Data-Driven Decision-Making In Enterprise Applications

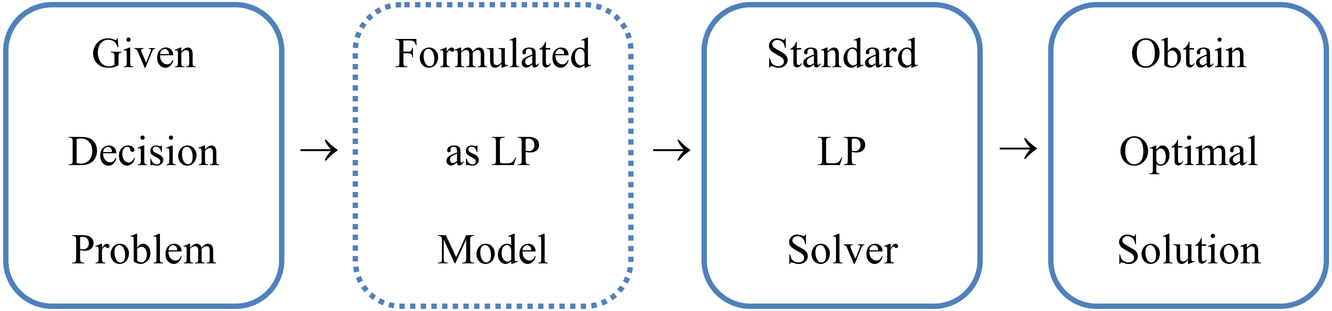
Linear Programming

Rainer Schlosser, Martin Boissier, Matthias Uflacker

Hasso Plattner Institute (EPIC)

April 25, 2019

Decision-Making Using Linear Programming



Tough Magic Easy Awesome!

Linear Programming

- **What is a Linear Program?**
- Theoretical Foundations
- Standard Solution Algorithms
- **Tricks to formulate decision problems as LP**
- **Examples, Examples, Examples, . . .**

What is a Linear Program?

Decision variables $x_1, x_2, \ldots$	The controls to be determined.
Constraints via $\leq, =, \geq$	Expressed linearly in $x_1, x_2, \ldots$.
One objective: max or min	Expressed linearly in $x_1, x_2, \ldots$.

- Does the solver help you to define the right LP?
- How to formulate a given problem as an LP?
- What does "expressed linearly" means?

Which Terms are Linear in the Variables $x_1, x_2, \ldots$?

$3x_1 - 2x_2$, x_1^{-3}, $a \cdot \ln(x_1)$, $\ln(a) \cdot x_1$

$a \cdot x_1$, $a \cdot b \cdot x_1$, $a^2 \cdot x_1$, $a \cdot x_1^2$

$|x_1|$, $\max(x_1, 5)$, x_1^2 / x_1 , $(x_1 - 3) \cdot (x_2 + 3)$

$x_1 \cdot x_2$, x_1 / x_2 , $1_{\{x_1 = 5\}} :=$ if $x_1 = 5$ then 1 else 0

Example of a Linear Program

Objective: $\max_{x_1, x_2 \in \mathbb{R}} 2 \cdot x_1 + 3 \cdot x_2$

Constraints: $0.5 \cdot x_1 + x_2 \leq 4$,

$0 \leq x_1 \leq 3$, $x_2 \geq 0$

- Only use linear expressions
- Only use $\leq$, $=$, $\geq$ in the constraints ($<$, $\neq$, $>$ are not allowed!)
- Is such an LP understandable for all solvers?

Example of a Linear Program in Standard Form

Objective: $$\max_{x_1, x_2 \geq 0} 2 \cdot x_1 + 3 \cdot x_2$$

subject to $$1 \cdot x_1 + 2 \cdot x_2 \leq 8$$

$$1 \cdot x_1 + 0 \cdot x_2 \leq 3$$

Standard form:

- Linear combinations of (*non-negative*) variables
- Use $\leq$ (or $=$) in all constraints
- No variables on the "*right-hand side*" of the constraints

