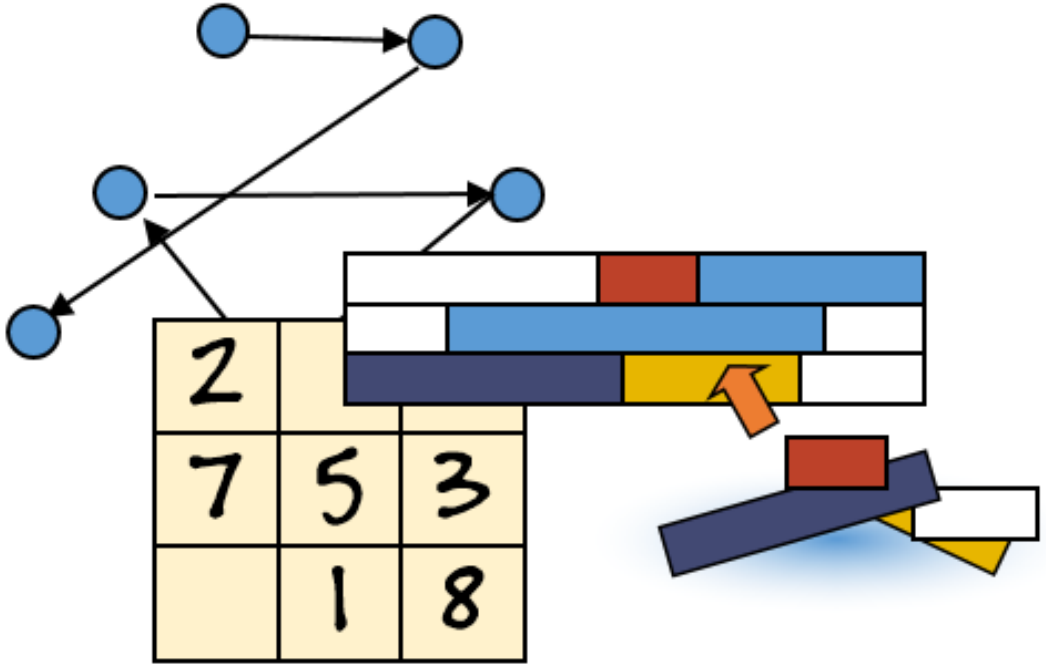
Introduction to Constraint Programming

Manuel Combarro Simón

Interdisciplinary Centre for Security, Reliability and Trust (SnT), Luxembourg

Agenda

1. Constraint satisfaction problem
2. Minizinc
3. Solving algorithm
4. Global constraint
5. Real problem



Constraint Satisfaction Problem (CSP)

$Triplet\ \ < X, D, C >$

$X: Set\ of\ variables$

$D: Domains\ of\ variables$

$C: Set\ of\ constraints$

$Example$

$< \{x, y\}, \left\{\{0,1,2\}_x, \{2,3,4\}_y\right\}, \{x \neq y, x > 1\} >$

$Variables: \{x, y\}$

$Domain:\ x: \{0,1,2\}, y: \{2,3,4\}$

$Constraints: x \neq y,\ \ x > 1$

Constraint Satisfaction Problem (CSP)

Assignment is a function $asn: X \rightarrow \mathbb{Z}$

Example:

- $asn: \{x \rightarrow 0, y \rightarrow 0\}$
- $asn: \{x \rightarrow 2, y \rightarrow 4\}$

A **solution** is an assigment that satisfies all the constraints

$asn: \{x \rightarrow 2, y \rightarrow 4\}$ satisfies $x \neq y,\ x > 1$

A problem can have several solutions, when you want to find the best solution based in one parameter or objective we said it is an optimization problem.

$Example$

$Variables: \{x, y\}$

$Domain:\ x: \{0,1,2\}, y: \{2,3,4\}$

$Constraints: x \neq y,\ x > 1$

MiniZinc: Basic structure

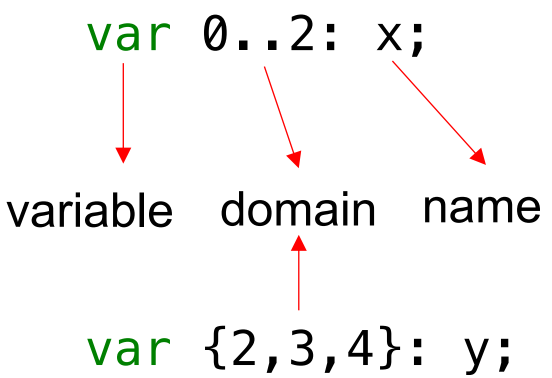
Is modeling language to specify a CSP. https://www.minizinc.org/

