Seminararbeit über Abstraktion in Texas Hold'em Poker

Marian Wieczorek

Abstract

Diese Seminararbeit ist eine Ausarbeitung der Paper [Billings07] und [Gilpin06]. Die Paper werden getrennt behandelt und am Schluss einander gegenüber gestellt. Da beide Paper lineares Programmieren nutzen um eine möglichst optimale Strategie zu berechnen, widmet sich das erste Kapitel grob der Formulierung von Texas Hold'em Heads Up als lineares Programmierproblem. Im zweiten Kapitel wird die Problemstellung des ersten Papers erklärt. Darauf hin folgt in Kapitel drei die Bewertung der angedachten Abstraktionen. Die Bewertungen ergeben sich aus Experimenten und Berechnungen der Autoren. Im letzten Kapitel zum ersten Paper werden die Vereinfachungen in drei Spielprogrammen zusammengeführt. Das erste Kapitel zum zweiten Paper arbeitet dessen Problemstellung heraus. Im sechsten Kapitel wird die Suche einer optimalen Strategie für die ersten zwei Runden und im siebten Kapitel für die letzten zwei Runden erleutert.

Das erste Paper stellt unterschiedliche Stufen der Abstraktion für die Zweispielervariante von Texas Hold'em Poker, Heads up genannt, vor. Dabei wird der ursprüngliche Entscheidungsbaum um hundert Billionen Blätter reduziert. Auf der vereinfachten Variante, die offline berechnet wird, wird mit linearer Programmierung eine optimale Strategie berechnet. Einige Abstraktionen arbeiten die Autoren als wertvoll heraus und erstellen auf ihnen drei Spielprogramme (Bots) *psOpti0*, *psOpti1* und *psOpti2*.

Das zweite Paper nimmt sich zur Aufgabe Heads up Hold'em mit minimiertem „Expertenwissen“ zu spielen. Die Autoren entwickeln einen Bot *GS1*, der zunächst eine große Menge an Informationen vorberechnet und eine nahezu optimale Strategie bis zum Flop entwickelt. Für die Runden Turn und River wird die Strategie hauptsächlich durch Echtzeit (online) lineares Programmieren gefunden.

1. Poker als lineares Programmierproblem

Texas Hold'em Poker hat eine einfache Struktur. Ein Spiel besteht aus mehreren Runden. Es gibt Signalrunden, in denen die Spieler Karten erhalten, oder eine oder mehrere Board-Karten aufgedeckt werden und Wettrunden in denen die Teilnehmer gegeneinander spielen. In den Wettrunden wählen die Spieler verschiedene Strategien. Sie können in geregelten Reihenfolgen checken (check), wetten (bet), austeigen (fold), mitgehen (call) und erhöhen (raise). In Limit Poker sind die Anzahl der Wetten begrenzt. Durch Aufzählung aller Strategiekombinationen lassen sich theoretisch für jede Runde die Erwartungswerte aller Spieler berechnen.

Bei bekannten Erwartungswerten kann eine Spielmatrix $M \in \mathbb{R}^{m \times n}$ aufstellt werden, die zu jeder Strategie der Spieler den Gewinn beziehungsweise Verlust (negativer Gewinn) angibt. m ist die Anzahl der Strategien von Spieler A und n die Anzahl der Strategien von Spieler B. Wählt Spieler A eindeutig eine Zeile der Spielmatrix (also eine Strategie), nennt man dies „pure strategy“. Der Gegner B versucht dann durch Wahl einer geeigneten Spalte (seine Gegenstrategie) mit seinem Zug den Gewinn für Spieler A zu minimieren (m_{ij} ist der Gewinn von Spieler A und Verlust von Spieler B). Dies bedeutet bei einem Zweispieler Zero-Sum Spiel wie Poker, dass sich der Gewinn für Spieler B erhöht. Entscheidet Spieler

A auf Grund einer Verteilung, welche Strategie er wählt, nennt man dies „mixed strategie“. Dieses Vorgehen hat den Vorteil, dass Spieler A nun seinen erwarteten Gewinn durch die Wahl einer geeigneten Verteilung statistisch vergrößern kann. Spieler A wählt immer eine Zeile und Spieler B eine Spalte. Haben beide Spieler ihre Strategie gewählt, lässt sich der Gewinn/Verlust in der Matrix ablesen.

