合肥學院
HEFEI UNIVERSITY

Metaheuristic Optimization

23. Linear Programming

Thomas Weise · 汤卫思

tweise@hfuu.edu.cn · http://iao.hfuu.edu.cn

Hefei University, South Campus 2
Faculty of Computer Science and Technology
Institute of Applied Optimization
230601 Shushan District, Hefei, Anhui, China
Econ. & Tech. Devel. Zone, Jinxiu Dadao 99

合肥学院 南艳湖校区/南2区
计算机科学与技术系
应用优化研究所
中国 安徽省 合肥市 蜀山区 230601
经济技术开发区 锦绣大道99号

Outline

website

Section Outline

Linear Programming

- Metaheuristics can be used to tackle arbitrary problems.

Linear Programming

- Metaheuristics can be used to tackle arbitrary problems.
- But many problems in the real world are either simple, e.g., sums of products of decision variables with constants

Linear Programming

- Metaheuristics can be used to tackle arbitrary problems.
- But many problems in the real world are either simple, e.g., sums of products of decision variables with constants
- The more knowledge we have about a problem, the better we can solve it.

Linear Programming

- Metaheuristics can be used to tackle arbitrary problems.
- But many problems in the real world are either simple, e.g., sums of products of decision variables with constants
- The more knowledge we have about a problem, the better we can solve it.
- . . . because we can design specialized algorithms

Linear Programming

- Metaheuristics can be used to tackle arbitrary problems.
- But many problems in the real world are either simple, e.g., sums of products of decision variables with constants
- The more knowledge we have about a problem, the better we can solve it.
- . . . because we can design specialized algorithms, i.e., algorithms which do not need to handle any complexity not present in the problem and instead make maximum use of the knowledge we have about the nature of the problem

Linear Programming

- Metaheuristics can be used to tackle arbitrary problems.
- But many problems in the real world are either simple, e.g., sums of products of decision variables with constants
- The more knowledge we have about a problem, the better we can solve it.
- . . . because we can design specialized algorithms, i.e., algorithms which do not need to handle any complexity not present in the problem and instead make maximum use of the knowledge we have about the nature of the problem
- The best algorithms for many classical problems, such as the Traveling Salesman Problem or the Maxmum Satisfiability Problem, are metaheuristics that make extensive use of the knowledge about the nature of these problems

Linear Programming

- Metaheuristics can be used to tackle arbitrary problems.
- But many problems in the real world are either simple, e.g., sums of products of decision variables with constants
- The more knowledge we have about a problem, the better we can solve it.
- . . . because we can design specialized algorithms, i.e., algorithms which do not need to handle any complexity not present in the problem and instead make maximum use of the knowledge we have about the nature of the problem
- The best algorithms for many classical problems, such as the Traveling Salesman Problem or the Maxmum Satisfiability Problem, are metaheuristics that make extensive use of the knowledge about the nature of these problems
- However, the vast majority of industry applications in optimization do not use metaheuristics.

Linear Programming

- Metaheuristics can be used to tackle arbitrary problems.
- But many problems in the real world are either simple, e.g., sums of products of decision variables with constants
- The more knowledge we have about a problem, the better we can solve it.
- . . . because we can design specialized algorithms, i.e., algorithms which do not need to handle any complexity not present in the problem and instead make maximum use of the knowledge we have about the nature of the problem
- The best algorithms for many classical problems, such as the Traveling Salesman Problem or the Maxmum Satisfiability Problem, are metaheuristics that make extensive use of the knowledge about the nature of these problems
- However, the vast majority of industry applications in optimization do not use metaheuristics.
- They use Linear Programming tools such as as CPLEX and GAMS.

Definition (Linear Programming)

Linear programming (LP) is a method to find the optimal solution of a problem whose (potentially multiple) constraints and (single) objective are linear relationships.

Definition (Linear Programming)

Linear programming (LP) is a method to find the optimal solution of a problem whose (potentially multiple) constraints and (single) objective are linear relationships.

- The objective function f is a simple linear function of n real-valued decision variables $x_1, x_2, \ldots, x_n \in \mathbb{R}$, i.e., we can write $f(\vec{x}) : \mathbb{R}^n \mapsto \mathbb{R}$.

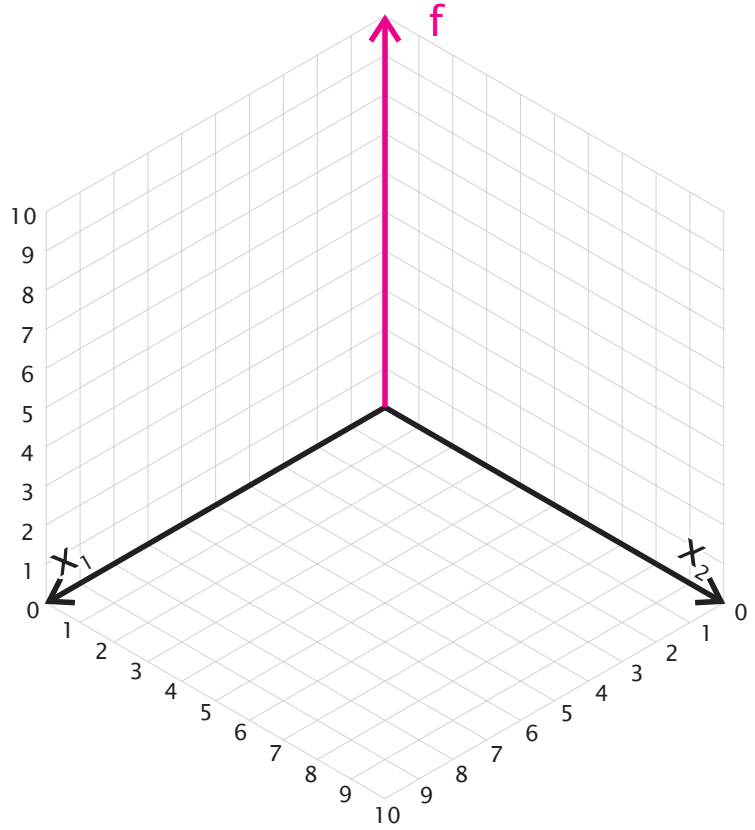
Definition (Linear Programming)

Linear programming (LP) is a method to find the optimal solution of a problem whose (potentially multiple) constraints and (single) objective are linear relationships.

- The objective function f is a simple linear function of n real-valued decision variables $x_1, x_2, \ldots, x_n \in \mathbb{R}$, i.e., we can write $f(\vec{x}) : \mathbb{R}^n \mapsto \mathbb{R}$.
- There are any number m of inequality constraints g_i and any number of equality constraints h_j (see Lesson 16: *Constraint Handling*), each of which are linear.

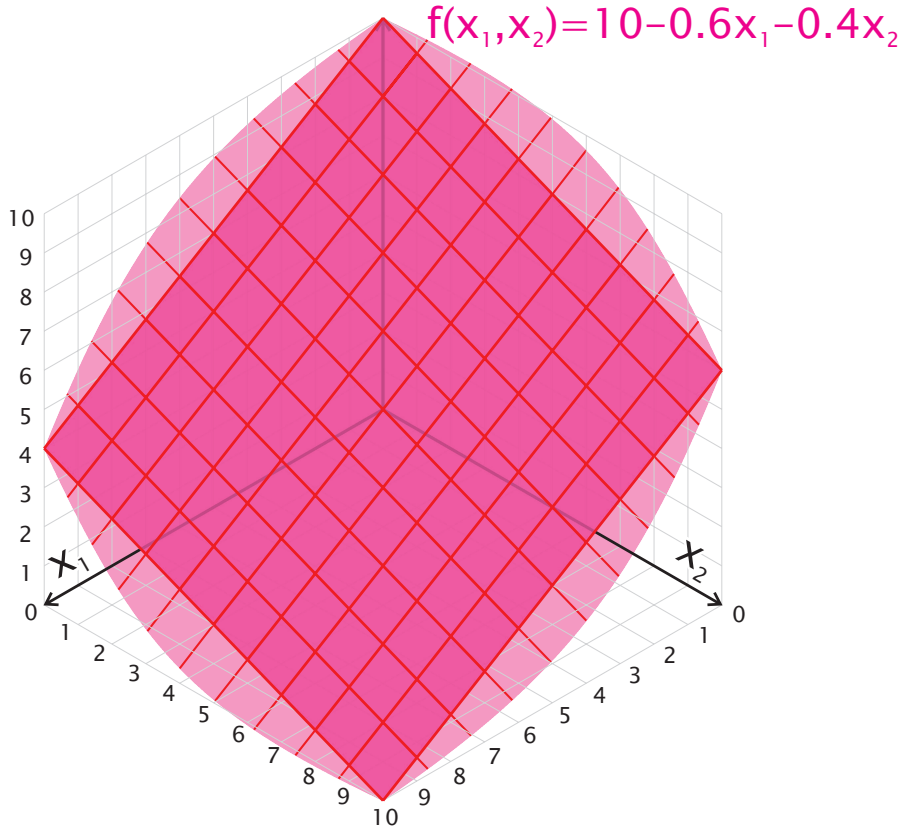
Example

- Let us define a simple linear programming problem with two decision variables and an objective function.



