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

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

R 4 Piecewise

functions

Step function

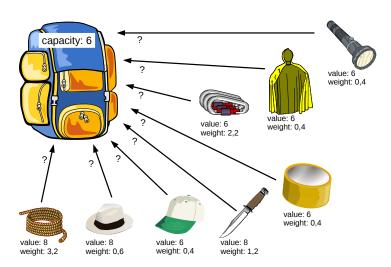
Piecewise linear fu

mmand

3.1 Modeling of logical expressions

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: Compac implementation

3.4 Piecewi functions

Step function

Piecewise linear functi OPL: the piecewise

Index sets:

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 exklusive or ("xor")
- ⇒ logical implication
- ⇔ logical equivalence

Truth table in numerical representation

_	Α	В	$\neg A$	$\neg B$	$A \wedge B$	$A \vee B$	$A \vee B$	$A \Rightarrow B$	$A \Leftrightarrow B$
	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

constraints

OPL: modeling of time eriods

3.3 OPL: Compa

nplementation

functions

Sten functions

Piecewise linear fu

Piecewise linear function
OPL: the piecewise

Logical operators in binary optimization models

3 Methods of binary programming

CC-BY-SA A. Popp

Example: Let x_1 and x_2 be binary decision variables of a knapsack problem, representing items I_1 and I_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.

►
$$x_1 + x_2 \ge 1$$

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

►
$$x_1 + x_2 \le 1$$

3.1 Modeling of logical expressions

3.2 Decision dependent

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

.3 OPL: Compac

3.4 Piecew functions

Step functions

Piecewise linear function

ommand

Logical operators in binary optimization models

3 Methods of binary programming

> CC-BY-SA A. Popp

3.1 Modeling of logical expressions

logical expressions

2 Decision ependent

The Big-M-Method

OPL: modeling of time periods

Disjunctive Constraints

.3 OPL: Compa

3.4 Piecewi

Step functions

Piecewise linear fu

PL: the piecewise

Example: Let x_1 and x_2 be binary decision variables of a knapsack problem, representing items I_1 and I_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.

$$x_1 + x_2 = 1$$

 $l_1 \Rightarrow l_2$: If l_1 is packed, l_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 tim periods

3.3 OPL: Comp

implementation

3.4 Piecew

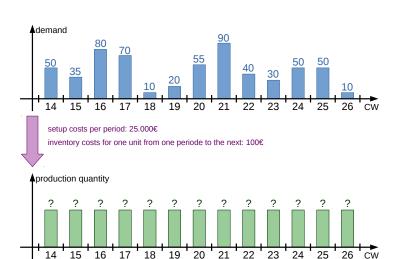
inctions

Piecewise linear function

PPL: 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 expression:

3.2 Decision dependent constraints

The Big-M-Method

OPL: modeling of time

Disjunctive Constraints

mplementation

3.4 Pieces

Step functions

Piecewise linear f

OPL: the piecewise command

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:

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

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

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: Compac

3.4 Piecev

unctions

iecewise linear functions

The Big-M-Method

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

$$f(\overline{\mathbf{x}}) \le b$$
 resp. $f(\overline{\mathbf{x}}) \ge b$

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

Constraint of example "Lewig Wakuxi"

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

Implementation attempt 1

```
\{\text{string}\}\ T = \{\text{"KW14"}, \text{"KW15"}, \text{"KW16"}, \text{"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

- OPL: modeling of time neriods

3 Methods of binary programming

CC-BY-SA

A. Popp

Constraint of example "Lewig Wakuxi"

$$i_t = i_{t-1} + x_t - d_t \qquad \forall t \in T$$
 (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];
Olimber Index out of bound for array "i": 13.
```

OPL: modeling of time neriods

Problem with implementation of time periods

Constraint of example "Lewig Wakuxi"

$$i_t = i_{t-1} + x_t - d_t \qquad \forall t \in \mathcal{T}$$
 (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

