Modeling and Optimization with OPL

3 Methods of binary programming

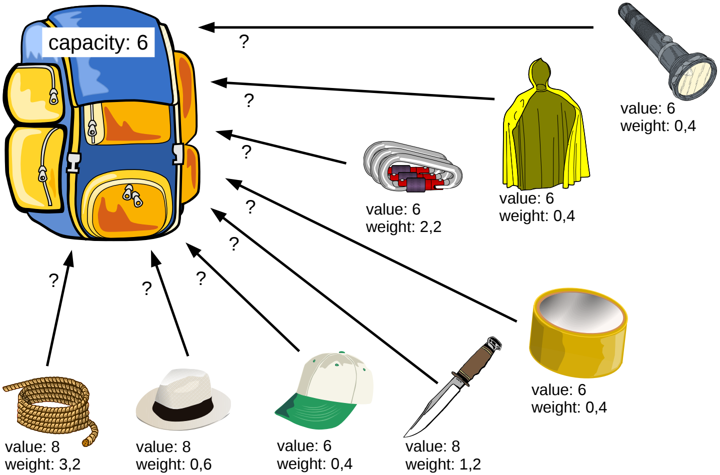
Andreas Popp

Inhalt

A. Popp

3.1 Modeling of logical expressions

Example: Adventure Inc.



Model: Knapsack problem

Index sets:

I Set of items

Parameters:

w_i weight of item $i \in I$

u_i value of item $i \in I$

c capactiy of the knapsack

Decision variables:

x_i binary decision variable; indicates of item $i \in I$ is packed

Model description:

$$\begin{aligned} \max \quad & \sum_{i \in I} u_i \cdot x_i \\ s.t. \quad & \sum_{i \in I} w_i \cdot x_i \le c && \text{(I)} \\ & x_i \in \{0,1\} \qquad \forall i \in I \end{aligned}$$

Logical operators

- $\neg$ logical **negation**
- $\wedge$ logical **and**
- $\vee$ logical **or**
- $\veebar$ logical **exklusive or** ("xor")
- $\Rightarrow$ logical **implication**
- $\Leftrightarrow$ logical **equivalence**

Truth table in numerical representation

A	B	$\neg A$	$\neg B$	$A \wedge B$	$A \vee B$	$A \veebar B$	$A \Rightarrow B$	$A \Leftrightarrow B$
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

A. Popp

Logical operators in binary optimization models

Example: Let x_1 and x_2 be binary decision variables of a knapsack problem, representing items I_1 and I_2.

$\neg I_1$: Get the value of I_1 not being packed.

- $1 - x_1$

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

- $x_1 + x_2 = 2$

$I_1 \vee I_2$: At least one of the items has to be packed.

- $x_1 + x_2 \geq 1$

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

- $x_1 + x_2 \leq 1$

Logical operators in binary optimization models

Example: Let x_1 and x_2 be binary decision variables of a knapsack problem, representing items I_1 and I_2.

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

- $x_1 + x_2 = 0$

$I_1 \veebar I_2$: 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.

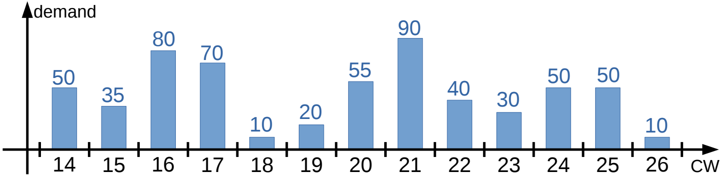
- $x_1 \leq x_2$

$I_1 \Leftrightarrow I_2$: The decision is identical for both items.

- $x_1 = x_2$

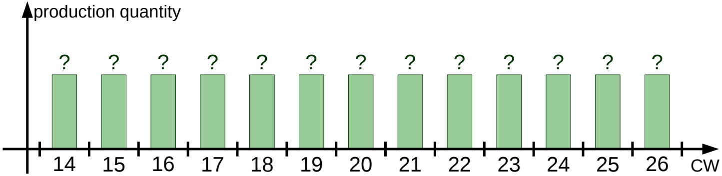
3.2 Decision dependent constraints

Example: Lewig Wakuxi



setup costs per period: 25.000€

inventory costs for one unit from one periode to the next: 100€



Model: Wagner-Whitin-problem

Index sets:

T	planning periods $\{t_{min}, \dots, t_{max}\}$

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:

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

The Big-M-Method

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

$$f(\bar{\mathbf{x}}) \leq b \qquad \text{bzw.} \qquad f(\bar{\mathbf{x}}) \geq b$$

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

Decision dependent constraint

Let M be a sufficiently big number.

$$\begin{array}{lcl} f(\bar{\mathbf{x}}) \leq b & \rightarrow & f(\bar{\mathbf{x}}) \leq b + M \cdot y \\ f(\bar{\mathbf{x}}) \geq b & \rightarrow & f(\bar{\mathbf{x}}) \geq b - M \cdot y \end{array}$$

Problem with implementation of time periods

Constraint of example „Lewig Wakuxi“

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

Implementation try 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.