Optimal Solutions

Objective: $$\max_{x_1, x_2 \geq 0} 2 \cdot x_1 + 3 \cdot x_2$$

s.t. $$1 \cdot x_1 + 2 \cdot x_2 \leq 8$$

$$1 \cdot x_1 + 0 \cdot x_2 \leq 3$$

$$\max_{x_1, x_2 \geq 0} \vec{c}\,'\vec{x} \quad \text{s.t.} \quad A \cdot \vec{x} \leq \vec{b}$$

with $\vec{c} := (2,3)$, $\vec{b} := (8,3)$,

$$A := \begin{pmatrix} 1 & 2 \\ 1 & 0 \end{pmatrix}$$

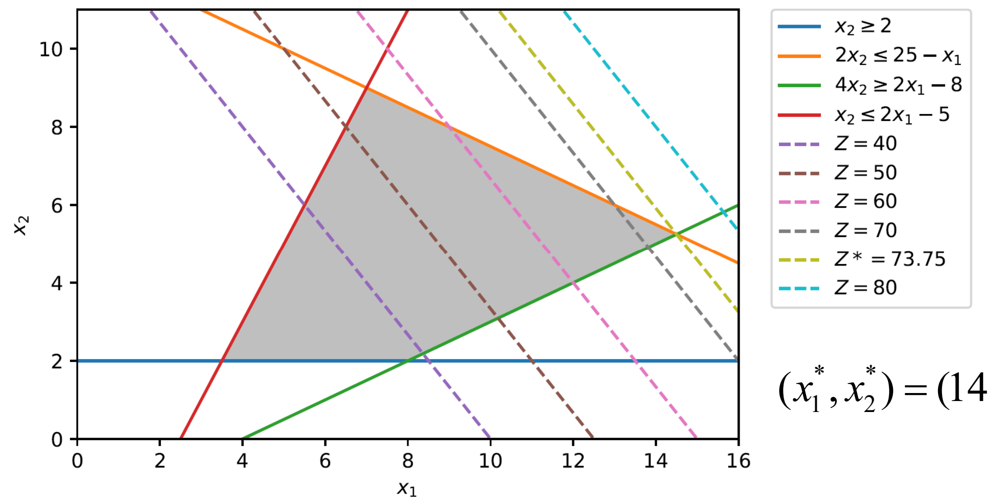
Optimal Solution: Computed by solvers using standard algorithms

For further reading see: Simplex Algorithm,

Convex Polyeder Theory, etc.

Graphical Solutions (Continuous LP)

Example: $\max_{x_1 \geq 0, x_2 \geq 2} 4 \cdot x_1 + 3 \cdot x_2 \quad$ s.t. $\; 2 \cdot x_2 \leq 25 - x_1, \; 4 \cdot x_2 \geq 2x_1 - 8, \; x_2 \leq 2x_1 - 5$

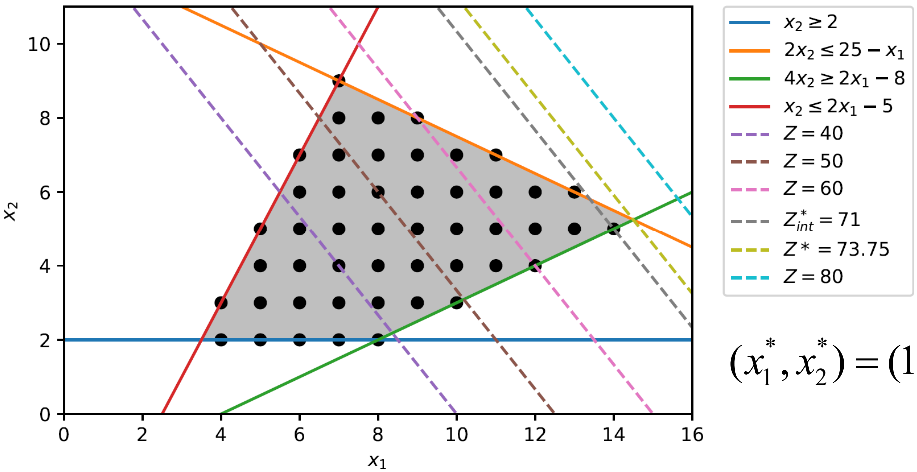


$(x_1^*, x_2^*) = (14.5, 5.25)$

- **Continuous LP:** One corner of the feasible space is always optimal!