$Variables: \{x, y\}$

$Domain:\ x: \{0,1,2\}, y: \{2,3,4\}$

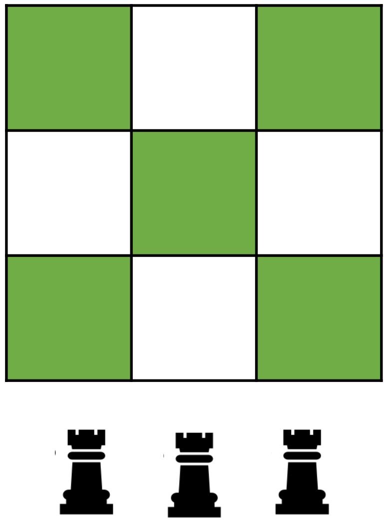
$Constraints: x \neq y,\ \ x > 1$

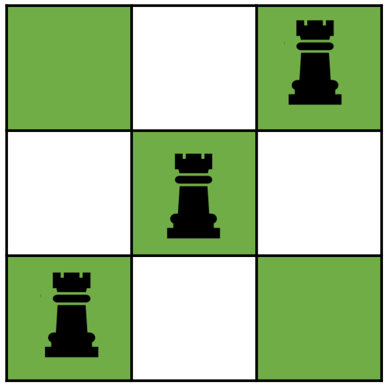
```
var 0..2: x;
var {2,3,4}: y;
constraint x != y;%arithmetic operators, {>,>=,=<,<,!=,=}
constraint x > 1;
solve satisfy;
```



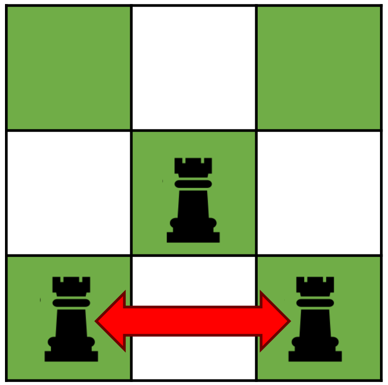
MiniZinc: 3-towers

Can you put 3 towers in a chessboard of 3x3, in a way that they cannot attack each other?





This is a solution



This is **not** a solution

MiniZinc: 3-towers model

Model: Variables, Domains, Constraints

Variables: $\{T_1, T_2, T_3\}$

Domain: $T_1\colon\{0,1,2\}, T_2\colon\{0,1,2\}, T_3\colon\{0,1,2\}$. Domain represents the column

Constraints: $T_1 \neq T_2, T_1 \neq T_3, T_2 \neq T_3$

```
var 0..2: T1;
var 0..2: T2;
var 0..2: T3;
constraint T1 != T2;
constraint T1 != T3;
constraint T2 != T3;
solve satisfy;
```

```
Running 3_towers.mzn
T1 = 2;
T2 = 1;
T3 = 0;
----------
T1 = 1;
T2 = 2;
T3 = 0;
----------
T1 = 2;
T2 = 0;
T3 = 1;
----------
T1 = 0;
T2 = 2;
T3 = 1;
----------
T1 = 1;
T2 = 0;
T3 = 2;
----------
T1 = 0;
T2 = 1;
T3 = 2;
----------
==========
```

MiniZinc: N-queens

Queens -> Row, Column, Diagonal
N -> Parameter not fixed

Exercise:
1. Complete with the missing constraints.
2. Is it possible to get a solution with n=3?
3. How many queens can you solve in less than 5 seconds?

```
int: n=?;
array[1..n] of var 1..n: queens;

constraint forall(i in 1..n, j in i+1..n)
( queens[i]+i != queens[j]+j
/\ queens[i]-i != queens[j]-j);

solve satisfy;
```