- 3.1 Modeling of logical expressions
- dependent constraints

The Big-M-Method

OPL: modeling of time periods

Disjunctive Constraints

- implementation
- 3.4 Piecew functions

Tunctions

Piecewise linear functions

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

- 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

Piecewise linear functions

Piecewise linear functions OPL: the piecewise

Disjunctive Constraints I

A model shall have the following constraints:

$$f(\overline{\mathbf{x}}) \le b$$
$$g(\overline{\mathbf{x}}) \le d$$

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 Decisio dependent constraints

The Big-M-Method OPL: modeling of tir

Disjunctive Constraints

3.3 OPL: Compac

- 3.4 Piecewis
 - unctions Stop functions

Piecewise linear f

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.

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

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

≥-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 fund OPL: the piecewise

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

R 4 Piecewise

unctions

Step functions

Piecewise linear functions

PL: the piecewise

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 func

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

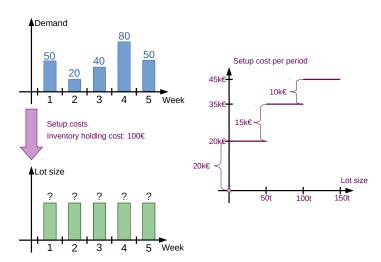
functions

Piecewise linear functions

OPL: the piecewise

3.4 Piecewise functions

Example: Lewig Xanxi



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

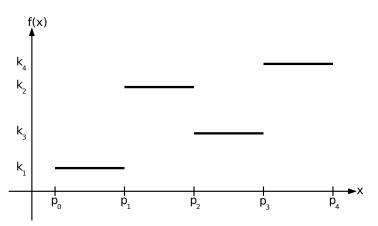
 OPL: modeling of time periods
- 3.3 OPL: Compa
- 3.4 Piecew functions

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

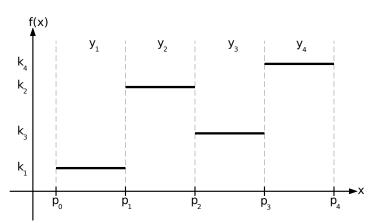
- 3.1 Modeling of logical expressions
 - 3.2 Decision dependent constraints
 - The Big-M-Method OPL: modeling of time periods
 - Disjunctive Constraints
 - mplementation
 - functions

Step functions

Piecewise linear functions
OPL: the piecewise

Step functions

Let x be a continous decision variable:



3 Methods of binary programming

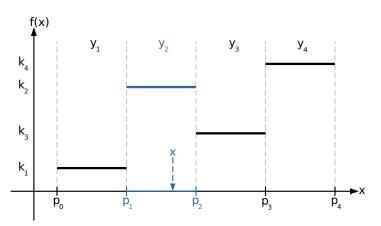
> CC-BY-SA A. Popp

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

Step functions

Piecewise linear functions

OPL: the piecewise



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: Compac implementation

3.4 Piecewi functions

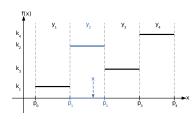
Step functions

Piecewise linear functions

L: the piecewise mmand

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

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
- functions

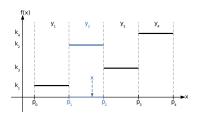
Step functions

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

Step runc

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
 - OPL: modeling of time periods
 Disjunctive Constraints
 - implementation
- functions
 Step functions

Diep rune

OPL: the piecewise command

У,

 y_4

Example: Lewig Tadelbach

Assume a production problem.

Capacities

	Machine A	Machine B	Machine C
I_1	2,2	1,6	2,8
I_2	1,2	1,9	2,3
	72	48	60

Sales revenue

$$f_1(x_1) = 2000 \cdot \sqrt{x_1}$$

 $f_2(x_2) = 1800 \cdot \sqrt{x_2}$

3 Methods of binary programming

> CC-BY-SA A. Popp

3.1 Modeling of logical expressions

3.2 Decision dependent constraints

he Big-M-Method
PL: modeling of time
eriods
isjunctive Constraints

3.3 OPL: Compa

3.4 Piecew

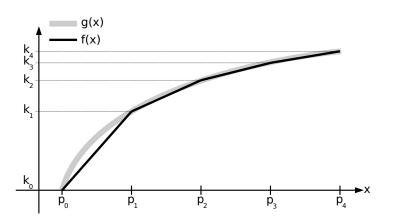
inctions

.