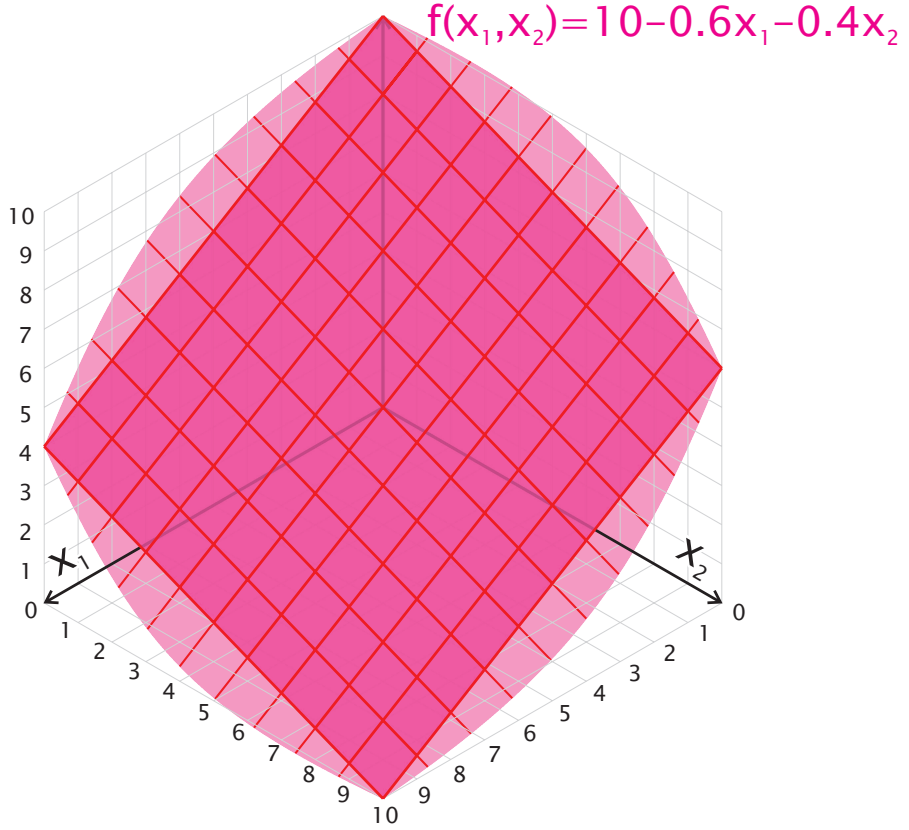
Example

- Let us define a simple linear programming problem with two decision variables and an objective function.
- Assume we want to maximize the objective function $f(x_1, x_2) = 10 - 0.6x_1 - 0.4x_2$ with the two decision variables x_1 and x_2.



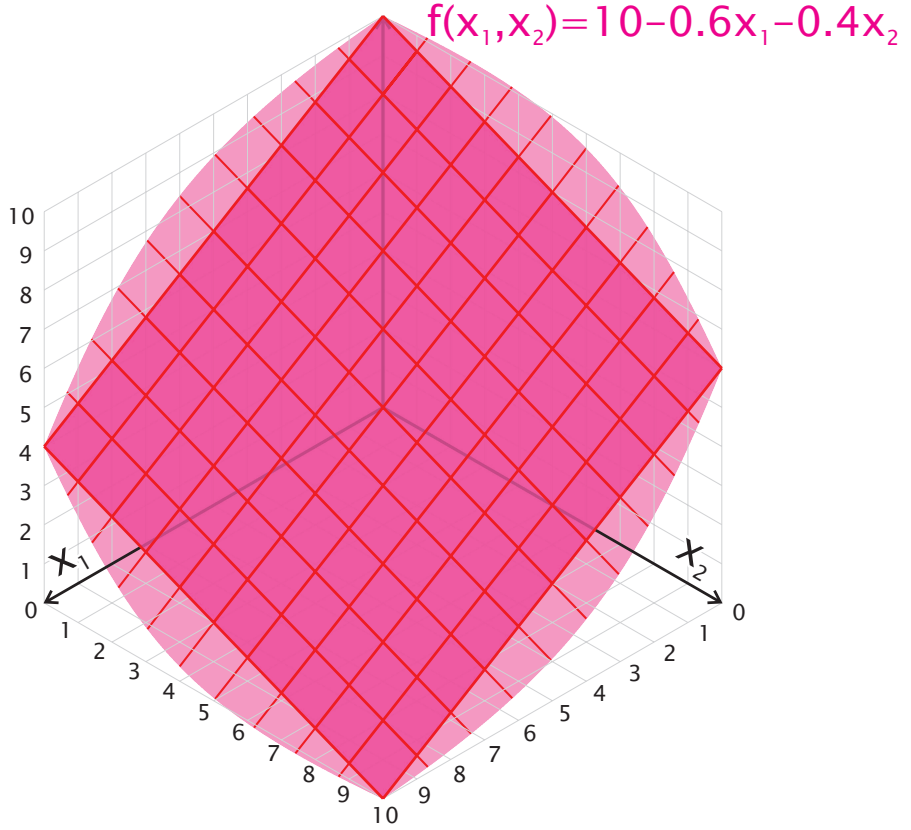
Example

- Let us define a simple linear programming problem with two decision variables and an objective function.
- Assume we want to maximize the objective function $f(x_1, x_2) = 10 - 0.6x_1 - 0.4x_2$ with the two decision variables x_1 and x_2.
- This function would have its optimum somewhere at $x_1 \to \infty, x_2 \to \infty$, because it is unconstraint. There are no limits on x_1 and x_2, they can become arbitrarily big.



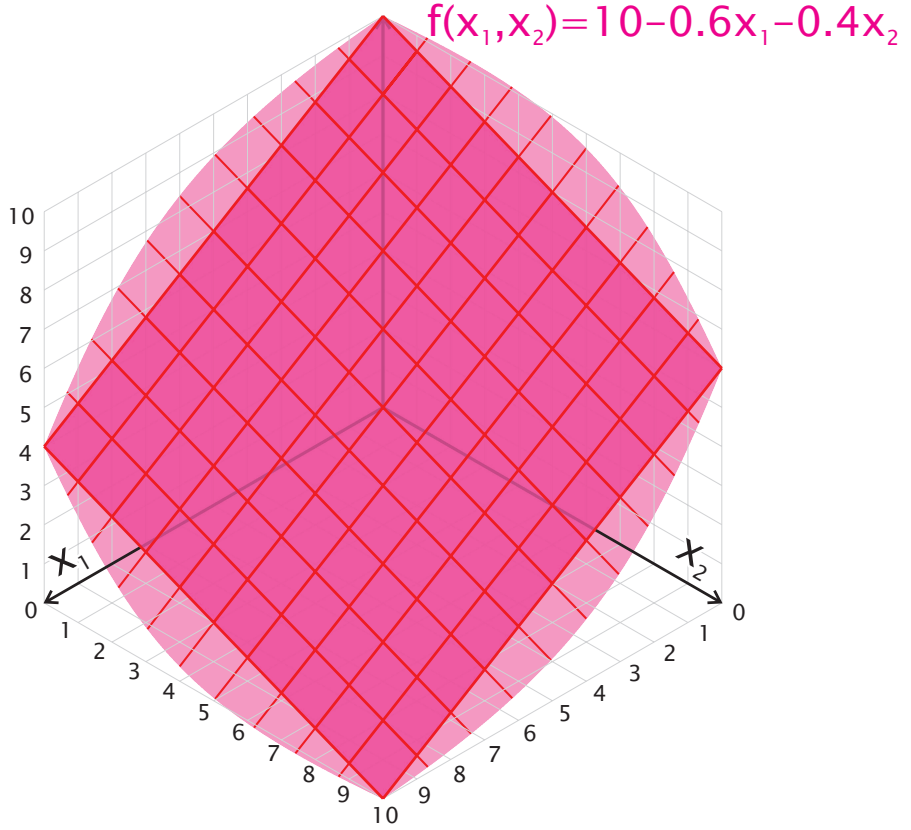
Example

- Let us define a simple linear programming problem with two decision variables and an objective function.
- Assume we want to maximize the objective function $f(x_1, x_2) = 10 - 0.6x_1 - 0.4x_2$ with the two decision variables x_1 and x_2.
- This function would have its optimum somewhere at $x_1 \to \infty, x_2 \to \infty$, because it is unconstraint. There are no limits on x_1 and x_2, they can become arbitrarily big.
- Usually, the values of variables are limited in some way.



