# A Gentle Introduction to Modern Optimization Tools in R

## rcbc & minizinc

Shuai Wang    Eugene Pyatigorsky

84.51 Operations Research

CinDay R User Meetup, May 22 2019

# Table of Contents

# Table of Contents

# What's mathematical optimization anyway?

- "Optimization" comes from the same root as "optimal", which means best. When you optimize something, you are "making it best".
- But "best" can vary. If you're a football player, you might want to maximize your running yards, and also minimize your fumbles. Both maximizing and minimizing are types of optimization problems.

Cin-Day

# Mathematical Optimization in the "Real World"

Mathematical Optimization is a branch of applied mathematics which is useful in many different fields. Here are a few examples:

- Manufacturing
- Production
- Inventory control
- Transportation
- Scheduling
- Networks
- Finance
- Economics
- Control engineering
- Marketing
- Policy Modeling
- Mechanics

# Optimization Model Components

Your basic optimization problem consists of:

1. The objective function, f(x), which is the output you're trying to maximize or minimize. e.g. maximize the gross profit margin; minimize travel distance of a pizza delivery car.
2. Variables, $x_1, x_2, x_3$ and so on, which are the inputs – things you can control.
3. Constraints, which are equations that place limits on how big or small some variables can get. e.g. The pizza delivery should be on time.

Cin-Day

# Optimization Example

A football coach is planning practices for his running backs.

- His main goal is to maximize running yards – this will become his objective function.
- He can make his athletes spend practice time in the weight room; running sprints; or practicing ball protection. The amount of time spent on each is a variable.
- However, there are limits to the total amount of time he has. Also, if he completely sacrifices ball protection he may see running yards go up, but also fumbles, so he may place an upper limit on the amount of fumbles he considers acceptable. These are constraints.

Note that the variables influence the objective function and the constraints place limits on the domain of the variables.

Cin-Day

# Table of Contents

# Knapsack problem

- You only bring one knapsack with a capacity limit to rob a bank.
- Different gold has various amount of value and weight.
- Try to get as much value as possible.
- So, which ones to choose with the capacity limit of the knapsack.

# Knapsack problem application

In any real-world problems where you have resources with certain values and you want to waste as little as possible.

- Shipping containers, to be packed as efficiently as possible.
- To cut large pieces of materials into smaller packages (paper, metal, wood-logs).
- To optimize portfolios (which shares and how many should you buy).

Cin-Day

# Knapsack problem math modeling

KP has the following Integer Linear Programming (ILP) formulation:

$$
\begin{aligned}
\text{maximize} \quad & \sum_{j \in N} p_j x_j && (1) \\
\text{subject to} \quad & \sum_{j \in N} w_j x_j \le c && (2) \\
& x_j \in \{0, 1\}, \quad j \in N, && (3)
\end{aligned}
$$

where each binary variable $x_j$, $j \in N$, is equal to 1 if and only if item $j$ is selected. $p_j$: price/value of each item; $w_j$: weight of each item.
We cannot take all items because the total weight of the chosen items cannot exceed the knapsack capacity $c$.

Cin-Day

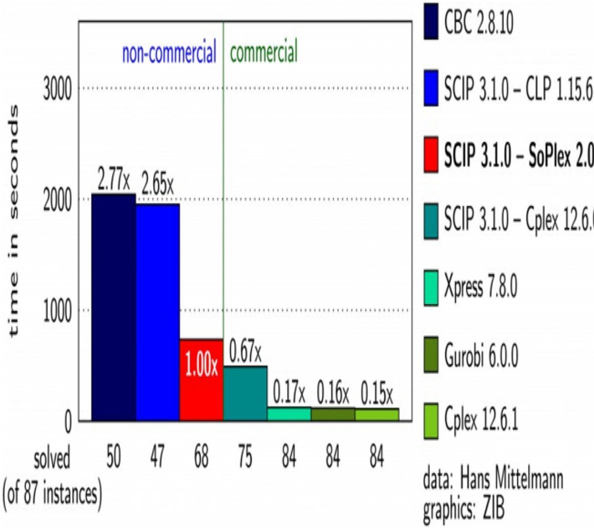
# Common solvers for linear and integer optimization problems