Von Neumann bewies, dass optimales Spielen dem Spieler mindestens den Spieletheoretischen Wert des Spiels ermöglicht. Optimales Spielen mit einer „mixed strategie“ stellt also die Aufgabe, die beste Verteilung über den Strategien zu ermitteln. Dabei versucht man den minimalen Gewinn zu maximieren, was zu einem linearen Programmierproblem führt [Wayne07]. Grundlegend ist jedoch, dass die Spielmatrix bekannt ist. Bei vollständigem Heads up Hold'em ist der Entscheidungsbaum jedoch viel zu groß.

2. Problemstellung

[Billings07] sucht eine optimale Strategie für Texas Hold'em Heads up. Da der Entscheidungsbaum zu groß ist (ca. 10^{18} Blätter), werden verschiedene Abstraktionen untersucht. Ziel der Abstraktion ist es den Suchbaum so zu verkleinern, so dass auf diesem mit linearer Programmierung und erschöpfender Berechnung der abstrahierten Erwartungswerte eine optimale Strategie berechnet werden kann. Die Autoren erkennen, dass die Strategien auf dem ursprünglichen Problem keine Optimalität gewährleisten und nennen sie deshalb „pseudo-optimal“. Die Qualität der Abstraktion ist entscheidend. Aus diesem Grund werden verschiedene Abstraktionen analysiert. Es stellt sich heraus, dass Abstraktionen oft nicht effektiv genug sind, um den Suchbaum entscheidend zu verkleinern. Durch Andere verliert das Spiel an taktischen Eigenschaften, die den resultierenden Computer-Spieler schwach machen.

3. Abstraktionen

In diesem Abschnitt werden die verschiedenen Abstraktionen vorgestellt. Sowohl für Hände als auch für die Wettrunden gibt es gewinnbringende Vereinfachungen.

3.1 Abstraktion über das Blatt

Die Autoren stellen kurz übliche Reduktionen vor. Naheliegend ist die Abstraktion über Farben und Rang der Hände. Die Farbe spielt beim Pokern für die Wertung keine Rolle. Bei Rangabstraktionen fasst man Hände zusammen, die gleiche Wertigkeit aber verschiede Kombinationen aufweisen. Diese beiden Reduktionen bieten nicht viel. Durch Reduktion der Farben verkleinert man das Problem um den Faktor 24!. Gleichwertige Kombinationen treten im Poker zu selten auf, als dass sich die Abstraktion lohnt.

Eine starke Verkleinerung des Suchbaums würde sich ergeben, wenn man das Deck von 52 Karten verringern würde. Experimente haben ergeben, dass optimale Strategien auf dem reduzierten Suchbaum im tatsächlichen Spiel zu ungenau sind.

3.2 Abstraktion der Wettrunden

Vielversprechend ist die Abstraktion der Wettrunden. Man kann sich zu nutze machen, dass selbst bei unlimit Poker selten mehr als vier Runden gewettet wird. Eine vollständige Aufzählung der möglichen Strategien bei vier Runden ergibt 19 verschiedene Möglichkeiten. Fold und check gefolgt von einem Fold treten in der Praxis nicht auf. Endet eine Wettrunde mit einem Fold, ist das Spiel vorbei. Die Autoren schlagen vor die übrigen neun Kombinationen auf drei Runden zu verringern. Dieses Vorgehen birgt noch keine erkennbaren Verluste. In Experimenten wurde jedoch nachgewiesen, dass die Reduktion um weitere Runden auf einmal einen deutlichen Einbruch an Genauigkeit aufweist. Schon mit der nur drei statt vier Runden kann der Suchbaum um ca. 10^{11} $(= 2 * \binom{48}{3} * 9 * 45 * 9 * 44 * 19 + 2 * 45 * 9 * 44 * 19 + 2 * 44 * 19 + 4)$ auf ca. 10^7 Blätter reduziert werden.

