Modeling and Optimization with OPL 3 Methods of binary programming

Andreas Popp



These slides are licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.

3 Methods of binary programming

> CC-BY-SA A. Popp

3.1 Modeling of logical expressions

dependent constraints

The Big-M-Method

OPL: modeling of time periods

Disjunctive Constraints

3.3 OPL: Comp

2.4 Diocowic

functions

Step functions

Piecewise linear functions

Inhalt

3.1 Modeling of logical expressions

3.2 Decision dependent constraints

The Big-M-Method

OPL: modeling of time periods

Disjunctive Constraints

3.3 OPL: Compact implementation

3.4 Piecewise functions

Step functions

Piecewise linear functions

OPL: the piecewise command

3 Methods of binary programming

> CC-BY-SA A. Popp

3.1 Modeling of logical expressions

dependent constraints

The Big-M-Method OPL: modeling of time periods

.3 OPL: Comp

3.4 Piecewise

functions

Step functi

Piecewise linear fu

TL: the piecewise mmand

3.1 Modeling of logical expressions

dependent constraints

The Big-M-Method
OPL: modeling of time periods

3.3 OPL: Comp

implementation

3.4 Piecew functions

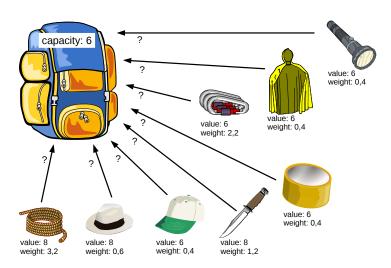
Sten functions

Piecewise linear funct

Piecewise linear functions
OPL: the piecewise

3.1 Modeling of logical expressions

Example: Adventure Inc.



3 Methods of binary programming

CC-BY-SA A. Popp

3.1 Modeling of logical expressions

3.2 Decision dependent constraints

The Big-M-Method

OPL: modeling of time eriods

3.3 OPL: Compact mplementation

3.4 Piecev functions

Step function

iecewise linear functions
PL: the piecewise

Set of items

Parameters:

weight of item $i \in I$

value of item $i \in I$

capactiy of the knapsack

Decision variables:

binary decision variable; indicates if item $i \in I$ is packed Xi

Model description:

$$\max \sum_{i \in I} u_i \cdot x_i$$

$$s.t. \sum_{i \in I} w_i \cdot x_i \le c$$

$$x_i \in \{0, 1\} \qquad \forall i \in I$$

3 Methods of binary programming

> CC-BY-SA A. Popp

3.1 Modeling of logical expressions

Logical operators

- ¬ logical negation
- ∧ logical and
- ∨ logical or
- ⇒ logical implication
- ⇔ logical equivalence

Truth table in numerical representation

Α	В	$\neg A$	$\neg B$	$A \wedge B$	$A \lor B$	$A \stackrel{\vee}{_} B$	$A \Rightarrow B$	$A \Leftrightarrow B$
1	1	0	0	1	1	0	1	1
1	0	0	1	0	1	1	0	0
0	1	1	0	0	1	1	1	0
0	0	1	1	0	0	0	1	1

3 Methods of binary programming

> CC-BY-SA A. Popp

3.1 Modeling of logical expressions

3.2 Decision dependent constraints

The Big-M-Method OPL: modeling of time periods

3 OPL: Compa

implementation

3.4 Piecew functions

Step functions Piecewise linear

Piecewise linear funct OPL: the piecewise

Logical operators in binary optimization models

3 Methods of binary programming

> CC-BY-SA A. Popp

3.1 Modeling of logical expressions

Example: Let x_1 and x_2 be binary decision variables of a knapsack problem, representing items l_1 and l_2 .

 $\neg l_1$: Get the value of l_1 not being packed.

▶
$$1 - x_1$$

 $I_1 \wedge I_2$: Both I_1 and I_2 must be packed.

$$x_1 + x_2 = 2$$

 $l_1 \vee l_2$: At least one of the items has to be packed.

$$x_1 + x_2 > 1$$

 $\neg (I_1 \land I_2)$: At most one of the items may be packed.