→ Generate a conjunction of constraints /\

Solving algorithm

Naive algorithm: Enumerate all possible combination of values

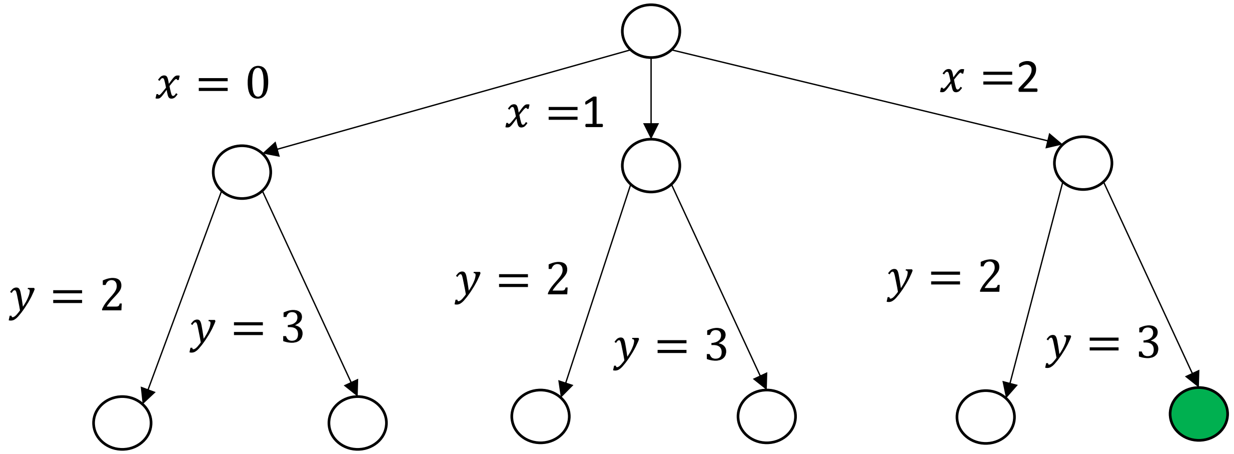
$Variables: \{x, y\}$

$Domain:\ x: \{0,1,2\}, y: \{2,3\}$

$Constraints: x \neq y,\ \ x > 1$

$x = 0, y = 2$
$x = 0, y = 3$
$x = 1, y = 2$
$x = 1, y = 3$
$x = 2, y = 2$
$x = 2, y = 3$

We can get all the possible combinations with the search tree



Solving algorithm

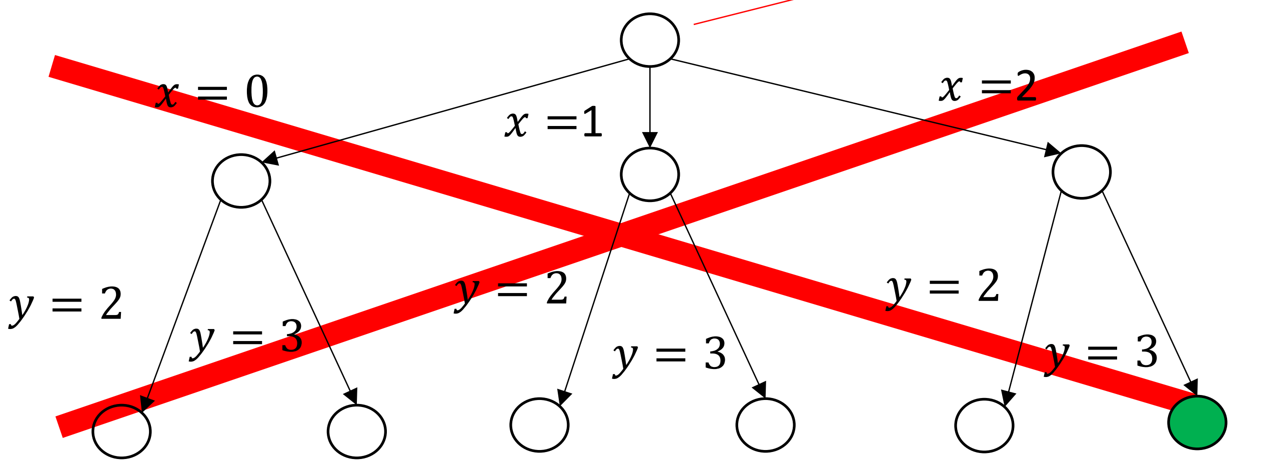
CP solvers perform an inference step, called **propagation**, in each node

- Given the domains and **one** constraint, can we remove values from the domains?

$Variables: \{x, y\}$

$Domain: \ x: \{0,1,2\}, y: \{2,3\}$

$Constraints: x \neq y, \ \ x > 1$



$x \neq y$:
$x: \{0,1,2\}, y: \{2,3\}$

$x > 1$:
$x: \{~~0,1~~, \mathbf{2}\}, y: \{2,3\}$

$x \neq y$:
$x: \{2\}, y: \{2,3\}$

All constraints are satisfied, search is not necessary.

Solutions:
$x = 2, y = 3$

Solving algorithm

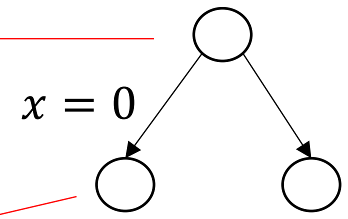
Not always we can find the solutions without searching