3.3 Einschränken der Runden

Aus der Reduktion von Runden werden verschiedene Modelle abgeleitet. Die Autoren stellen die Möglichkeiten vor den River oder Preflop zu ignorieren. Daraus entstehen entsprechend Pre- und Postflop-Modelle. In Preflop Modellen verliert man nur strategische Raffinessen. Um nach drei Runden nicht auf den Show down zu setzen, können die Erwartungswerte über das Turn und River vorberechnet und für das lineare Programmieren genutzt werden. Für vereinfachte Pokervarianten wurden die Runden sogar bis auf eine reduziert. Hierbei verliert man sämtliche Taktiken für folgende Runden. Die Autoren benutzen ein solches Modell von Alex Selby um Strategien in einem Postflop-Modell exakter zu berechnen. Werden keine weiteren Vereinfachungen getroffen, muss das Preflop-Modell auf eine Runde beschränkt werden, damit das Problem lösbar wird. Eine gute Vereinfachung bietet Bucketing (siehe dazu Abschnitt 3.4), wodurch auch Dreirundenmodelle möglich sind.

Der Vorteil an postflop Modellen ist, dass sie den hohen Branching-Faktor des Preflops vernachlässigen. Mit anschließender Wettrunde erhält man eine Reduktion um $\binom{52}{2}\binom{50}{2} * 9 \approx 10^7$ Blätter. Der Spielverlauf ist jedoch eigentlich für Entscheidungen in späteren Runden entscheident. Bei Postflop- versucht man wie bei Preflop-Modellen nicht Erwartungswerte aber Eingangswahrscheinlichkeiten zu ermitteln. Die Autoren konstruieren für die Bots *psOpti0*, *psOpti1* und *psOpti2* verschiedene Verfahren, um die Eingangswahrscheinlichkeiten zu berechnen und messen deren Brauchbarkeit.

Anstatt Runden zu beschränken wurde angedacht, Runden zusammenzufassen. Die Modelle ergaben grobe Ungenauigkeiten, weswegen sie verworfen wurden.

3.4 Abstraktion der Strategien

Poker ist ein Spiel mit lückenhafter Information. Manche Kombinationen von Karten sind stärker, andere stellen sich im Laufe des Spiels als stark heraus. Auch Bluffen ist auf dieser Grundlage möglich. Die möglichen Varianten einer Hole-Boardcard-Kombination sind in jeder Runde bestimmbar. Das Zusammenfassen strategisch gleicher Kombinationen nennt man Bucketing. Die Autoren erkennen, dass sich die Stärke einer Kombination durch mehr als eine Dimension auszeichnet. Sie stellen die Projektion auf zwei Dimensionen vor. Erste ist der erwartete, mögliche Gewinn durch die Kombination. Als zweite Dimension wählen sie das Potential der Kombination. Dies beschreibt die Wahrscheinlichkeit, dass sich aus

der momentanen Situation eine Verbesserung ergibt. Jede zu einer Runde möglichen Kombination lässt sich als Punkt in die Ebene projezieren. Clustering-Methoden ergeben die Einteilung in die so genannten „Buckets“. Die Autoren beobachten, dass die Buckets ungleich gefüllt sind. Buckets mit hohem Erwartungswert die Hand zu gewinnen, sind weniger gefüllt als jene mit niedrigem Erwartungswert. Karten mit niedrigen Erfolgsaussichten scheinen also auch strategisch ähnlich und somit wertloser zu sein. Sichert man einem Bucket Kombinationen, die schwach im Erwartungswert sind, aber hohe Verbesserungsaussichten haben zu, kann man beobachten, dass dieser Bucket am häufigsten in Bluff-Situationen eingesetzt wird. Die Autoren wählen lediglich sechs Buckets, von denen einer für das Bluffen bereitgestellt wird. Dies ist eine sehr grobe aber trotz dessen erfolgreiche Abstraktion. Dass es keinen Unterschied macht in unterschiedlichen Runden eine unterschiedliche Anzahl von Buckets zu benutzen, wurde in Experimenten festgestellt.

Hat man für jede Runde die Buckets berechnet, wird auch jedem Gegner ein Bucket zugeteilt. Da die Karten nun abstrahiert sind, werden Übergänge zwischen den Runden nur noch als Transitionen zwischen den $(\underbrace{n \times n \times \ldots \times n}_{\text{Anzahl der Spieler}})$ Buckets betrachtet. Dazu müssen Übergangswahrscheinlichkeiten ermittelt werden. In jeder Runde ist ein Blatt mit einer gewissen Häufigkeit in allen Bucket-Tupel dieser Runde vertreten. Durch austeilen einer weiteren Karte, wechselt dies mit den angegebenen Übergangswahrscheinlichkeiten in die Bucket-Tupel der nächsten Runde. Bucketing reduziert die Komplexität von Heads up Texas Hold'em stark. Das gesamte Spiel hat nun nur noch $(6^2)^4 * 9^3 * 19 \approx 10^{10}$ und ein dreistufiges Preflop-Modell 10^7 Blätter.