►
$$x_1 + x_2 \le 1$$

Logical operators in binary optimization models

3 Methods of binary programming

CC-BY-SA A. Popp

3.1 Modeling of logical expressions

logical expressions
3.2 Decision

ependent onstraints

The Big-M-Method

OPL: modeling of time periods

Disjunctive Constraints

.3 OPL: Compa

3.4 Piecewi functions

Step functions

Piecewise linear fund

OPL: the piecewise ommand

Example: Let x_1 and x_2 be binary decision variables of a knapsack problem, representing items l_1 and l_2 .

 $\neg (I_1 \lor I_2)$: None of the items may be packed.

$$x_1 + x_2 = 0$$

 $l_1 \vee l_2$: Exactly one of the items must be packed.

$$x_1 + x_2 = 1$$

 $I_1 \Rightarrow I_2$: If I_1 is packed, I_2 must also be packed.

▶
$$x_1 \le x_2$$

 $l_1 \Leftrightarrow l_2$: The decision is identical for both items.

3.2 Decision dependent constraints

OPL: modeling of time periods

3.3 OPL: Comp

implementation

3.4 Piecewi

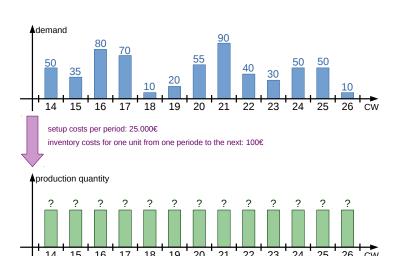
unctions

iecewise linear functions

PL: the piecewise

3.2 Decision dependent constraints

Example: Lewig Wakuxi



3 Methods of binary programming

CC-BY-SA A. Popp

3.1 Modeling of logical expressions

3.2 Decision dependent constraints

The Big-M-Method
OPL: modeling of time periods

Disjunctive Constraints

implementation

3.4 Piecew functions

Step functions

Piecewise linear fu

OPL: the piecewise

Parameters:

 d_t demand in period $t \in T$ s_t setup costs in period $t \in T$

 h_t inventory costs per item in period $t \in T$

 $i_{t_{min}-1}$ initial inventory

M a big number

Decision variables:

 x_t production quantity in period $t \in T$ i_t inventory at the end of period $t \in T$ y_t production decision in period $t \in T$

Model description:

$$\min \quad \sum_{t \in T} s_t \cdot y_t + h_t \cdot i_t$$

s.t.
$$i_t = i_{t-1} + x_t - d_t$$
 $\forall t \in T$ (I)
 $x_t \leq \mathbf{M} \cdot y_t$ $\forall t \in \mathbf{T}$ (II)
 $x_t, i_t > 0; y_t \in \{0, 1\}$ $\forall t \in T$

3 Methods of binary programming

> CC-BY-SA A. Popp

3.1 Modeling of logical expressions

3.2 Decision dependent constraints

The Big-M-Method
OPL: modeling of time periods

3.3 OPL: Compa

3.4 Piecew

unctions

Step functions

Piecewise linear functions
OPL: the piecewise
command

The Big-M-Method

Let $\overline{\mathbf{x}}$ be the vector of decision variables and f be a linear function. The constraint

$$f(\overline{\mathbf{x}}) \leq b$$
 resp. $f(\overline{\mathbf{x}}) \geq b$

shall only be constraining if a decision represented by the binary variable y assuming the value 0 has been made.

Decision dependent constraint

Let M be a sufficiently big number.

$$f(\overline{\mathbf{x}}) \leq b \quad \rightarrow \quad f(\overline{\mathbf{x}}) \leq b + M \cdot y$$

$$f(\overline{\mathbf{x}}) \geq b \quad \rightarrow \quad f(\overline{\mathbf{x}}) \geq b - M \cdot y$$

3 Methods of binary programming

> CC-BY-SA A. Popp

The Big-M-Method

Problem with implementation of time periods

Constraint of example "Lewig Wakuxi"

$$i_t = i_{t-1} + x_t - d_t \quad \forall t \in T \quad (I)$$

Implementation attempt 1

```
{string} T = {"KW14", "KW15", "KW16", "KW17"};
dvar float+ i[T];
  forall(t in T) i[t] == i[t-1] + x[t] - d[t];
```

Operator for string - int not available.