Problem with implementation of time periods

Constraint of example „Lewig Wakuxi“

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

Implementation try 2

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

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

Problem with implementation of time periods

Constraint of example „Lewig Wakuxi“

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

Implementation try 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];
```

Problem with implementation of time periods

Constraint of example „Lewig Wakuxi“

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

Implementation try 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];
```

Disjunctive Constraints I

A model shall have to following constraints:

$$f(\bar{\mathbf{x}}) \leq b$$
$$g(\bar{\mathbf{x}}) \leq d$$

It is enough to only fullfil one constraint.

Disjunctive Constraints

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

$$f(\bar{\mathbf{x}}) \leq b + M \cdot y$$
$$g(\bar{\mathbf{x}}) \leq d + M \cdot (1 - y)$$

$\geq$-constraint analog

Disjunctive Constraints II

A model shall have to following constraint:

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

This constraint only needs to be fullfiled if it holds:

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

Disjunctive Constraints

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

$$f(\bar{\mathbf{x}}) \leq b + M \cdot y$$
$$g(\bar{\mathbf{x}}) \leq d + M \cdot (1 - y)$$

$\geq$-constraint analog

3.3 OPL: Compact implementation

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;
```

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:

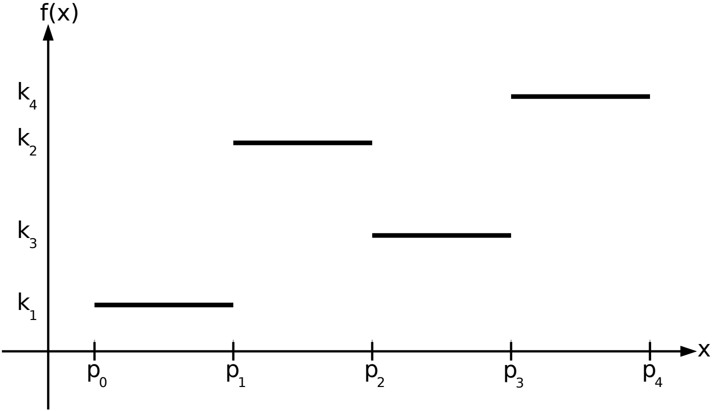
```
//decision expressions
dexpr float periodCost[t in T]
 = s[t]*y[t] + h[t]*i[t];