4. Kombination von pre- und postflop Modellen

Generell wird nach einer optimalen Auswahl der Strategien gesucht. Diese gewinnt man durch Lösen des Minimax-Problems mit Hilfe von linearem Programmieren auf dem abstrahierten Suchbaum. Als Abstraktionen stehen pre- und postflop Modelle zur Verfügung. Es liegt nahe für jede Runde einer Hand unabhängig eine Strategie zu berechnen. Der Spielverlauf ist aber maßgeblich für alle Entscheidungen, die während eines Spiels getroffen werden. In geringem Maße ist Dekomposition jedoch möglich.

Strategie für *psOpti0* Für die Strategie von *psOpti0* in den Wettrunden wird das lineare Programmierproblem auf einem dreistufigen postflop Modell berechnet. Dieses nutzt die Abstraktionen der Reduktion in den Wettrunden und Bucketing. Da das Preflop für die Optimierung der Strategie nicht benutzt wird, müssen die Eingangswahrscheinlichkeiten abgeschätzt werden. Hier wird für *psOpti0* Gleichverteilung der Strategien im Preflop angenommen. Spielt *psOpti0* Poker, wird es in der ersten Runde die Aktionen des Gegners stets mit call abnicken. Im weiteren Verlauf nutzt es die optimierte Strategie.

Strategie für *psOpti1* *psOpti1* geht bei der Berechnung der Strategie für das Postflop-Modell ebenfalls von einer Gleichverteilung aus. Es berechnet vier Modelle für Potgrößen von zwei, vier, sechs und acht bets. Im Gegensatz zu *psOpti0* spielt *psOpti1* das Preflop mit dem Einrundenmodell von Alex Selby (siehe Abschnitt 3.3) und wechselt danach zum zur Potgröße passenden postflop Modell. Dieses Aneinandersetzen ist

wie bereits bemerkt spieltheoretisch gesehen falsch. Es wird jedoch das beste Ergebnis erziehlen.

Strategie für *psOpti2* Für *psOpti2* werden sieben postflop Modelle berechnet. Ihre Eingangswahrscheinlichkeiten ergeben sich aus einem von Profis „handgefertigten“ Preflop-Modell. Im Gegensatz zum Einrundenmodell wird diesmal ein Dreistufiges benutzt. Der enorme Branching-Faktor der Preflop-Phase wird durch Bucketing reduziert. Die Autoren erhoffen sich dadurch eine bessere Approximation der Anfangsbedingungen.

Alle Programme benutzen sechs Buckets pro Runde.

5. Problemstellung

[Gilpin06] behandelt das Spielen von Poker mit nur minimalem Vorwissen über das Spiel. Um nach wie vor in annehmbarer Zeit Strategien zu berechnen, lagern die Autoren aufwendige Berechnungen einmal in Datenbanken aus. Der Artikel geht nicht ausführlich auf die Abstraktionen und ihre Wirkung ein, benutzt sie aber implizit. Der Einfluss der meisten, benutzten Vereinfachungen wurde bereits in den vorherigen Kapiteln erleutert. Ist der Entscheidungsbaum auf handhabbare Größe reduziert, wird wiederum lineares Programmieren für Suche nach einer optimalen Strategie verwendet. Zur Reduktion des Entscheidungsbaums wird der Algorithmus *GameShrink* benutzt. Durch Angabe von einem Abstraktionsgrad und Zusatzinformationen über Beziehungen der Knoten, liefert *GameShrink* ein vereinfachtes Spiel zurück. Da die Autoren nicht auf Expertenwissen setzen, verbessert sich der Algorithmus automatisch mit dem Stand der Technik. Tiefergehende Abstraktionen können berechnet werden, während bei anderen Spielprogrammen durch neue Erkenntnisse in der Forschung auf Poker-Wissen basierende Algorithmen komplett verworfen werden müssen. Im Gegensatz zu *psOpti* berechnet *GS1* Strategien sowohl off- als auch online. Dies bringt neue Herausforderungen mit sich.

