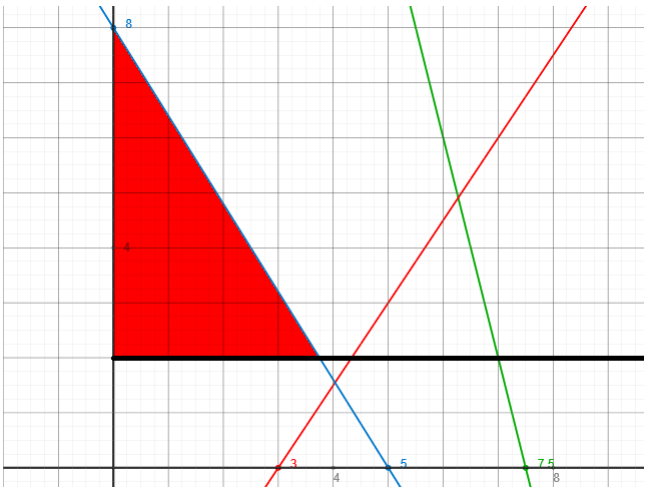
Schritt 3:



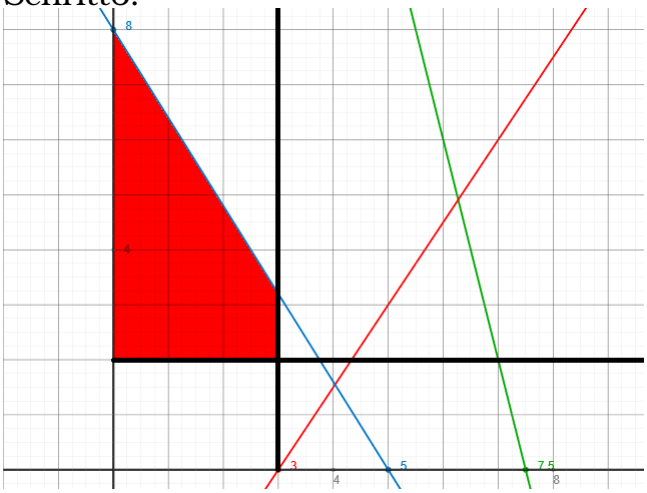
Schritt 4:



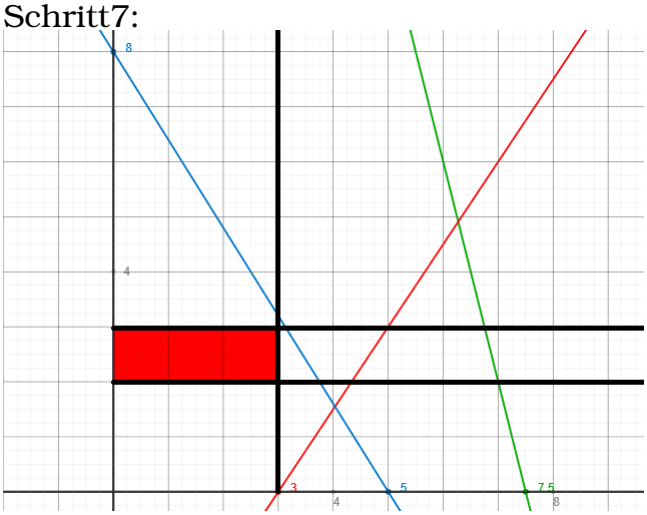
Schritt 5:



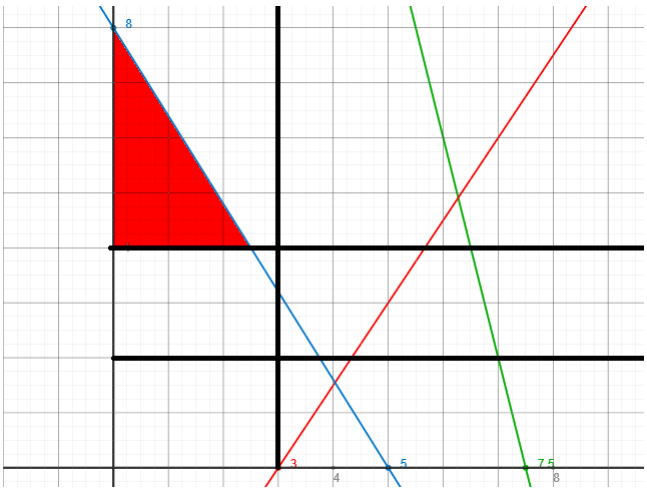
Schritt6:



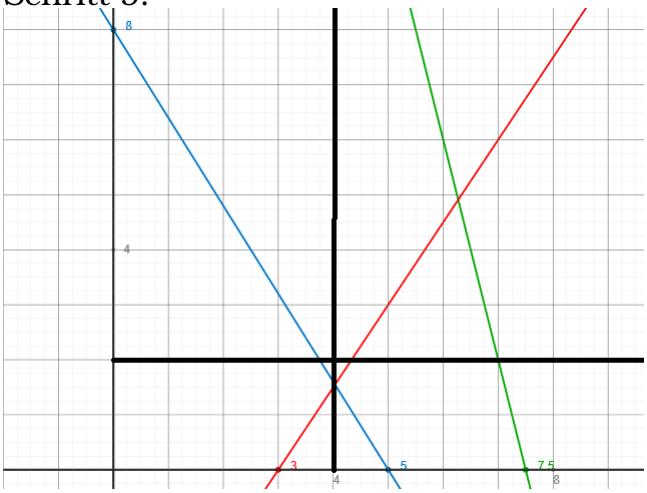
Schritt7:



Schritt 8:



Schritt 9:



3.3 Aussagenlogik - Erfüllbarkeit

Wir wollen eine aussagenlogische Formel auf Erfüllbarkeit prüfen.

- a v b ^ ~a v b

Problemmodellierung

- Maximiere a + b
  - a + b >= 1
  - (1-a) + b >= 1 <=> -a + b >= 0
  - a,b binär

Ergebnis : 2.0 a: 1.0 b: 1.0
Die Formel ist erfüllbar, da das Ergebnis der Anzahl der Klauseln entspricht.

4 Schlussbemerkungen

Die Lösung von diskreten linearen Problemen ist deutlich schwieriger als die Lösung von linearen Problemen - nämlich NP-Schwer. Dennoch wird von Informatikern/Praktikern erwartet, Probleme dieser Art zu lösen. Es ist wichtig diese Probleme zu erkennen und nicht zu versprechen, dass man sie optimal löst. Schnell eine gute, korrekte (nicht unbedingt optimale) Lösung zu finden, ist eine interessante Herausforderung.

Literatur

[1] C. Papadimitriou K.Steiglitz, Combinatorial Optimization, Dover Publications, 1982

[2] P.Murzel R.Weiskircher, Optimizing over all combinatorial embeddings of a planar Graph aus Integer Programming and Combinatorial Optimization, Springer, 1999

[3] P. Brucker, Ganzzahlige lineare Programmierung mit ökonomischen Anwendungen, Verlag Anton Hain, 1975

[4] Sandor Nevelö, Ganzzahlige lineare Programmierung mit Hilfe von Branch & Bound, Grin Verlag, 2002