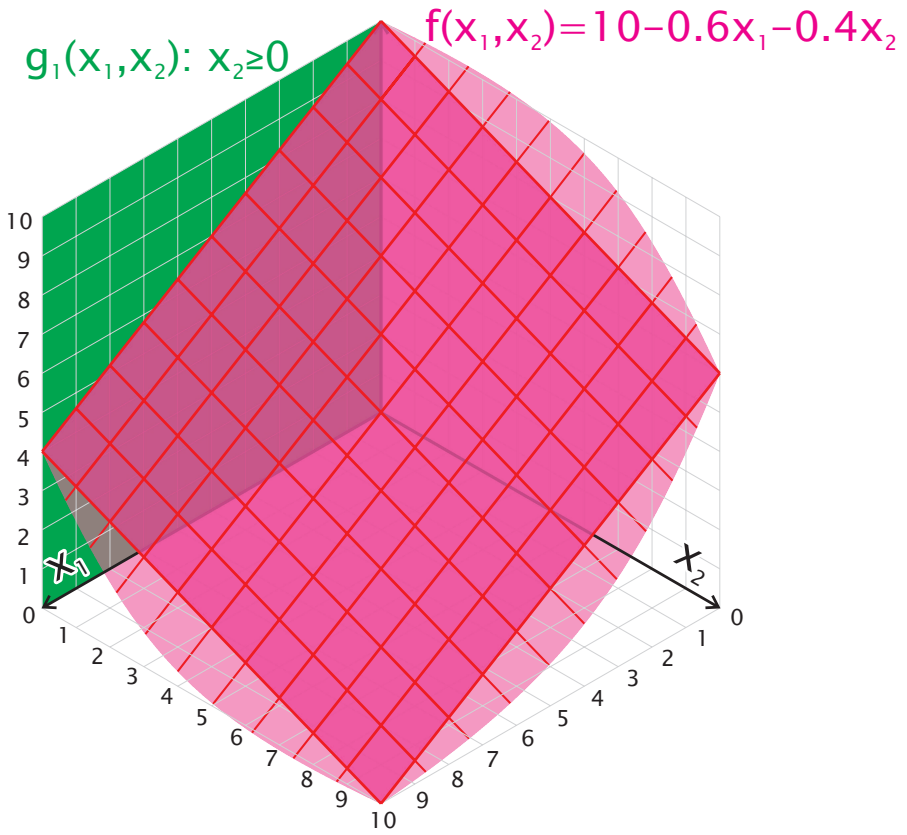
Example

- Let us define a simple linear programming problem with two decision variables and an objective function.
- Assume we want to maximize the objective function $f(x_1, x_2) = 10 - 0.6x_1 - 0.4x_2$ with the two decision variables x_1 and x_2.
- This function would have its optimum somewhere at $x_1 \to \infty, x_2 \to \infty$, because it is unconstraint. There are no limits on x_1 and x_2, they can become arbitrarily big.
- Usually, the values of variables are limited in some way. Let's say that both x_1 and x_2 must be positive.



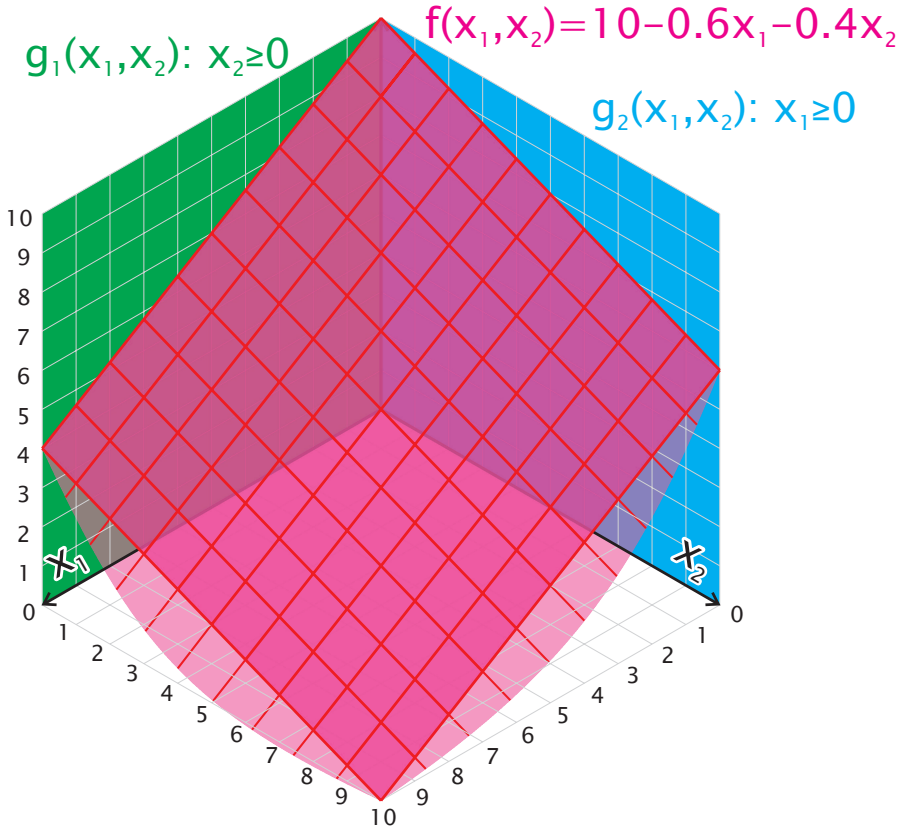
Example

- Let us define a simple linear programming problem with two decision variables and an objective function.
- Assume we want to maximize the objective function $f(x_1, x_2) = 10 - 0.6x_1 - 0.4x_2$ with the two decision variables x_1 and x_2.
- This function would have its optimum somewhere at $x_1 \to \infty, x_2 \to \infty$, because it is unconstraint. There are no limits on x_1 and x_2, they can become arbitrarily big.
- Usually, the values of variables are limited in some way. Let's say that both x_1 and x_2 must be positive.



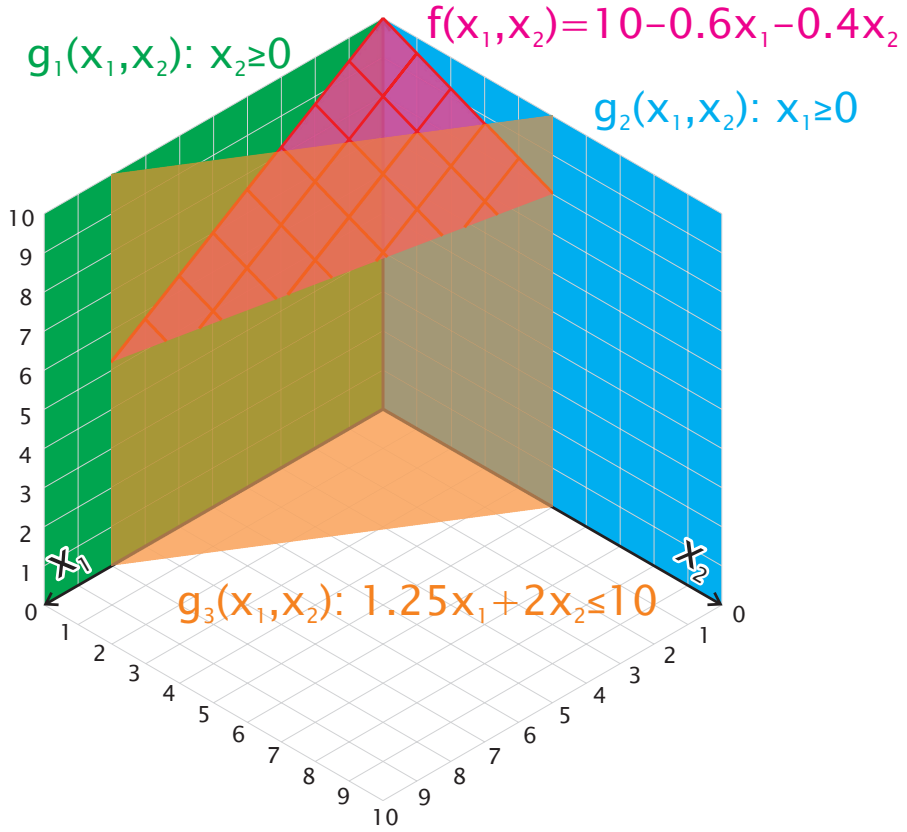
Example

- Let us define a simple linear programming problem with two decision variables and an objective function.
- Assume we want to maximize the objective function $f(x_1, x_2) = 10 - 0.6x_1 - 0.4x_2$ with the two decision variables x_1 and x_2.
- This function would have its optimum somewhere at $x_1 \to \infty, x_2 \to \infty$, because it is unconstraint. There are no limits on x_1 and x_2, they can become arbitrarily big.
- Usually, the values of variables are limited in some way. Let's say that both x_1 and x_2 must be positive.