6. Der iterative Aufbau von *GS1*

Dieser Abschnitt beschäfftigt sich mit den einzelnen Schritten in denen *GS1* berechnet wird. Wie bei den vorherigen Bots werden die Runden getrennt behandelt. Das Postflop-Modell wird jedoch nicht offline auf der Basis von Preflop-Wissen generiert. Vielmehr wartet *PS1* ab, wie sich das Spiel bis zum Turn entwickelt und berechnet daraufhin von seinem Standpunkt die Strategie für den weiteren Verlauf.

6.1 Strategie für preflop und flop

Um eine Abstraktion auf den ersten zwei Runden mit *GameShrink* durchzuführen, müssen zunächst einige Vorberechnungen stattfinden. Ohne die Auslagerung von Ergebnissen wäre die Rechenzeit nicht tragbar. *GameShrink* ist in der Lage einen Spielbaum nur durch Angabe eines Grenzwerts auf strategische Gleichheit zu vereinfachen. Doch dauert dies häufig sehr lange, wenn das Programm selbst entscheiden muss, wie ähnlich sich Instanzen sind. Damit die *GameShrink* auf Hold'em anwendbar ist wird ein Maß für Ähnlichkeit definiert und eine Datenbank angelegt mit der *GameShrink* effizient einzelne Hände vergleichen kann.

Zu jeder zwei-Karten Kombination lassen sich die mittlere Anzahl an Erfolgen und Misserfolgen bestimmen. Zwei Hände werden ähnlich genannt, wenn der Abstand der Erfolge zusammen mit dem Abstand der Misserfolge niedrig ist. Diese Relation ähnelt dem Bucketing des ersten Papers. Damit *GameShrink* die Relationen schnell vergleichen kann, werden die Werte für den mittleren Erfolg und Misserfolg für alle preflop-flop-Kombination in eine Datei ausgelagert. Diese enthält $\binom{52}{2}\binom{50}{3} = 25\ 989\ 600$ Einträge. Die Berechnung dieser Datei benötigt dabei eine weitere Datenbank mit $\binom{52}{7} = 133\ 784\ 560$ Werten, die lediglich den Rang aller Hände speichert.

Von dort an, da *GameShrink* auf den Datenbanken in nur vier Stunden neue Abstraktionen berechnen kann, haben die Autoren einige reduzierte Spielbäume evaluiert. Für das weitere Vorgehen wurde eine Vereinfachung gewählt, die 169 Klassen für das preflop und 2 465 Klassen für das flop benutzt. Die 169 Klassen für das preflop entsprechen dabei einer verlustfreien Reduktion um Farbäquivalenz der hole-Karten Kombinationen.

Da die Erwartungswerte für die lineare Programmierung notwenig sind, werden schlussendlich auch diese in einer Datenbank vorberechnet zu Verfügung gestellt. Ohne die Auslagerung wäre eine ausreichende Evaluierung unterschiedlicher Abstraktionen nicht möglich. Die Datenbank besteht aus $\frac{\binom{52}{2}\binom{50}{2}}{2}\binom{48}{3} = 14\ 047\ 378\ 800$ Einträgen, sie wird stets Stückweise geladen und konnte auf die Hälfte ihrer eigentlichen Größe reduziert werden, da der Gewinn des Gegners gleich dem eigenen Verlust ist. Erwartungswerte zweier Hände können also durch Anpassen der Vorzeichen einfach den Spielern zugewiesen werden. Mit dieser Datenbank ist es möglich mit linearem Programmieren eine optimale Strategie in rund einer Woche zu finden.

7. Strategie für turn und river

Die Suche nach einer optimalen Strategie besteht wie in den ersten zwei Runden aus einer Vereinfachung mit *GameShrink* und anschließendem linearem Programmieren. Allerdings wird Letzteres diesmal zur Spielzeit durchgeführt. Bei der Abstraktion wird an einer Stelle zum ersten Mal spezielles Wissen eingesetzt. Die Autoren erkennen, dass die Strategie von späteren Runden nie auf vorangegangene Wetten aufbaut. Es ist also nur der Spielverlauf der Karten entscheidend. Außerdem nutzen die Autoren die Farbäquivalenz der Karten aus. Eine Reduktion, die im vorherigen Paper nicht nötig war, bringt nun bei Echtzeitberechnung wertvolle, weitere Sekunden. Auf allen möglichen $\binom{52}{4}$ Flop-Resultaten, mit dem hinzugekommenen Turn, abzüglich der Farbäquivalenz werden nun Abstraktionen berechnet. Da der Grenzwert für jede Folge ein Anderer sein kann, definieren die Autoren einen gewünschten Abstraktionsgrad. Dieser wird dann mit binärer Suche über den Grenzwerts mit *GameShrink* möglichst genau angenähert. Die Abstraktionen von turn und river werden getrennt berechnet.