3 Methods of binary programming

> CC-BY-SA A. Popp

- 3.1 Modeling of logical expressions
- dependent constraints
- The Big-M-Method
 OPL: modeling of time

periods

Disjunctive Constraints

implementation

3.4 Piecewi functions

functions

Piecewise linear function

OPL: the piecewise

Problem with implementation of time periods

Constraint of example "Lewig Wakuxi"

$$i_t = i_{t-1} + x_t - d_t \quad \forall t \in T \quad (I)$$

Implementation attempt 2

```
{int} T = {14, 15, 16, 17};
dvar float+ i[T];
  forall(t in T) i[t] == i[t-1] + x[t] - d[t];
```

Ondex out of bound for array "i": 13.

3 Methods of binary programming

> CC-BY-SA A. Popp

- 3.1 Modeling of logical expressions
 - dependent constraints
 - The Big-M-Method
 - OPL: modeling of time periods

Disjunctive Constraints

- 3.3 OPL: Compacimplementation
- 3.4 Piecewi

functions

Piecewise linear functio

Constraint of example "Lewig Wakuxi"

$$i_t = i_{t-1} + x_t - d_t \quad \forall t \in T \quad (I)$$

Implementation attempt 3

```
{int} T = \{14, 15, 16, 17\};
\{int\}\ T0 = \{13, 14, 15, 16, 17\};
dvar float+ i[T0]:
  forall(t in T) i[t] == i[t-1] + x[t] - d[t]:
```

3 Methods of binary programming

> CC-BY-SA A. Popp

- OPL: modeling of time neriods

Problem with implementation of time periods

Constraint of example "Lewig Wakuxi"

$$i_t = i_{t-1} + x_t - d_t \quad \forall t \in T$$
 (I)

Implementation attempt 4

```
int Tmin = 14:
int Tmax = 17:
range T = Tmin..Tmax;
dvar float+ i[Tmin-1..Tmax];
  forall(t in T) i[t] == i[t-1] + x[t] - d[t];
```

3 Methods of binary programming

> CC-BY-SA A. Popp

- OPL: modeling of time neriods

Disjunctive Constraints I

A model shall have the following constraints:

$$f(\overline{\mathbf{x}}) \leq b$$

 $g(\overline{\mathbf{x}}) < d$

It is enough to only fullfil one constraint.

Disjunctive Constraints

Let M be a sufficiently large number and y be a binary auxiliary variable.

$$f(\overline{\mathbf{x}}) \le b + M \cdot y$$

 $g(\overline{\mathbf{x}}) \le d + M \cdot (1 - y)$

≥-constraint analog

3 Methods of binary programming

> CC-BY-SA A. Popp

3.1 Modeling of logical expressions

3.2 Decision dependent constraints

The Big-M-Method

OPL: modeling of time
periods

Disjunctive Constraints

3.3 OPL: Compact implementation

3.4 Piecew functions

Step functions

Piecewise linear funct OPL: the piecewise

Disjunctive Constraints II

A model shall have to following constraint:

$$g(\overline{\mathbf{x}}) \leq d$$

This constraint only needs to be fullfiled if it holds:

$$f(\overline{\mathbf{x}}) > b$$

Disjunctive Constraints

Let M be a sufficiently large number and y be a binary auxiliary variable.

$$f(\overline{\mathbf{x}}) \le b + M \cdot y$$

 $g(\overline{\mathbf{x}}) \le d + M \cdot (1 - y)$

 \geq -constraint analog

3 Methods of binary programming

CC-BY-SA A. Popp

3.1 Modeling of logical expressions

3.2 Decision dependent

The Big-M-Method OPL: modeling of tim periods

Disjunctive Constraints

3.3 OPL: Compac mplementation

3.4 Piecev functions

Step functions

Piecewise linear fu

OPL: the piecewi

3.3 OPL: Compact implementation

3 Methods of binary programming

> CC-BY-SA A. Popp

3.1 Modeling of logical expressions

dependent constraints

The Big-M-Method

OPL: modeling of time periods

Disjunctive Constraints

3.3 OPL: Compact implementation

3.4 Piecewise

unctions

Step functions

liecewise linear functions

Decision expressions

Objective function of the Wagner-Whitin-problem:

```
// objective function
minimize sum(t in T)(s[t]*y[t] + h[t]* i[t]);
```