Piecewise linear functions

OPL: the piecewise

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 DPL: modeling of time periods

3.3 OPL: Comp

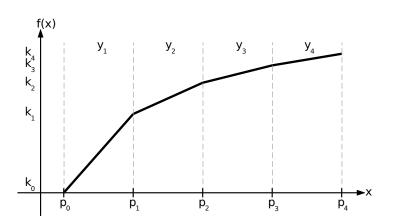
mpiementatio

unctions

Piecewise linear functions

Piecewise linear functions

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

3.3 OPL: Comp

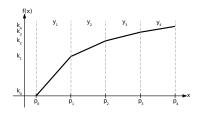
3.4 Piecew functions

... 6......

Piecewise linear functions

OPL: the piecewise

Function values as convex combination



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

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

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

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

3 Methods of binary programming

CC-BY-SA A. Popp

- 3.1 Modeling of logical expressions
 - 3.2 Decision dependent constraints

OPL: modeling of time periods
Disjunctive Constraints

3.3 OPL: Compa

3.4 Piecewise

functions

Step functio

Piecewise linear functions

OPL: the piecewise command

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
 - dependent constraints
 - The Big-M-Method

 OPL: modeling of time periods

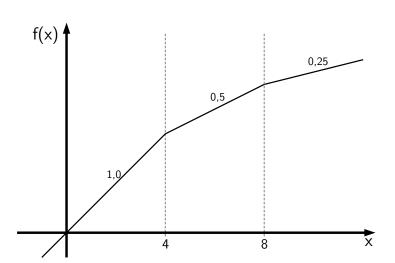
 Disjunctive Constraints
 - 3.3 OPL: Com
 - 3.4 Piecewise

Step function

Piecewise linear functions

OPL: the piecewise

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

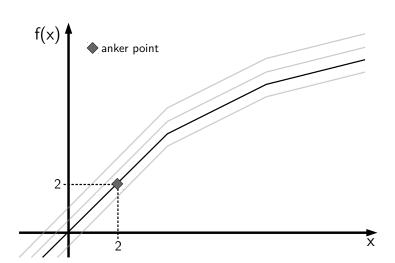
3.4 Piecewise functions

Step functions

Piecewise linear functio

OPL: the piecewise command

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

mplementation

3.4 Piecev functions

Step function

Piecewise linear function

OPL: the piecewise command

Array p of supporting points and array s of slopes:

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

Example of figure above

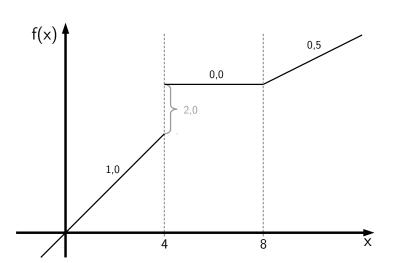
```
int N = 2:
float p[1..N] = [4, 8];
float s[1..N+1] = [1.0, 0.5, 0.25];
dvar float+ x:
piecewise(i in 1..N){
  s[i] \rightarrow p[i];
  s[N+1]
} (2, 2) x:
```

3 Methods of binary programming

> CC-BY-SA A. Popp

OPL: the piecewise command

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 Piecewise functions

Step functions

Piecewise linear functi

OPL: the piecewise command

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] -> p[i];
  s[N+1]
} x:
```

3 Methods of binary programming

> CC-BY-SA A. Popp

- 3.1 Modeling of logical expressions
 - ependent onstraints

The Big-M-Method
OPL: modeling of time

eriods
Disjunctive Constraints

- 3.3 OPL: Compa implementation
- 3.4 Piecewi functions

unctions
Step functions

Piecewise linear funct

OPL: the piecewise command