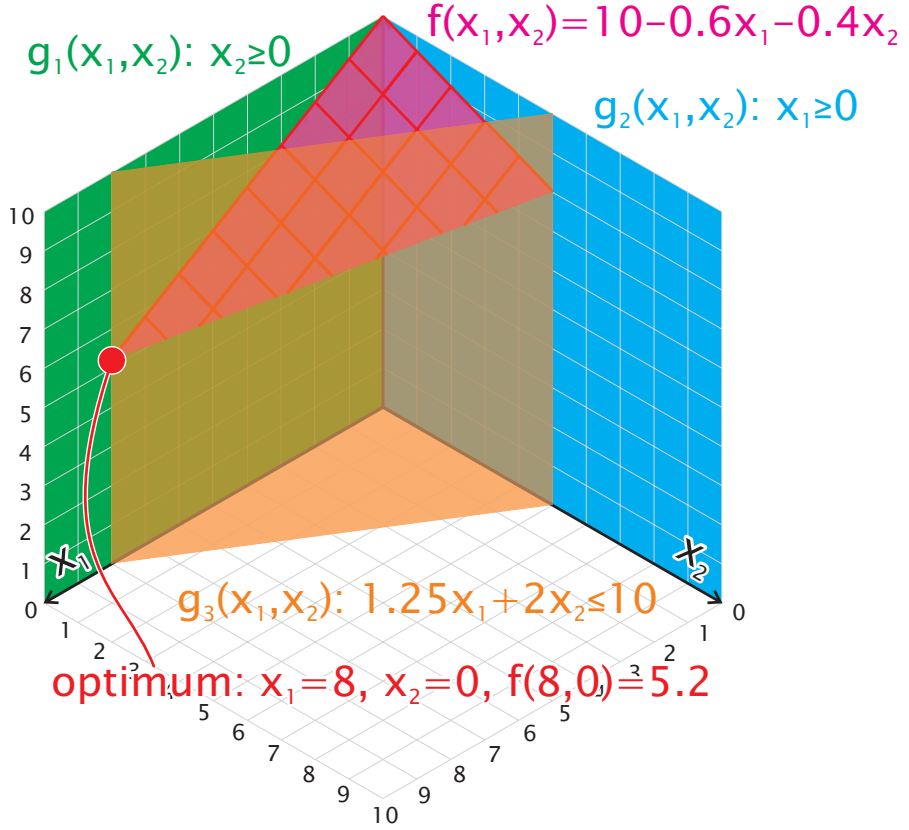
Example

- Let us define a simple linear programming problem with two decision variables and an objective function.
- Assume we want to maximize the objective function $f(x_1, x_2) = 10 - 0.6x_1 - 0.4x_2$ with the two decision variables x_1 and x_2.
- This function would have its optimum somewhere at $x_1 \to \infty, x_2 \to \infty$, because it is unconstraint. There are no limits on x_1 and x_2, they can become arbitrarily big.
- Usually, the values of variables are limited in some way. Let's say that both x_1 and x_2 must be positive.
- Furthermore, let's say that $1.25x_1 + 2x_2 \leq 10$ should hold.



Example

- Let us define a simple linear programming problem with two decision variables and an objective function.
- Assume we want to maximize the objective function $f(x_1, x_2) = 10 - 0.6x_1 - 0.4x_2$ with the two decision variables x_1 and x_2.
- This function would have its optimum somewhere at $x_1 \to \infty, x_2 \to \infty$, because it is unconstraint. There are no limits on x_1 and x_2, they can become arbitrarily big.
- Usually, the values of variables are limited in some way. Let's say that both x_1 and x_2 must be positive.
- Furthermore, let's say that $1.25x_1 + 2x_2 \leq 10$ should hold.
- The search space is now limited by linear constraints and within these constraints, the objective function has an optimum with value 5.2 at $x_1 = 8$ and $x_2 = 0$.



Example Discussion

- The n decision variables span a n-dimensional space

Example Discussion

- The n decision variables span a n-dimensional space
- The objective function f assigns a value to each point in this space

Example Discussion

- The n decision variables span a n-dimensional space
- The objective function f assigns a value to each point in this space
- The m constraints define limits on which values of the decision variables are permitted / feasible

Example Discussion

- The n decision variables span a n-dimensional space
- The objective function f assigns a value to each point in this space
- The m constraints define limits on which values of the decision variables are permitted / feasible
- They can also define interactions among the decision varibles

Example Discussion

- The n decision variables span a n-dimensional space
- The objective function f assigns a value to each point in this space
- The m constraints define limits on which values of the decision variables are permitted / feasible
- They can also define interactions among the decision varibles
- There are many, many problems in the real-world which fit to this schema

Example Discussion

- The n decision variables span a n-dimensional space
- The objective function f assigns a value to each point in this space
- The m constraints define limits on which values of the decision variables are permitted / feasible
- They can also define interactions among the decision varibles
- There are many, many problems in the real-world which fit to this schema
- There are several highly efficient tools for solving such problems to optimality

Example Discussion

- The n decision variables span a n-dimensional space
- The objective function f assigns a value to each point in this space
- The m constraints define limits on which values of the decision variables are permitted / feasible
- They can also define interactions among the decision varibles
- There are many, many problems in the real-world which fit to this schema
- There are several highly efficient tools for solving such problems to optimality
- If we can bring a problem into this schema, we can apply these tools and get the best possible solution quickly

Formal Definition

- But what is this "schema"?

Formal Definition

- But what is this "schema"?
- Formally, we have:

$$\textit{minimize} \quad f(\vec{x}) \quad = \quad \vec{c}^T \, \vec{x} \tag{2}$$

- where $\vec{c}$ is a n-dimensional cost vector, $\cdot^T$ means transposition, $\vec{x}$ the n-dimensional vector of decision variables

Formal Definition

- But what is this "schema"?
- Formally, we have:

$$\textit{minimize} \quad f(\vec{x}) = \vec{c}^T \vec{x} \tag{1}$$

$$\textit{subject to} \quad A\vec{x} \geq \vec{b} \tag{2}$$

- where $\vec{c}$ is a n-dimensional cost vector, $\cdot^T$ means transposition, $\vec{x}$ the n-dimensional vector of decision variables, A a $m \times n$ matrix and $\vec{b}$ another vector used to specify the constraints, and the "$\geq$" is to be understood as element-wise comparison

Formal Definition (Example)

- Let us rephrase our example problem to fit to this schema

Formal Definition (Example)

- Let us rephrase our example problem to fit to this schema
- We first rewrite the cost function f into this vector shape

Formal Definition (Example)

- Let us rephrase our example problem to fit to this schema
- We first rewrite the cost function f into this vector shape
- The cost function $f(\vec{x}) = 10 - 0.6x_1 - 0.4x_2$ from our example can be reduced to $F(\vec{x}) = -0.6x_1 + -0.4x_2$ since the constant offset does not matter, the solution vector $\vec{x}$ woud be the same

Formal Definition (Example)

- Let us rephrase our example problem to fit to this schema
- We first rewrite the cost function f into this vector shape
- The cost function $f(\vec{x}) = 10 - 0.6x_1 - 0.4x_2$ from our example can be reduced to $F(\vec{x}) = -0.6x_1 + -0.4x_2$ since the constant offset does not matter, the solution vector $\vec{x}$ woud be the same
- It can also be written as $(-0.6 - 0.4) \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ as row vector/column vector product

Formal Definition (Example)

- Let us rephrase our example problem to fit to this schema
- We first rewrite the cost function f into this vector shape
- The cost function $f(\vec{x}) = 10 - 0.6x_1 - 0.4x_2$ from our example can be reduced to $F(\vec{x}) = -0.6x_1 + -0.4x_2$ since the constant offset does not matter, the solution vector $\vec{x}$ woud be the same
- It can also be written as $(-0.6 - 0.4) \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ as row vector/column vector product
- so we get $\vec{c} = \begin{pmatrix} -0.6 \\ -0.4 \end{pmatrix}$, i.e.,

$$F(\vec{x}) = \vec{c}^T \vec{x} = \begin{pmatrix} -0.6 \\ -0.4 \end{pmatrix}^T \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \tag{3}$$

Formal Definition (Example)