 // objective function
minimize sum (t in T)(periodCost[t]);
```


A. Popp

3.4 Piecewise functions

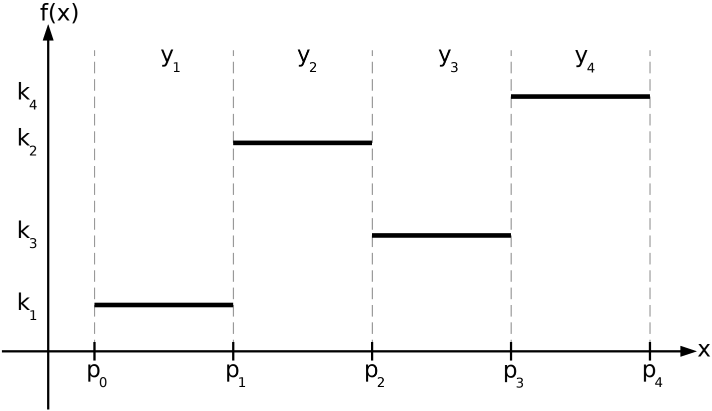
Step functions

Let x be a continous decision variable:



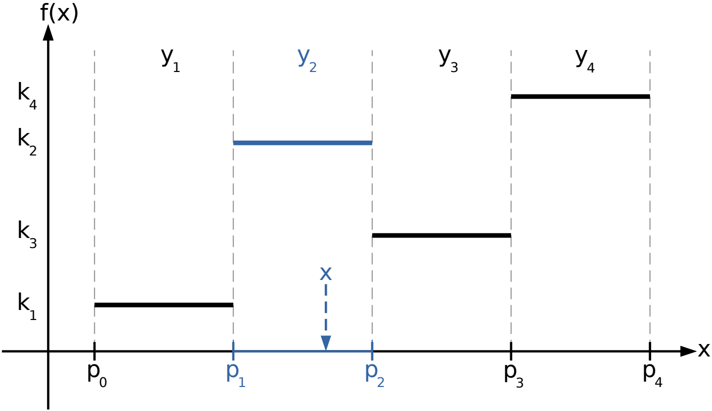
Step functions

Let x be a continous decision variable:



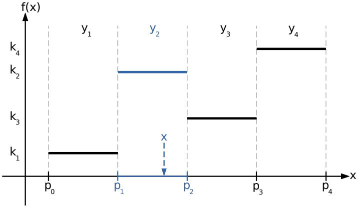
Step functions

Let x be a continous decision variable:



- z.B.: $x = \frac{1}{3} \cdot p_1 + \frac{2}{3} \cdot p_2$

Decision variable as convex combination of the supporting points

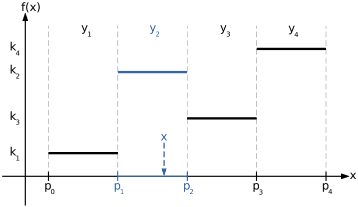


$$x = \sum_{n=0}^{N} z_n \cdot p_n$$

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

$$0 \leq z_n \leq 1 \quad \forall n \in \{1, \ldots, N\}$$

Choice of the correct interval



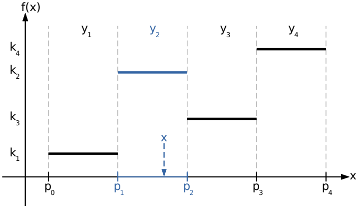
$$\sum_{n=1}^{N} y_n = 1$$

$$z_0 \leq y_1$$

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

$$z_N \leq y_N$$

Complete modeling



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

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

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

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

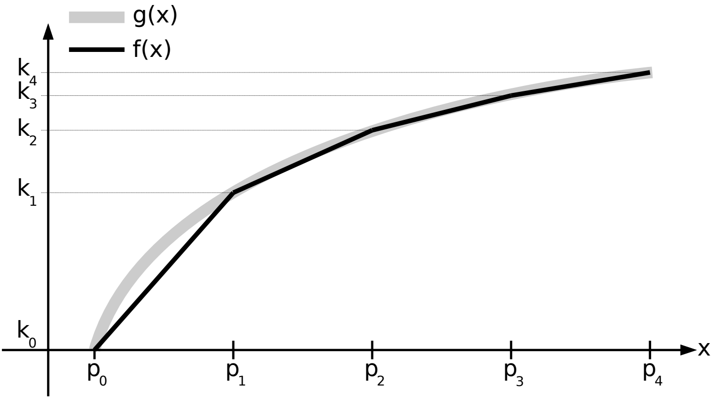
$$\sum_{n=1}^{N} y_n = 1$$

$$z_0 \leq y_1$$

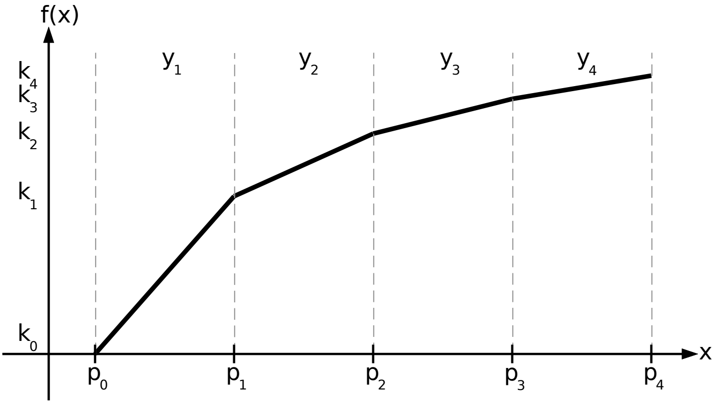