Damit die Strategieberechnung representativ ist, werden wieder die Eingangswahrscheinlichkeiten abgeschätzt. In diesem Paper verwenden die Autoren die Regeln von Bayes. Auf der Basis des bekannten, approximierten Verlaufs werdem die Wahrscheinlichkeiten zu jeder hole-Karten Kombination des Gegners berechnet. Es gilt: $Pr[\theta|h,s] = \frac{Pr[h|\theta,s]Pr[\theta]}{\sum_{\theta' \in \Theta} Pr[h|\theta',s]}$, wobei θ das betrachtete Hole des Gegners, Θ die Menge aller hole-Karten Paare, h die History und s die bisherige Strategie des Gegners ist. $Pr[h|\theta,s]$ lässt sich einfach aus dem beobach-

teten Spielverlauf berechnen. In jeder Runde müssen alle Wahrscheinlichkeiten berechnet werden. Daraufhin kann die Echtzeitstrategiesuche erfolgen. Ist *GS1* an der Reihe ohne ein entgültiges Ergebnis erziehlt zu haben, handelt er nach dem Derzeitigen und rechnet von diesem Punkt weiter. Findet *GS1* das Ergebnis früher, fängt er bereits mit Berechnungen für die nächste Runde an. Dabei ist zu bemerken, dass zu jeder Zeit lineares Programmieren eine Lösung bereitstellt. Diese muss aber während der Rechenzeit nicht die optimale sein.

8. Vergleich der Verfahren

Die vorgestellten Verfahren sind schnell im Spielen. Selbst die Variante, die eine optimale Strategie für das turn und river in Echtzeit sucht, braucht im Mittel nur neun Sekunden. Jedoch ist das berechnen der Datenbanken und abschätzen der Erwartungswerte sehr aufwendig. Diese betrugen Wochen und Monate auf zum Teil mehr als einem Computer.

Das zweite Verfahren stellt einen Computer-Spieler vor, der einige state-of-the-art Programme schlägt. Dies zeigt, dass Poker ohne Expertenwissen erfolgreich gespielt werden kann. Die *psOpti* Programme untersuchen verschiedene Approximation von Postflop Modellen. Dabei stellt sich raus, dass *psOpti1* überraschender Weise überragt. *psOpti2*, dessen Eingangswahrscheinlichkeiten auf einem, von einem Experten konstruiertem, Preflop Modell basiert, enthält nach Einschätzung der Entwickler taktische Fehler. *psOpti1* erziehlt selbst gegen Weltklasse menschliche Spieler gute Ergebnisse. Gegen „thecount“ scheint *psOpti1* zunächst zu Gewinnen. Die entscheidende Wende ergibt sich, als die Autorem dem Poker-Meister offenbaren, dass *psOpti1* nicht lernt. Dadurch war es ihm möglich schwächer zu spielen und dabei die Schwächen des Programms zu erforschen. Nach kurzer Zeit unterlag *psOpti1*.

Keines der vorgestellten Programme lernt. Dies scheint zunächst von der psychologischen Seite von Poker ein großer Nachteil zu sein, wird aber durch die Versuche widerlegt. Beide Verfahren untersuchen Heads Up Hold'em. Die Abstraktion für Mehrspieler Partien ist um einiges schwieriger.

Literatur

Billings, D., Burch, N., Davidson, A., Holte, R., Schaeffer, J., Schauenberg, T., & Szafron, D. (2007). Approximating game-theoretic optimal strategies for full-scale poker. *International Joint Conference on Artificial Intelligence (IJCAI)*, 661–668.

Gilpin, A., & Sandholm, T. (2006). A competitive texas hold'em poker player via automated abstraction and real-time equilibrium computation..

Wayne, K. (2007). Lineare programming lecture iii..