- Let us rephrase our example problem to fit to this schema
- We first rewrite the cost function f into this vector shape
- The cost function $f(\vec{x}) = 10 - 0.6x_1 - 0.4x_2$ from our example can be reduced to $F(\vec{x}) = -0.6x_1 + -0.4x_2$ since the constant offset does not matter, the solution vector $\vec{x}$ woud be the same
- It can also be written as $(-0.6 - 0.4) \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ as row vector/column vector product
- so we get $\vec{c} = \begin{pmatrix} -0.6 \\ -0.4 \end{pmatrix}$, i.e.,

$$F(\vec{x}) = \vec{c}^T \vec{x} = \begin{pmatrix} -0.6 \\ -0.4 \end{pmatrix}^T \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \tag{3}$$

(if we solve the problem using F and get the optimal values for the decision variables x_1 and x_2, we can later compute the value of f)

Formal Definition (Example)

- Now we need to translate the $m = 3$ constraints $g_1 : x_2 \geq 0$, $g_2 : x_1 \geq 0$, and $g_3 : 1.25x_1 + 2x_2 \leq 10$ into the form $A\,\vec{x} \geq \vec{b}$

$$\begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \\ A_{3,1} & A_{3,2} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \geq \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} \tag{4}$$

Formal Definition (Example)

- Now we need to translate the $m = 3$ constraints $g_1 : x_2 \geq 0$, $g_2 : x_1 \geq 0$, and $g_3 : 1.25x_1 + 2x_2 \leq 10$ into the form $A\,\vec{x} \geq \vec{b}$
- The first two constraints can be directly transformed to rows in A with a corresponding 0 in $\vec{b}$

$$\begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \\ A_{3,1} & A_{3,2} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \geq \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} \tag{4}$$

Formal Definition (Example)

- Now we need to translate the $m = 3$ constraints $g_1 : x_2 \geq 0$, $g_2 : x_1 \geq 0$, and $g_3 : 1.25x_1 + 2x_2 \leq 10$ into the form $A\,\vec{x} \geq \vec{b}$
- The first two constraints can be directly transformed to rows in A with a corresponding 0 in $\vec{b}$

$$\begin{pmatrix} 0 & 1 \\ A_{2,1} & A_{2,2} \\ A_{3,1} & A_{3,2} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \geq \begin{pmatrix} 0 \\ b_2 \\ b_3 \end{pmatrix} \tag{4}$$

Formal Definition (Example)

- Now we need to translate the $m = 3$ constraints $g_1 : x_2 \geq 0$, $g_2 : x_1 \geq 0$, and $g_3 : 1.25x_1 + 2x_2 \leq 10$ into the form $A\,\vec{x} \geq \vec{b}$
- The first two constraints can be directly transformed to rows in A with a corresponding 0 in $\vec{b}$

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \\ A_{3,1} & A_{3,2} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \geq \begin{pmatrix} 0 \\ 0 \\ b_3 \end{pmatrix} \tag{4}$$

Formal Definition (Example)

- Now we need to translate the $m = 3$ constraints $g_1 : x_2 \geq 0$, $g_2 : x_1 \geq 0$, and $g_3 : 1.25x_1 + 2x_2 \leq 10$ into the form $A\,\vec{x} \geq \vec{b}$
- The first two constraints can be directly transformed to rows in A with a corresponding 0 in $\vec{b}$
- The third constraint $g_3 : 1.25x_1 + 2x_2 \leq 10$ can be turned around to become $g_3 : -1.25x_1 + -2x_2 \geq -10$

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \\ A_{3,1} & A_{3,2} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \geq \begin{pmatrix} 0 \\ 0 \\ b_3 \end{pmatrix} \tag{4}$$

Formal Definition (Example)

- Now we need to translate the $m = 3$ constraints $g_1 : x_2 \geq 0$, $g_2 : x_1 \geq 0$, and $g_3 : 1.25x_1 + 2x_2 \leq 10$ into the form $A\,\vec{x} \geq \vec{b}$
- The first two constraints can be directly transformed to rows in A with a corresponding 0 in $\vec{b}$
- The third constraint $g_3 : 1.25x_1 + 2x_2 \leq 10$ can be turned around to become $g_3 : -1.25x_1 + -2x_2 \geq -10$

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \\ -1.25 & -2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \geq \begin{pmatrix} 0 \\ 0 \\ -10 \end{pmatrix} \tag{4}$$

Section Outline

General Algebraic Modeling System (GAMS)

- Such problems can be solved using programs such as CPLEX

General Algebraic Modeling System (GAMS)

- Such problems can be solved using programs such as CPLEX
- For doing so, we can conveniently express them as models, using, for instance the General Algebraic Modeling System (GAMS), which you can download from `http://www.gams.com`

GAMS Example: Definition

- The previous example looks like this in GAMS notation:

Listing: example1.gams

```
Variables f, x1, x2;

Equations obj, g1, g2, g3;

obj.. f =e= 10 - 0.6*x1 - 0.4*x2;
g1..  x2 =g= 0;
g2..  x1 =g= 0;
g3..  1.25*x1 + 2*x2 =l= 10;

Model example1 / obj, g1, g2, g3 /;

solve example1 using LP minimizing f;
```

- where `=e=` means "$=$", `=l=` means "$\leq$", and `=g=` means "$\geq$"

GAMS Example: Execution

- We now apply GAMS:

Listing: Shell command to start GAMS

```
gams example1.gams -o example1.lst -logOption 2
```

- where `-o example1.lst` tells the system to store the output in file `example1.lst`
- and `-logOption 2` stores the console/log output in file `example1.log`
- non-printable characters in `example1.lst` can be removed via `tr -d '000-011013014016-037' < example1.lst > example1.lst2` in Linux

GAMS Example: Log Output (1)

- The log output to the console stored in `example1.log` is:

Listing: example1.gams, lines 1–14

```
--- Job example1.gams Start 11/28/17 10:10:55 24.9.2 r64480 LEX-LEG x86 64bit/Linux
GAMS 24.9.2   Copyright (C) 1987-2017 GAMS Development. All rights reserved
Licensee: GAMS Development Corporation, USA                 G871201/0000CA-ANY
          Free Demo, +1 202-342-0180, support@gams.com, www.gams.com DC0000
--- Starting compilation
--- example1.gams(12) 2 Mb
--- Starting execution: elapsed 0:00:00.001
--- Generating LP model example1
--- example1.gams(12) 3 Mb
---   4 rows  3 columns  7 non-zeroes
--- Executing CPLEX: elapsed 0:00:00.002

IBM ILOG CPLEX   24.9.2 r64480 Released Nov 14, 2017 LEG x86 64bit/Linux
Cplex 12.7.1.0
```

GAMS Example: Log Output (2)

- The log output to the console stored in `example1.log` is:

Listing: `example1.gams`, lines 16–32

```
Reading data...
Starting Cplex...
Space for names approximately 0.00 Mb
Use option 'names no' to turn use of names off
CPXPARAM_Advance                                 0
CPXPARAM_Simplex_Display                         2
CPXPARAM_Simplex_Limits_Iterations               2000000000
CPXPARAM_TimeLimit                               1000
CPXPARAM_Threads                                 1
CPXPARAM_Parallel                                1
CPXPARAM_Tune_TimeLimit                          200
Tried aggregator 1 time.
LP Presolve eliminated 4 rows and 3 columns.
All rows and columns eliminated.
Presolve time = 0.00 sec. (0.00 ticks)
LP status(1): optimal
Cplex Time: 0.00sec (det. 0.00 ticks)
```

GAMS Example: Log Output (3)

- The log output to the console stored in `example1.log` is:

Listing: example1.gams, lines 34–41

```
Optimal solution found.
Objective :           5.200000

--- Restarting execution
--- example1.gams(12) 2 Mb
--- Reading solution for model example1
*** Status: Normal completion
--- Job example1.gams Stop 11/28/17 10:10:56 elapsed 0:00:00.020
```

GAMS Example: Solution Output (1)

- The output of the result stored in `example1.lst2` is:

Listing: example1.gams, lines 1–20

```
GAMS 24.9.2  r64480 Released Nov 14, 2017 LEX-LEG x86 64bit/Linux

    11/28/17 10:10:55 Page 1
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
C o m p i l a t i o n


   1  Variables f, x1, x2;
   2
   3  Equations obj, g1, g2, g3;
   4
   5  obj.. f =e= 10 - 0.6*x1 - 0.4*x2;
   6  g1..  x2 =g= 0;
   7  g2..  x1 =g= 0;
   8  g3..  1.25*x1 + 2*x2 =l= 10;
   9
  10  Model example1 / obj, g1, g2, g3 /;
  11
  12  solve example1 using LP minimizing f;


COMPILATION TIME     =        0.000 SECONDS      2 MB  24.9.2 r64480 LEX-LEG
```