Structuring with decision expressions:

```
// decision expressions
dexpr float setupCost = sum(t in T)(s[t]*y[t]);
dexpr float inventoryCost = sum(t in T)(h[t]*i[t]);
// objective function
minimize setupCost + inventoryCost;
```

3 Methods of binary programming

CC-BY-SA A. Popp

- 3.1 Modeling of logical expressions
- dependent constraints

The Big-M-Method

OPL: modeling of time periods

Disjunctive Constraints

3.3 OPL: Compact implementation

3.4 Piecewise functions

Step functions

Piecewise linear fur

Piecewise linear function
OPL: the piecewise

Arrays of decision expressions

Objective function of the Wagner-Whitin-problem:

```
// objective function
minimize sum(t in T)(s[t]*y[t] + h[t]* i[t]);
```

Structuring with decision expressions:

```
//decision expressions
dexpr float periodCost[t in T]
 = s[t]*y[t] + h[t]*i[t];
// objective function
minimize sum (t in T)(periodCost[t]);
```

3 Methods of binary programming

CC-BY-SA A. Popp

3.3 OPL: Compact implementation

CC-BY-SA A. Popp

- 3.1 Modeling of logical expressions
- 3.2 Decision dependent constraints

The Big-M-Method

OPL: modeling of time periods

Disjunctive Constraints

3.3 OPL: Comp

3.4 Piecewise

functions

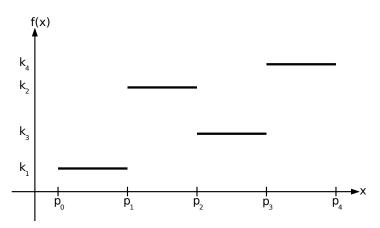
Piecewise linear functions

OPL: the piecewise

3.4 Piecewise functions

Step functions

Let x be a continous decision variable:



3 Methods of binary programming

> CC-BY-SA A. Popp

- 3.1 Modeling of logical expressions
 - 3.2 Decision dependent constraints
 - The Big-M-Method OPL: modeling of time periods
 - periods Disjunctive Constraints
 - mplementation
 - 3.4 Piecew functions

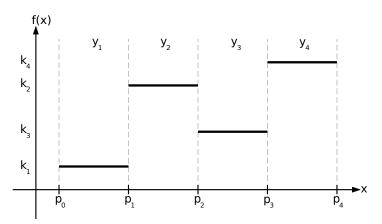
Step functions

Piecewise linear functions

OPL: the piecewise

Step functions

Let x be a continous decision variable:



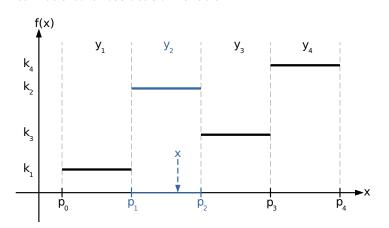
3 Methods of binary programming

> CC-BY-SA A. Popp

- 3.1 Modeling of logical expressions
- 3.2 Decision dependent constraints
- The Big-M-Method OPL: modeling of time periods
- Disjunctive Constraints
- mplementation
- functions

Step functions

Piecewise linear functions



• e.g.: $x = \frac{1}{3} \cdot p_1 + \frac{2}{3} \cdot p_2$

3 Methods of binary programming

> CC-BY-SA A. Popp

3.1 Modeling of logical expressions

3.2 Decision dependent constraints

The Big-M-Method

OPL: modeling of time periods

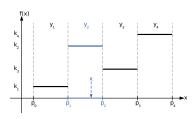
3.3 OPL: Compaimplementation

3.4 Piecew functions

Step functions

Piecewise linear functions

Decision variable as convex combination of the supporting points



$$x = \sum_{n=0}^{N} z_n \cdot p_n$$

$$\sum_{n=0}^{N} z_n = 1$$

$$0 \le z_n \le 1 \quad \forall n \in \{1, \dots, N\}$$

3 Methods of binary programming

> CC-BY-SA A. Popp

- 3.1 Modeling of logical expressions
- 3.2 Decision dependent constraints
- The Big-M-Method
 OPL: modeling of time periods
 Disjunctive Constraints
- 3.3 OPL: Compact implementation
- 3.4 Piecewise functions