Commercial:

- IBM CPLEX
- Gurobi
- FICO EXPRESS
- MOSEK

Open-Source:

- SCIP (commercial use restriction)
- GLPK
- COIN-OR /CBC
- COIN-OR/SYMPHONY



# Solver Benchmark

The third line lists the number of problems (86 total) solved.

| 1 thr | CBC | CPLEX | GUROBI | SCIPC | SCIPS | XPRESS | MATLB | SAS | MIPCL | GLPK | LP_SOL |
|---|---|---|---|---|---|---|---|---|---|---|---|
| unscal | 1639 | 72.2 | 41.6 | 239 | 330 | 83.1 | 3002 | 121 | 453 | 6925 | 5616 |
| scaled | 39 | 1.74 | 1 | 5.75 | 7.94 | 2.00 | 72.2 | 2.90 | 10.9 | 167 | 135 |
| solved | 53 | 87 | 87 | 83 | 76 | 86 | 32 | 84 | 76 | 2 | 7 |

| 4 thr | CBC | CPLEX | FSCIPC | FSCIPS | GUROBI | XPRESS | MIPCL | SAS |
|---|---|---|---|---|---|---|---|---|
| unscal | 843 | 36.4 | 240 | 294 | 24.2 | 40.3 | 177 | 72.6 |
| scaled | 34.8 | 1.50 | 9.90 | 12.1 | 1 | 1.66 | 7.29 | 3.00 |
| solved | 66 | 86 | 80 | 79 | 87 | 87 | 84 | 85 |

| 12 thr | CBC | CPLEX | FSCIPC | FSCIPS | GUROBI | XPRESS | MIPCL | SAS |
|---|---|---|---|---|---|---|---|---|
| unscal | 668 | 37.5 | 247 | 328 | 25.2 | 39.5 | 165 | 85.4 |
| scaled | 27 | 1.49 | 9.80 | 13.0 | 1 | 1.57 | 6.53 | 3.39 |
| solved | 69 | 87 | 78 | 76 | 87 | 87 | 82 | 82 |

Source: http://plato.asu.edu/talks/informs2018.pdf

# Using library(rcbc)

https://github.com/dirkschumacher/rcbc

```
max_capacity <- 1000
n <- 100
weights <- round(runif(n, max = max_capacity))
price <- round(runif(n) * 100)

A <- matrix(weights, ncol = n, nrow = 1) # matrix for constrains

result <- cbc_solve(
 obj = price,  # define objective sum(price_i)
 mat = A,      # weight_i * n
 # variable x(j) is binary
 is_integer = rep.int(TRUE, n),
# row bound for contraints
 row_lb = 0, row_ub = max_capacity, max = TRUE,
#column bound for vairable
 col_lb = rep.int(0, n), col_ub = rep.int(1, n))
```

Cin-Day

# Minizinc: The Right Tool for the Right Job.

Minizinc is a free and open-source constraint modeling language.
You can use Minizinc to model constraint satisfaction and optimization problems in a high-level, solver-independent way.