GAMS Example: Solution Output (2)

- The output of the result stored in `example1.lst2` is:

Listing: `example1.gams`, lines 21–43

```
GAMS 24.9.2  r64480 Released Nov 14, 2017 LEX-LEG x86 64bit/Linux

    11/28/17 10:10:55 Page 2
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
Equation Listing    SOLVE example1 Using LP From line 12


---- obj  =E=

obj..  f + 0.6*x1 + 0.4*x2 =E= 10 ; (LHS = 0, INFES = 10 ****)


---- g1  =G=

g1..  x2 =G= 0 ; (LHS = 0)


---- g2  =G=

g2..  x1 =G= 0 ; (LHS = 0)


---- g3  =L=

g3..  1.25*x1 + 2*x2 =L= 10 ; (LHS = 0)
```

GAMS Example: Solution Output (3)

- The output of the result stored in `example1.lst2` is:

Listing: example1.gams, lines 45–72

```
GAMS 24.9.2  r64480 Released Nov 14, 2017 LEX-LEG x86 64bit/Linux

    11/28/17 10:10:55 Page 3
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
Column Listing      SOLVE example1 Using LP From line 12


---- f

f
                (.LO, .L, .UP, .M = -INF, 0, +INF, 0)
        1       obj


---- x1

x1
                (.LO, .L, .UP, .M = -INF, 0, +INF, 0)
        0.6     obj
        1       g2
        1.25    g3


---- x2

x2
                (.LO, .L, .UP, .M = -INF, 0, +INF, 0)
        0.4     obj
        1       g1
        2       g3
```

GAMS Example: Solution Output (4)

- The output of the result stored in `example1.lst2` is:

Listing: `example1.gams`, lines 74–89

```
GAMS 24.9.2  r64480 Released Nov 14, 2017 LEX-LEG x86 64bit/Linux

    11/28/17 10:10:55 Page 4
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
Model Statistics    SOLVE example1 Using LP From line 12


MODEL STATISTICS

BLOCKS OF EQUATIONS           4     SINGLE EQUATIONS            4
BLOCKS OF VARIABLES           3     SINGLE VARIABLES            3
NON ZERO ELEMENTS             7


GENERATION TIME      =        0.001 SECONDS      3 MB  24.9.2 r64480 LEX-LEG


EXECUTION TIME       =        0.001 SECONDS      3 MB  24.9.2 r64480 LEX-LEG
```

GAMS Example: Solution Output (5)

- The output of the result stored in `example1.lst2` is:

Listing: example1.gams, lines 90–116

```
GAMS 24.9.2  r64480 Released Nov 14, 2017 LEX-LEG x86 64bit/Linux

    11/28/17 10:10:55 Page 5
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
Solution Report     SOLVE example1 Using LP From line 12


               S O L V E      S U M M A R Y

     MODEL   example1            OBJECTIVE  f
     TYPE    LP                  DIRECTION  MINIMIZE
     SOLVER  CPLEX               FROM LINE  12

**** SOLVER STATUS     1 Normal Completion
**** MODEL STATUS      1 Optimal
**** OBJECTIVE VALUE                5.2000

 RESOURCE USAGE, LIMIT          0.003      1000.000
 ITERATION COUNT, LIMIT         0    2000000000

IBM ILOG CPLEX   24.9.2 r64480 Released Nov 14, 2017 LEG x86 64bit/Linux
Cplex 12.7.1.0

Space for names approximately 0.00 Mb
Use option 'names no' to turn use of names off
LP status(1): optimal
Cplex Time: 0.00sec (det. 0.00 ticks)
Optimal solution found.
Objective :           5.200000
```

GAMS Example: Solution Output (6)

- The output of the result stored in `example1.lst2` is:

Listing: `example1.gams`, lines 116–138

```
                           LOWER          LEVEL          UPPER         MARGINAL

---- EQU obj                10.0000        10.0000        10.0000         1.0000
---- EQU g1                   .              .            +INF           0.5600
---- EQU g2                   .            8.0000         +INF            .
---- EQU g3                 -INF           10.0000        10.0000        -0.4800

                           LOWER          LEVEL          UPPER         MARGINAL

---- VAR f                  -INF            5.2000        +INF            .
---- VAR x1                 -INF            8.0000        +INF            .
---- VAR x2                 -INF             .            +INF            .


**** REPORT SUMMARY :        0     NONOPT
                             0 INFEASIBLE
                             0  UNBOUNDED


EXECUTION TIME       =        0.001 SECONDS      2 MB   24.9.2 r64480 LEX-LEG
```

GAMS Example: Solution Output (7)

- The output of the result stored in `example1.lst2` is:

Listing: `example1.gams`, lines 141–53

```
USER: GAMS Development Corporation, USA              G871201/0000CA-ANY
      Free Demo, +1 202-342-0180, support@gams.com, www.gams.com DC0000


**** FILE SUMMARY

Input      example1.gams
GAMS 24.9.2  r64480 Released Nov 14, 2017 LEX-LEG x86 64bit/Linux

    11/28/17 10:10:55 Page 6
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
Solution Report     SOLVE example1 Using LP From line 12


Output     example1.lst
```

Section Outline

A Second Example

- We now discuss a second example, directly taken from the GAMS tutorial by Rosenthal [1], which in turn based on a transportation problem defined by Dantzig [2]

A Second Example

- We now discuss a second example, directly taken from the GAMS tutorial by Rosenthal [1], which in turn based on a transportation problem defined by Dantzig [2]
- We know
  - there are several factories i that all produce the same product

A Second Example

- We now discuss a second example, directly taken from the GAMS tutorial by Rosenthal [1], which in turn based on a transportation problem defined by Dantzig [2]
- We know
  - there are several factories i that all produce the same product and we know
  - the supplies a_i of that product produced at the factory i

A Second Example

- We now discuss a second example, directly taken from the GAMS tutorial by Rosenthal [1], which in turn based on a transportation problem defined by Dantzig [2]
- We know
  - there are several factories i that all produce the same product and we know
  - the supplies a_i of that product produced at the factory i,
  - there are several markets j

A Second Example

- We now discuss a second example, directly taken from the GAMS tutorial by Rosenthal [1], which in turn based on a transportation problem defined by Dantzig [2]
- We know
  - there are several factories i that all produce the same product and we know
  - the supplies a_i of that product produced at the factory i,
  - there are several markets j and we know
  - the demands b_j for the product at each market j

A Second Example

- We now discuss a second example, directly taken from the GAMS tutorial by Rosenthal [1], which in turn based on a transportation problem defined by Dantzig [2]
- We know
  - there are several factories i that all produce the same product and we know
  - the supplies a_i of that product produced at the factory i,
  - there are several markets j and we know
  - the demands b_j for the product at each market j, and
  - the costs $c_{i,j}$ of shipping one unit the commodity from each factory i to each market j.

A Second Example

- We now discuss a second example, directly taken from the GAMS tutorial by Rosenthal [1], which in turn based on a transportation problem defined by Dantzig [2]
- We know
  - there are several factories i that all produce the same product and we know
  - the supplies a_i of that product produced at the factory i,
  - there are several markets j and we know
  - the demands b_j for the product at each market j, and
  - the costs $c_{i,j}$ of shipping one unit the commodity from each factory i to each market j.
- For each factory i and market j, we want to find the exact amount $x_{i,j}$ of the product to ship from i to j such that the overall transportation cost f is minimized (obviously $x_{i,j} \geq 0 \; \forall i, j$).