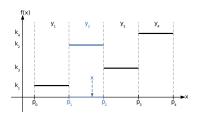
Step functions

Piecewise linear functions

OPL: the piecewise

command

Choice of the correct interval



$$\sum_{n=1}^{N} y_n = 1$$

$$z_0 \le y_1$$

$$z_n \le y_n + y_{n+1} \quad \forall n \in \{1, \dots, N-1\}$$

$$z_N \le y_N$$

3 Methods of binary programming

> CC-BY-SA A. Popp

- 3.1 Modeling of logical expressions
- 3.2 Decision dependent constraints
 - OPL: modeling of time periods Disjunctive Constraints
- implementation
- functions

Step functions

Piecewise linear functions

OPL: the piecewise

command

Complete modeling

 $z_N < y_N$

$$f(x) = \sum_{n=1}^{N} y_n \cdot k_n$$

$$x = \sum_{n=0}^{N} z_n \cdot p_n$$

$$\sum_{n=0}^{N} z_n = 1$$

$$0 \le z_n \le 1 \quad \forall n \in \{1, \dots, N\}$$

$$\sum_{n=1}^{N} y_n = 1$$

$$z_0 \le y_1$$

 $z_n \leq y_n + y_{n+1} \quad \forall n \in \{1, \dots, N-1\}$

3 Methods of binary programming

CC-BY-SA A. Popp

- 3.1 Modeling of logical expressions
- 3.2 Decision dependent constraints
 - The Big-M-Method
 OPL: modeling of time
 periods
 Disjunctive Constraints
- ninpiementation
- functions
 Step functions