Graphical Solutions (Integer LP)

Example: $\max\limits_{\substack{x_1=0,1,2,\ldots \\ x_2=2,3,4,\ldots}} 4\cdot x_1 + 3\cdot x_2$ s.t. $2\cdot x_2 \le 25 - x_1$, $4\cdot x_2 \ge 2x_1 - 8$, $x_2 \le 2x_1 - 5$



$(x_1^*, x_2^*) = (14, 5)$

- **Integer LP**: Use discrete variables $x_1, x_2, \ldots \in \{0,1\}$ *or* $\in \{0,1,2,\ldots\}$

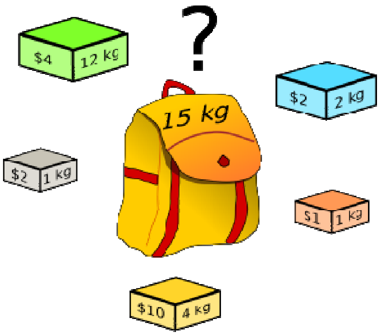
  Optimal solutions via **integer** solvers (see “Branch & Bound”)

Examples, Examples, Examples

- Knapsack Problem
- Magic Square
- Matrix Inversion
- Project Assignment Problems
- Mixed Equilibria (Game Theory)

I Knapsack Problem

Given parameters:



N	Number of potential items
u_i	Utility of packing item i, $i=1,\dots,N$
s_i	Space of item i, $i=1,\dots,N$
C	Capacity of the knapsack

Problem: Decide which items to take in order to maximize total utility while not exceeding the knapsack's capacity

I LP Model for the Knapsack Problem

Variables: x_i binary variable whether to pack item i, i=1,...,N

LP: $$\max_{x_1,\dots,x_N \in \{0,1\}} \sum_{i=1,\dots,N} u_i \cdot x_i$$ maximize total utility

s.t. $$\sum_{i=1,\dots,N} s_i \cdot x_i \le C$$ satisfy budget constraint

Let the solver do the rest!

Done.

I Implementation of the Knapsack LP (AMPL)

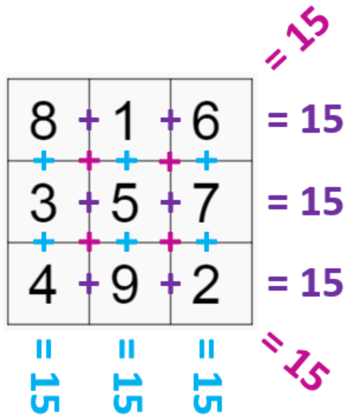
$$\max_{x_1,\ldots,x_N \in \{0,1\}} \sum_{i=1,\ldots,N} u_i \cdot x_i \qquad \text{s.t.} \qquad \sum_{i=1,\ldots,N} s_i \cdot x_i \le C$$

```
reset;
param C := Uniform(50,100);                        # capacity
param N := Uniform(100,200);                       # number of items
param u {i in 1..N} := Uniform(3,8)                # utility of item i
param s {i in 1..N} := Uniform(1,2)                # space of item i

var x {i in 1..N} binary;                          # binary variables

maximize   LP: sum{i in 1..N} u[i]*x[i]; # objective

subject to NB: sum{i in 1..N} s[i]*x[i] <= C;  # budget con.

solve; display x;                                  # obtain solution
```

II Magic Square

$N = n^2$ Numbers i=1,...,N to be filled in a square of size $n \times n$, $n \geq 3$

Problem: Assign the numbers i=1,...,N to the cells of a square such that the sum of each row and each column is the same.



Note: The magic number is $C := N \cdot (N+1)/2/n = n \cdot (n^2+1)/2 \overset{n=3}{=} 15$

II LP Model for the Magic Square

Variables: $x_{i,j,k}$ binary variable whether number k, k=1,...,N is assigned to row i and column j, i,j=1,...,n

LP:
$$\max_{x_{i,j,k}\in\{0,1\}^{N\times n\times n}} \sum_{i=1,\ldots,n,\ j=1,\ldots,n,\ k=1,\ldots,N} x_{i,j,k}$$
(no objective is needed!)