A Second Example: Constraints

- The goal is to minimize the overall transportation costs f

A Second Example: Constraints

- The goal is to minimize the overall transportation costs f, which is the unit transportation cost from each factory i to each market j times the amount $x_{i,j}$ of the product transported from i to j

$$\text{minimize } f(x) \quad = \quad \sum_{\forall i,j} c_{i,j} * x_{i,j} \tag{7}$$

A Second Example: Constraints

- The goal is to minimize the overall transportation costs f
- Clearly, if we transport nothing to nowhere, we get cost 0...

$$\text{minimize } f(x) \quad = \quad \sum_{\forall i,j} c_{i,j} * x_{i,j} \tag{7}$$

A Second Example: Constraints

- The goal is to minimize the overall transportation costs f
- Clearly, if we transport nothing to nowhere, we get cost 0... ...but there are two constraints limiting our choices

$$\text{minimize } f(x) = \sum_{\forall i,j} c_{i,j} * x_{i,j} \tag{7}$$

A Second Example: Constraints

- The goal is to minimize the overall transportation costs f
- There are two constraints limiting our choices:
- We cannot take more supplies from a factory i than the supply a_i available in i

$$\text{minimize } f(x) \quad = \quad \sum_{\forall i,j} c_{i,j} * x_{i,j} \tag{7}$$

A Second Example: Constraints

- The goal is to minimize the overall transportation costs f
- There are two constraints limiting our choices:
- We cannot take more supplies from a factory i than the supply a_i available in i, i.e., the sum of all amounts $x_{i,j}$ taken from factory i to all the markets j cannot exceed a_i

$$\text{minimize } f(x) = \sum_{\forall i,j} c_{i,j} * x_{i,j} \tag{5}$$

$$\text{subject to } \forall i \left(\sum_{\forall j} x_{i,j}\right) \leq a_i \tag{7}$$

A Second Example: Constraints

- The goal is to minimize the overall transportation costs f
- There are two constraints limiting our choices:
- We cannot take more supplies from a factory i than the supply a_i available in i
- We must satisfy the demand b_j of each market

$$\text{minimize } f(x) = \sum_{\forall i,j} c_{i,j} * x_{i,j} \tag{5}$$

$$\text{subject to } \forall i \left(\sum_{\forall j} x_{i,j}\right) \leq a_i \tag{7}$$

A Second Example: Constraints

- The goal is to minimize the overall transportation costs f
- There are two constraints limiting our choices:
- We cannot take more supplies from a factory i than the supply a_i available in i
- We must satisfy the demand b_j of each market, i.e., the sum of all amounts $x_{i,j}$ transported from all factories i to j must be at least b_j

$$\text{minimize } f(x) = \sum_{\forall i,j} c_{i,j} * x_{i,j} \tag{5}$$

$$\text{subject to } \forall i \quad \left(\sum_{\forall j} x_{i,j}\right) \leq a_i \tag{6}$$

$$\forall j \quad \left(\sum_{\forall i} x_{i,j}\right) \geq b_j \tag{7}$$

A Second Example: Constraints

- The goal is to minimize the overall transportation costs f
- There are two constraints limiting our choices:
- We cannot take more supplies from a factory i than the supply a_i available in i
- We must satisfy the demand b_j of each market, i.e., the sum of all amounts $x_{i,j}$ transported from all factories i to j must be at least b_j (actually, no sane solution would exceed b_j, since that would cause unnecessary cost)

$$\text{minimize } f(x) = \sum_{\forall i,j} c_{i,j} * x_{i,j} \tag{5}$$

$$\text{subject to } \forall i \quad \left(\sum_{\forall j} x_{i,j}\right) \leq a_i \tag{6}$$

$$\forall j \quad \left(\sum_{\forall i} x_{i,j}\right) \geq b_j \tag{7}$$

Problem Instance

- The above definition is a general problem, but for each planning task there will be a concrete scenario.

Problem Instance

- The above definition is a general problem, but for each planning task there will be a concrete scenario.
- Let us assume that we have the two plants `nanjingPlant` and `suzhouPlant` with supplies of 350 and 600 units of the product, respectively.

Problem Instance

- The above definition is a general problem, but for each planning task there will be a concrete scenario.
- Let us assume that we have the two plants `nanjingPlant` and `suzhouPlant` with supplies of 350 and 600 units of the product, respectively.
- There also be three markets, `hefeiSuguo`, `beijingSuning`, and `shanghaiTaobao` with demands of 325, 300, and 275, respectively.

Problem Instance

- The above definition is a general problem, but for each planning task there will be a concrete scenario.
- Let us assume that we have the two plants `nanjingPlant` and `suzhouPlant` with supplies of 350 and 600 units of the product, respectively.
- There also be three markets, `hefeiSuguo`, `beijingSuning`, and `shanghaiTaobao` with demands of 325, 300, and 275, respectively.
- The distances in KM between the plants and markets be as follows:

	`hefeiSuguo`	`beijingSuning`	`shanghaiTaobao`
`nanjingPlant`	169.7	1042.3	303.1
`suzhouPlant`	384.4	1150.9	95.6

Problem Instance

- The above definition is a general problem, but for each planning task there will be a concrete scenario.
- Let us assume that we have the two plants `nanjingPlant` and `suzhouPlant` with supplies of 350 and 600 units of the product, respectively.
- There also be three markets, `hefeiSuguo`, `beijingSuning`, and `shanghaiTaobao` with demands of 325, 300, and 275, respectively.
- The distances in KM between the plants and markets be as follows:

	`hefeiSuguo`	`beijingSuning`	`shanghaiTaobao`
`nanjingPlant`	169.7	1042.3	303.1
`suzhouPlant`	384.4	1150.9	95.6

- Transporting one unit of the product for one kilometer costs 2 RMB.

GAMS Model

Listing: transport.gams

```
Sets        i       factories    / nanjingPlant, suzhouPlant /
            j       markets      / hefeiSuguo, beijingSuning, shanghaiTaobao / ;

Parameters  a(i)    supply units available at factory i   /  nanjingPlant   350
                                                              suzhouPlant    600  /

            b(j)    demand at market j in units           /  hefeiSuguo      325
                                                              beijingSuning   300
                                                              shanghaiTaobao  275  / ;

Table       d(i,j)  distance in kilometers
                                        hefeiSuguo    beijingSuning    shanghaiTaobao
                          nanjingPlant      169.7          1042.3           303.1
                          suzhouPlant       384.4          1150.9           95.6   ;

Scalar      tc      transport costs in RMB per unit per kilometer /2/ ;

Parameter   c(i,j)  transport cost in RMB per unit;    c(i,j) = tc * d(i,j);

Variables   x(i,j)  shipment quantities in units
            f       total transportation costs ;

Positive Variable x ;

Equations cost, supply(i), demand(j) ;

cost ..         f  =e=  sum((i,j), c(i,j)*x(i,j)) ;
supply(i) ..    sum(j, x(i,j))  =l=  a(i) ;
demand(j) ..    sum(i, x(i,j))  =g=  b(j) ;

Model transport /all/ ;
Solve transport using lp minimizing f ;
Display x.l, x.m ;
```

Solution

- We can execute and solve the model via

```
gams transport.gams -o transport.lst -logOption 2
```

Solution

- We can execute and solve the model via

```
gams transport.gams -o transport.lst -logOption 2
```

- In `transport.lst`, we find the following solution:

Listing: `transport.lst`, lines 179–183, 196–199

```
                            LOWER          LEVEL          UPPER         MARGINAL

---- VAR f                  -INF       847995.0000        +INF             .

  f  total transportation costs
             hefeiSuguo  beijingSu~  shanghaiT~

nanjingPlant     325.000      25.000
suzhouPlant                  275.000     275.000
```

Solution

- We can execute and solve the model via

```
gams transport.gams -o transport.lst -logOption 2
```

- In `transport.lst`, we find the following solution:

Listing: `transport.lst`, lines 179–183, 196–199

```
                           LOWER          LEVEL          UPPER         MARGINAL

---- VAR f                 -INF       847995.0000        +INF             .

  f  total transportation costs
             hefeiSuguo  beijingSu~  shanghaiT~

nanjingPlant     325.000      25.000
suzhouPlant                  275.000     275.000
```

- The complete demand of 325 of `hefeiSuguo` is covered by the `nanjingPlant`

Solution

- We can execute and solve the model via

```
gams transport.gams -o transport.lst -logOption 2
```

- In `transport.lst`, we find the following solution:

Listing: `transport.lst`, lines 179–183, 196–199

```
                           LOWER          LEVEL          UPPER         MARGINAL

---- VAR f                 -INF       847995.0000        +INF             .

  f  total transportation costs
             hefeiSuguo  beijingSu~  shanghaiT~

nanjingPlant     325.000      25.000
suzhouPlant                  275.000     275.000
```

- The complete demand of 325 of `hefeiSuguo` is covered by the `nanjingPlant`
- The remaining 25 units from the `nanjingPlant` go to the `beijingSuning`

Solution

- We can execute and solve the model via

```
gams transport.gams -o transport.lst -logOption 2
```

- In `transport.lst`, we find the following solution:

Listing: `transport.lst`, lines 179–183, 196–199

```
                           LOWER          LEVEL          UPPER         MARGINAL

---- VAR f                 -INF       847995.0000        +INF             .

  f  total transportation costs
             hefeiSuguo  beijingSu~  shanghaiT~

nanjingPlant     325.000      25.000
suzhouPlant                  275.000     275.000
```

- The complete demand of 325 of `hefeiSuguo` is covered by the `nanjingPlant`
- The remaining 25 units from the `nanjingPlant` go to the `beijingSuning`
- The remaining 275 units required by `beijingSuning` need to be provided by the `suzhouPlant`

Solution

- We can execute and solve the model via

```
gams transport.gams -o transport.lst -logOption 2
```