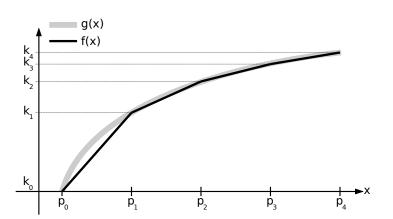
Step runc

OPL: the piecewise command

У,

 y_4

Piecewise linear functions

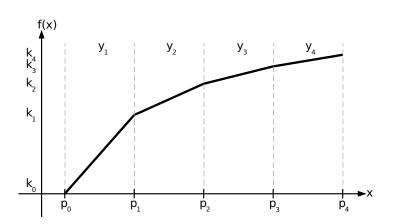


3 Methods of binary programming

CC-BY-SA A. Popp

Piecewise linear functions

Piecewise linear functions

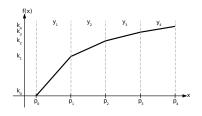


3 Methods of binary programming

CC-BY-SA A. Popp

Piecewise linear functions

Function values as convex combination



$$x = \sum_{n=0}^{N} z_n \cdot p_n$$

$$f(x) = \sum_{n=0}^{N} z_n \cdot f(p_n)$$

$$\sum_{n=0}^{\infty} z_n = 1$$

$$0 \le z_n \le 1 \quad \forall n \in \{1, \dots, N\}$$

3 Methods of binary programming

> CC-BY-SA A. Popp

- 3.1 Modeling of logical expressions
 - 3.2 Decision dependent constraints

OPL: modeling of time periods
Disjunctive Constraints

- 3.3 OPL: Compace mplementation
- 3.4 Piecewi functions

Stop functions

Piecewise linear functions

OPL: the piecewise

Complete modeling

 $z_0 \leq y_1$

 $z_N < y_N$

$$f(x) = \sum_{n=0}^{N} z_n \cdot k_n$$

$$x = \sum_{n=0}^{N} z_n \cdot p_n$$

$$\sum_{n=0}^{N} z_n = 1$$

$$0 \le z_n \le 1 \quad \forall n \in \{1, \dots, N\}$$

$$\sum_{n=1}^{N} y_n = 1$$

 $z_n \leq y_n + y_{n+1} \quad \forall n \in \{1, \dots, N-1\}$

3 Methods of binary programming

CC-BY-SA A. Popp

- 3.1 Modeling of logical expressions
 - 3.2 Decision dependent constraints

The Big-M-Method
OPL: modeling of time periods
Disjunctive Constraints

3.3 OPL: Compa

3.4 Piecewise

functions

Piecewise linear functions

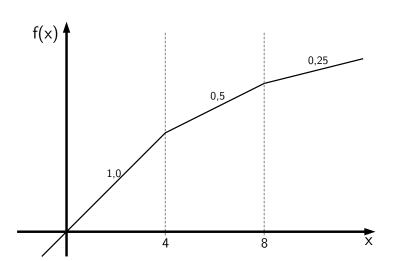
ODL --

OPL: the piecewise command

У,

y₃

Piecewise linear functions by slope



3 Methods of binary programming

CC-BY-SA A. Popp

3.1 Modeling of logical expressions

3.2 Decision dependent constraints

The Big-M-Method

OPL: modeling of time periods

Disjunctive Constraints

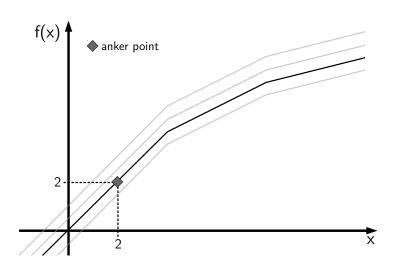
3.3 OPL: Compact mplementation

3.4 Piecew functions

Step functions

Piecewise linear function

Ankering of piecewise linear functions



3 Methods of binary programming

CC-BY-SA A. Popp

3.1 Modeling of logical expressions

3.2 Decision dependent constraints

The Big-M-Method OPL: modeling of time periods

sjunctive Constraints

3.3 OPL: Compac implementation

3.4 Piecev functions

Step function

Piecewise linear funct

Syntax of the piecewise command

Array p of supporting points and array s of slopes:

```
piecewise(i in 1..N){
  s[i] -> p[i];
  s[N+1]
} (anker point) x;
```

Example of figure above

```
int N = 2;
float p[1..N] = [4, 8];
float s[1..N+1] = [1.0, 0.5, 0.25];
dvar float+ x;
piecewise(i in 1..N){
   s[i] -> p[i];
   s[N+1]
} (2, 2) x;
```

3 Methods of binary programming

> CC-BY-SA A. Popp

3.1 Modeling of logical expressions

dependent constraints

The Big-M-Method
OPL: modeling of time periods

3.3 OPL: Compac

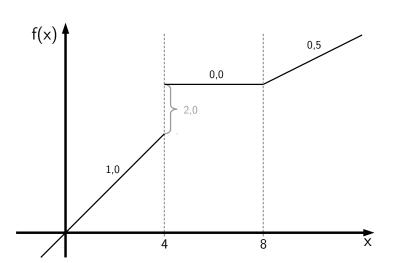
mplementation

3.4 Piecew functions

Step functions

Piecewise linear funct

Step functions and general discontinuities



3 Methods of binary programming

CC-BY-SA A. Popp

3.1 Modeling of logical expressions

3.2 Decision dependent constraints

The Big-M-Method OPL: modeling of time periods Disjunctive Constraints

3.3 OPL: Compact implementation

3.4 Piecewis functions

Piecewise linear function

Step functions and general discontinuities

Second slope value at the same supporting point in the piecewise command becomes step value.

Example of figure above

```
int N = 3:
float p[1..N] = [4, 4, 8];
float s[1..N+1] = [1.0, 2.0, 0.0, 0.5];
dvar float+ x:
piecewise(i in 1..N){
  s[i] \rightarrow p[i];
  s[N+1]
} x:
```

3 Methods of binary programming

> CC-BY-SA A. Popp