s.t.
$$\sum_{i=1,\ldots,n,\ k=1,\ldots,N} k\cdot x_{i,j,k} = C$$
for all j=1,...,n

$$\sum_{j=1,\ldots,n,\ k=1,\ldots,N} k\cdot x_{i,j,k} = C$$
for all i=1,...,n

$$\sum_{k=1,\ldots,N} x_{i,j,k} = 1$$
for all i,j=1,...,n

$$\sum_{i=1,\ldots,n,\ j=1,\ldots,n} x_{i,j,k} = 1$$
for all k=1,...,N

Hints

- At first: **Think**! Do not start to implement too early
- Do not hope that the LP/solver thinks in your favour!
- Remember: You have to **force** the solver to do the right thing
- Try as hard as you can to formulate the problem **linearly**
- Do not hesitate to use many variables and constraints
- 95% of the work is setting up the LP, 5% is implementation

III Matrix Inversion

$a_{i,j}$ Coefficients of a given matrix A of size $n \times n$, $i,j=1,\ldots,n$

Problem: Find the **inverse matrix** A^{-1} of matrix A

determined by its coefficients $x_{i,j}$, $i,j=1,\ldots,n$.

Note: The inverse A^{-1} is uniquely determined by satisfying:

$$A^{-1} \times A = A \times A^{-1} = I := \begin{pmatrix} 1 & & 0 \\ & \ldots & \\ 0 & & 1 \end{pmatrix}$$

III Implementation of the Matrix Inversion LP

Variables: $x_{i,j}$ continuous variable for the coefficient of row *i* and column *j* of the inverse matrix A^{-1}, *i*,*j*=1,...,*n*

LP: $$\max_{x_{i,j} \in \mathbb{R}^{n \times n}} \sum_{i,j=1,\ldots,n} x_{i,j}$$ (objective is optional!)

s.t. $$\sum_{k=1,\ldots,n} a_{i,k} \cdot x_{k,j} = 1_{\{i=j\}}$$ for all *i*,*j*=1,...,*n*

```
param n := 5;                                       # size of A
param a {i in 1..n,j in 1..n} := Uniform(-1,1); # coeff. of A

var   x {i in 1..n,j in 1..n};                      # coeff. of A^-1

subject to NB{i in 1..n,j in 1..n}:                 # identity
      sum{k in 1..n} a[i,k]*x[k,j] = if i=j then 1 else 0;

solve; display x;                                   # solution output
```

IV Project Assignment Problem

Given parameters:

N Number of workers/projects

$w_{i,j}$ Willingness of worker $i=1,\ldots,N$ to take project $j=1,\ldots,N$

Problem: Decide how to distribute all projects in order to maximize total welfare while assigning one project to each worker

IV LP Model for the Project Assignment Problem

Variables: $x_{i,j}$ binary variable whether project i, i=1,...,N is assigned to worker j, j=1,...,N

LP:
$$\max_{x_{i,j} \in \{0,1\}^{N \times N}} \sum_{i=1,\ldots,N,\ \ j=1,\ldots,N} w_{i,j} \cdot x_{i,j}$$

s.t.
$$\sum_{i=1,\ldots,N} x_{i,j} = 1$$ for all j=1,...,N (each worker gets 1 project)

$$\sum_{j=1,\ldots,N} x_{i,j} = 1$$ for all i=1,...,N (each project is assigned)

Next Week

Homework: Try to solve/implement problem example I and II!

Describe the HPI Master Project Assignment Problem

Outlook:

- More complex (nonlinear) Examples
- Tricks to linearize Nonlinearities
- Implementations and Solvers
- Multi-objective Approaches

Overview

2	April 25	Linear Programming
3	**April 29**	**Linear Programming II**
4	May 2	Linear/Logistic Regression
5	May 6	Exercise Implementations
6	May 16	Dynamic Programming I
7	May 20	Dynamic Programming II
8	May 23	Response Strategies / Game Theory
9	May 27	Project Assignments
10	June 3	Robust Optimization
11	June 13	Workshop / Group Meetings
12	June 20	Presentations (First Results)
13/14	June 24/27	Workshop / Group Meetings
15/16	July 1/4	Workshop / Group Meetings
17	July 11	Presentations (Final Results), Feedback, Documentation (Aug 31)