- In `transport.lst`, we find the following solution:

Listing: `transport.lst`, lines 179–183, 196–199

```
                            LOWER          LEVEL          UPPER         MARGINAL

---- VAR f                  -INF       847995.0000        +INF             .

  f  total transportation costs
              hefeiSuguo   beijingSu~   shanghaiT~

nanjingPlant      325.000       25.000
suzhouPlant                     275.000      275.000
```

- The complete demand of 325 of `hefeiSuguo` is covered by the `nanjingPlant`
- The remaining 25 units from the `nanjingPlant` go to the `beijingSuning`
- The remaining 275 units required by `beijingSuning` need to be provided by the `suzhouPlant`
- The `suzhouPlant` also supplies `shanghaiTaobao` with its required 275 units

Solution

- We can execute and solve the model via

```
gams transport.gams -o transport.lst -logOption 2
```

- In `transport.lst`, we find the following solution:

Listing: `transport.lst`, lines 179–183, 196–199

```
                            LOWER          LEVEL          UPPER         MARGINAL

---- VAR f                  -INF       847995.0000        +INF             .

  f  total transportation costs
              hefeiSuguo   beijingSu~   shanghaiT~

nanjingPlant      325.000       25.000
suzhouPlant                     275.000      275.000
```

- The complete demand of 325 of `hefeiSuguo` is covered by the `nanjingPlant`
- The remaining 25 units from the `nanjingPlant` go to the `beijingSuning`
- The remaining 275 units required by `beijingSuning` need to be provided by the `suzhouPlant`
- The `suzhouPlant` also supplies `shanghaiTaobao` with its required 275 units
- All in all, this incurs a transportation cost of 847'995 RMB

Section Outline

Summary

- Linear Programming technologies are highly advanced and there is rich software and modelling support

Summary

- Linear Programming technologies are highly advanced and there is rich software and modelling support
- As a result, they are often used in many industrial scenarios

Summary

- Linear Programming technologies are highly advanced and there is rich software and modelling support
- As a result, they are often used in many industrial scenarios
- Formulating a problem as Linear Programming task can often be easy, but in other situations it may also be hard and lead to many constraints

Summary

- Linear Programming technologies are highly advanced and there is rich software and modelling support
- As a result, they are often used in many industrial scenarios
- Formulating a problem as Linear Programming task can often be easy, but in other situations it may also be hard and lead to many constraints
- While GAMS provides a nice modeling environment, metaheuristics are often implemented by hand

Summary

- Linear Programming technologies are highly advanced and there is rich software and modelling support
- As a result, they are often used in many industrial scenarios
- Formulating a problem as Linear Programming task can often be easy, but in other situations it may also be hard and lead to many constraints
- While GAMS provides a nice modeling environment, metaheuristics are often implemented by hand
- Metaheuristics can easily be developed to address a much wider range of problems, although tools like GAMS also support quadratic and nonlinear-programming methods

Summary

- Linear Programming technologies are highly advanced and there is rich software and modelling support
- As a result, they are often used in many industrial scenarios
- Formulating a problem as Linear Programming task can often be easy, but in other situations it may also be hard and lead to many constraints
- While GAMS provides a nice modeling environment, metaheuristics are often implemented by hand
- Metaheuristics can easily be developed to address a much wider range of problems, although tools like GAMS also support quadratic and nonlinear-programming methods
- Sometimes, constraints or objectives which are *not* linear are represented as linear functions so that Linear Programming can be used.

Summary

- Linear Programming technologies are highly advanced and there is rich software and modelling support
- As a result, they are often used in many industrial scenarios
- Formulating a problem as Linear Programming task can often be easy, but in other situations it may also be hard and lead to many constraints
- While GAMS provides a nice modeling environment, metaheuristics are often implemented by hand
- Metaheuristics can easily be developed to address a much wider range of problems, although tools like GAMS also support quadratic and nonlinear-programming methods
- Sometimes, constraints or objectives which are *not* linear are represented as linear functions so that Linear Programming can be used. The optimal solutions for such mis-represented problems are not necessarily optimal solutions of the actual optimization problem...
- It is important to choose the right tool for the right task.

Metaheuristics vs. Linear Programming

Method	Advantages	Disadvantages
Linear Programming		
Metaheuristics		

Metaheuristics vs. Linear Programming

Method	Advantages	Disadvantages
Linear Programming	• industry standard: well-established technology that everyone in operations research knows • will provide the optimal solution for the specified problem • fast, mature • widely understood and used quasi-standard software such as CPLEX • ideal for problems where objective and constraints are linear • nice modelling languages and environments such as GAMS	
Metaheuristics		

Metaheuristics vs. Linear Programming

Method	Advantages	Disadvantages
Linear Programming	• industry standard: well-established technology that everyone in operations research knows • will provide the optimal solution for the specified problem • fast, mature • widely understood and used quasi-standard software such as CPLEX • ideal for problems where objective and constraints are linear • nice modelling languages and environments such as GAMS	• sometimes slow if problem involves many constraints • optimal only if problem is *really* a linear programming problem: sometimes, problems need to be simplified to be linear programming tasks, but the optimal solution of a simplified problem may not be the optimal solution of the real problem • translation of problem definition to LP not always trivial • *not ideal* for problems where objective and constraints are *not linear* • constraints in Linear Programming can be part of the representation in metaheuristics
Metaheuristics		

Metaheuristics vs. Linear Programming

Method	Advantages	Disadvantages
Linear Programming	• industry standard: well-established technology that everyone in operations research knows • will provide the optimal solution for the specified problem • fast, mature • widely understood and used quasi-standard software such as CPLEX • ideal for problems where objective and constraints are linear • nice modelling languages and environments such as GAMS	• sometimes slow if problem involves many constraints • optimal only if problem is *really* a linear programming problem: sometimes, problems need to be simplified to be linear programming tasks, but the optimal solution of a simplified problem may not be the optimal solution of the real problem • translation of problem definition to LP not always trivial • *not ideal* for problems where objective and constraints are *not linear* • constraints in Linear Programming can be part of the representation in metaheuristics
Metaheuristics	• applicable to virtually arbitrary problem definitions • prototypes can be developed quickly from problem definition • best approach for many classical problems (Max-SAT, TSP, ...) • can deal with multiple objectives • can deal with uncertainties and dynamic problems • can deal continuous/numerical problem with real-valued decision variables over convex objective functions • can solve problems efficiently where Linear Programming does not work well • we can quickly built prototypes which deliver acceptable solution quality • constraints are often natural part of the problem representation (e.g., for TSPs)	

Metaheuristics vs. Linear Programming

Method	Advantages	Disadvantages
Linear Programming	• industry standard: well-established technology that everyone in operations research knows • will provide the optimal solution for the specified problem • fast, mature • widely understood and used quasi-standard software such as CPLEX • ideal for problems where objective and constraints are linear • nice modelling languages and environments such as GAMS	• sometimes slow if problem involves many constraints • optimal only if problem is *really* a linear programming problem: sometimes, problems need to be simplified to be linear programming tasks, but the optimal solution of a simplified problem may not be the optimal solution of the real problem • translation of problem definition to LP not always trivial • *not ideal* for problems where objective and constraints are *not linear* • constraints in Linear Programming can be part of the representation in metaheuristics
Metaheuristics	• applicable to virtually arbitrary problem definitions • prototypes can be developed quickly from problem definition • best approach for many classical problems (Max-SAT, TSP, ...) • can deal with multiple objectives • can deal with uncertainties and dynamic problems • can deal continuous/numerical problem with real-valued decision variables over convex objective functions • can solve problems efficiently where Linear Programming does not work well • we can quickly built prototypes which deliver acceptable solution quality • constraints are often natural part of the problem representation (e.g., for TSPs)	• less well-known in the OR community, less respected • usually no guarantee for optimal results • vast majority of available approaches: sometimes hard to decide which one to apply (suggestion: use local search first, then develop Memetic Algorithm, this will usually be the right choice) • much research work rather dodgy • for problems where objective and constraint are linear, we should use Linear Programming • few good implementations, more "do it yourself"

谢谢

Thank you

Thomas Weise [汤卫思]
tweise@hfuu.edu.cn
http://iao.hfuu.edu.cn

Hefei University, South Campus 2
Institute of Applied Optimization
Shushan District, Hefei, Anhui,
China

Caspar David Friedrich, “Der Wanderer über dem Nebelmeer”, 1818
http://en.wikipedia.org/wiki/Wanderer_above_the_Sea_of_Fog

Bibliography

Bibliography I

1. Richard E. Rosenthal. A gams tutorial. URL `http://www.gams.com/latest/docs/UG_Tutorial.html`. accessed 2017.
2. George Bernard Dantzig. Linear programming and extensions. 1963.