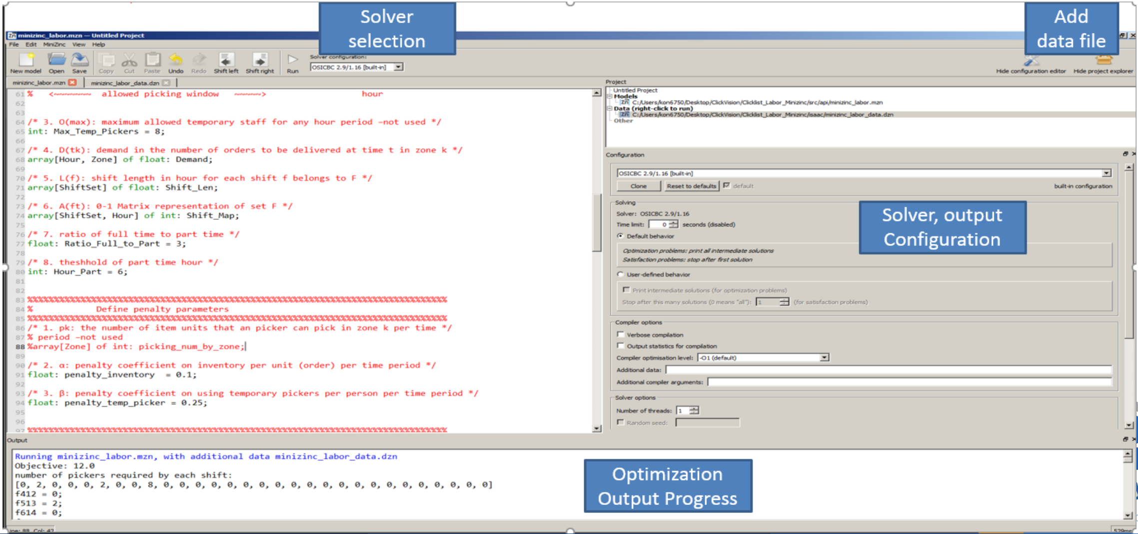
```
int: n; % number of objects
int: capacity;
array[1..n] of int: profit;
array[1..n] of int: size;

array[1..n] of var 0..1: x;

constraint sum(i in 1..n)(size[i] * x[i]) <= capacity;
solve maximize sum(i in 1..n)(profit[i] * x[i]);
```

Cin-Day

# Minizinc GUI



# A Typical Resource Planning Problem

- How much of each kind of product to make to maximize profit where manufacturing a product consumes varying amounts of some fixed resources
- Minizinc uses a data file with non-standard format

```
% Data file for simple production planning model
Products = { BananaCake, ChocolateCake };
profit = [400, 450]; % in cents

Resources = { Flour, Banana, Sugar, Butter, Cocoa };
capacity = [4000, 6, 2000, 500, 500];

consumption= [| 250, 2, 75, 100, 0,
              | 200, 0, 150, 150, 75 |];
```


Cin-Day

# Easy to Express in Minizinc

```
% Data
enum Products;                          % Products to be produced
array[Products] of int: profit;         % profit per unit for each product
enum Resources;                         % Resources to be used
array[Resources] of int: capacity;      % amount of each resource available

% units of each resource required to produce 1 unit of product
array[Products, Resources] of int: consumption;

% Parameter checking
constraint assert(forall (r in Resources, p in Products)
           (consumption[p,r] >= 0), "Error: negative consumption");

% Variables: How much to make of each product and
%            how much of each resource will be consumed
array[Products] of var 0..100: produce;

array[Resources] of var 0..max(capacity): used =
    [sum (p in Products)(consumption[p, r] * produce[p]) | r in Resources];

% Constraint: Cannot consume more than the available resources:
constraint forall (r in Resources)(used[r] <= capacity[r]);

% Maximize profit
solve maximize sum (p in Products)(profit[p]*produce[p]);
```

Cin-Day

- We treat the data file (dzn) as a template to be filled in by R, using handmade casting functions
- ‘glue‘ does the string interpolation

```
% Data file for simple production planning model
Products    = {to_enum(products)};
profit      = {to_array(profit)}; % in cents
Resources   = {to_enum(resources)};
capacity    = {to_array(capacity)};
consumption = {to_matrix(consumption)};
```

Cin-Day

# Minizinc with R: Calling the Executable

- A thin wrapper on top of the minizinc command line executable passes in the model and data files and receives a json-formatted result.

```r
# Data
products    <- c("BananaCake", "ChocolateCake")
profit      <- c(400, 500)
resources   <- c("Flour", "Banana", "Sugar", "Butter", "Cocoa")
capacity    <- c(4000, 6, 2000, 500, 500)
consumption <- matrix(c(250, 2, 75, 100, 0, 200, 0, 150, 150, 75),
                      nrow = 2, byrow = TRUE);

# Run
mzn <- "prod.mzn"
dzn <- read_file("prod.dzn")
dzn <- glue(dzn)
res <- solve_mz(mzn, dzn)

# $solution
# $solution$produce
# [1] 2 2
#
# $status
# [1] "optimal"
```

Cin-Day


Thank you