$$z_n \leq y_n + y_{n+1} \quad \forall n \in \{1, \dots, N-1\}$$

$$z_N \leq y_N$$

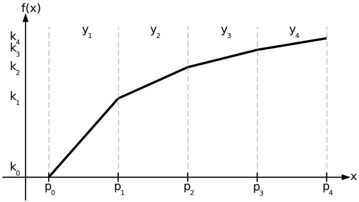
Piecewise linear functions



Piecewise linear functions



Function values as convex combination



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

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

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

$$0 \leq z_n \leq 1 \quad \forall n \in \{1, \ldots, N\}$$

Complete modeling

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

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

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

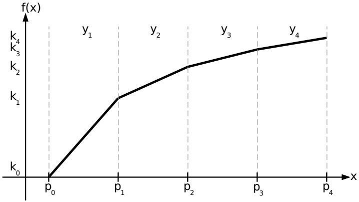
$$0 \leq z_n \leq 1 \quad \forall n \in \{1, \ldots, N\}$$

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

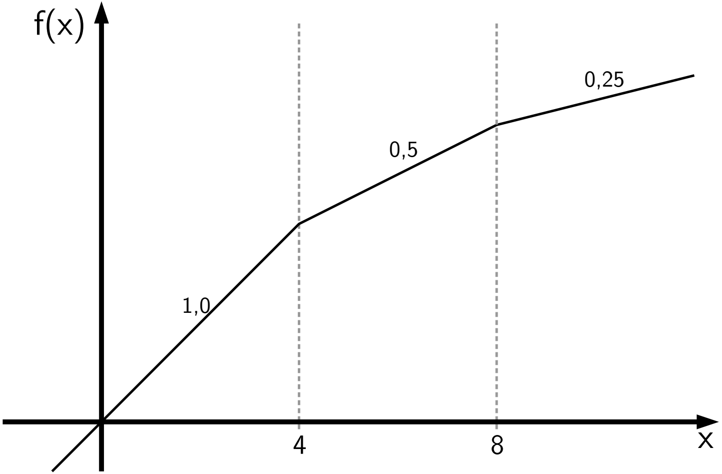
$$z_0 \leq y_1$$

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

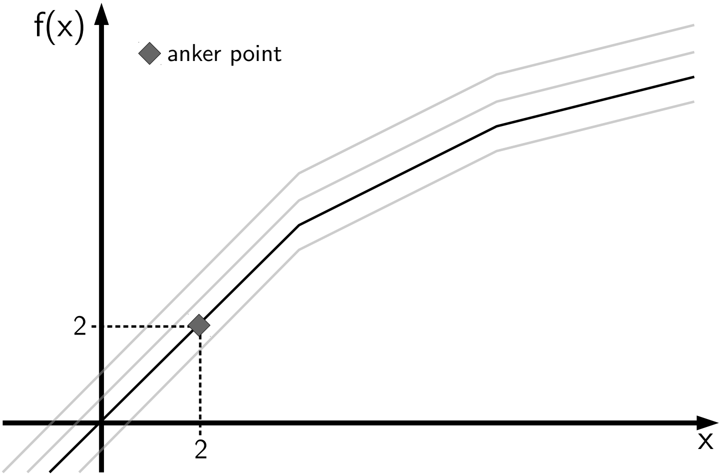
$$z_N \leq y_N$$



Piecewise linear functions by slope



Ankering of piecewise linear functions



Syntax of the `piecewise` command

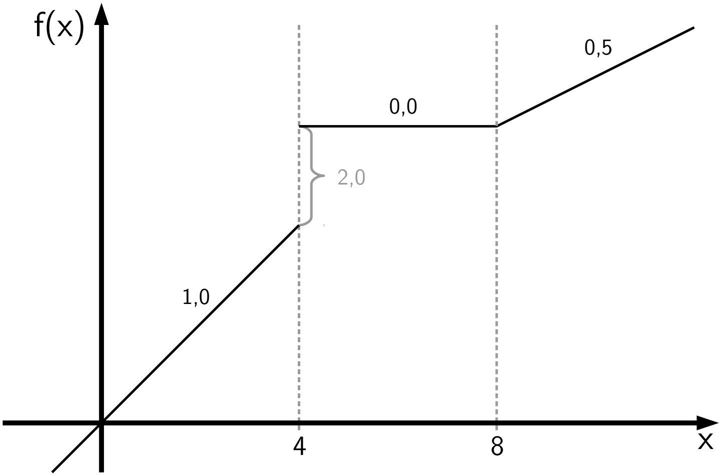
Array `p` of supporting points and array `s` of slopes:

```
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;
```

Step functions and general discontinuities



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;
```