$Variables: \{x, y, z\}$

$Domain:\ x: \{0,1\}, y: \{0,1\}, z: \{0,1\}$

$Constraints: x \neq y, y \neq z$

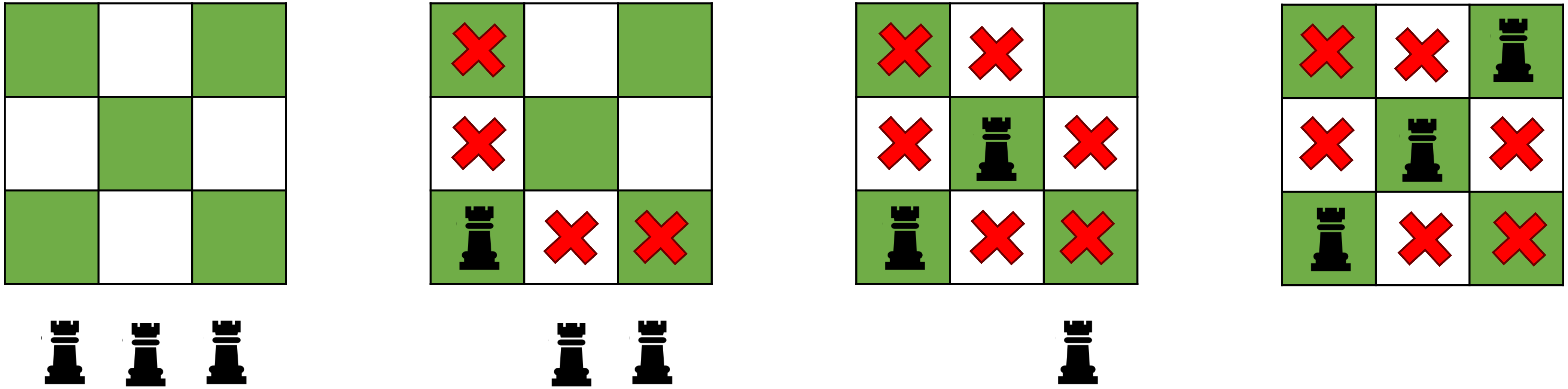
$x \neq y$ and $y \neq z$ do not produce a solution, search



1- $x = 0$ given $x \neq y$ we have $y = 1$, $z: \{0,1\}$
2- $y = 1$ given $y \neq z$ we have $z = 1$
3- Solution $\{x = 0, y = 1, z = 0\}$

MiniZinc: 3-towers

Can you put 3 towers in a chessboard of 3x3, in a way that they cannot attack each other?



Solving algorithm

The interleaving of propagate and search is called ***propagate-and-search*** algorithm.

$solve(< X, D, C >)$

$D' \leftarrow \text{propagate}(< X, D, C >)$

$\textbf{if } \forall d \in D', |d| = 1$

$\textbf{return } \{D'\}$ // we found a solution

$\textbf{if } \exists d \in D', |d| = 0$

$\textbf{return } \{\,\}$ // there are no solution

$\{L, R\} \leftarrow \text{split}(D')$

$\textbf{return } solve(< X, L, C >) \cup solve(< X, R, C >)$ // search

Global constraint

Reasoning locally on constraints is not always the most efficient way to solve the problem

- Global constraints help to reason more globally, find infeasibilities earlier, prune domain better.

$Variables: \{x, y, z\}$

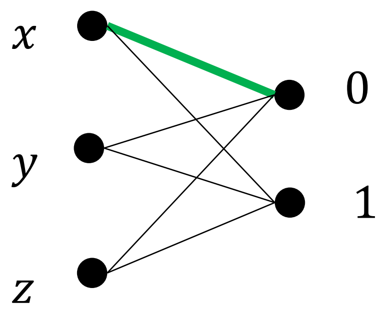
$Domain:\ x: \{0,1\}, y: \{0,1\}, z: \{0,1\}$

$Constraints: x \neq y, y \neq z, y \neq z$

We cannot detect failure when we apply the constraints individually. But with the global constraint $alldifferent$ we can.

Global constraint - alldifferent

$alldifferent(x_1, x_2, \ldots, x_n)$ semantically equivalent to $\{\, x_i \neq x_j \; for\; all\; i \neq j \,\}$ but provides a more efficient propagation algorithm (graph matching).



$Variables: \{x, y, z\}$

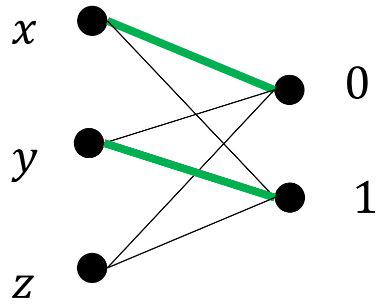
$Domain:\; x: \{0,1\}, y: \{0,1\}, z: \{0,1\}$

$Constraints: x \neq y, y \neq z, y \neq z$

Matching: Subset of edges s.t. no common endpoint exists for any pair of edges.

Global constraint - alldifferent

$alldifferent(x_1, x_2, \ldots, x_n)$ semantically equivalent to $\{ x_i \neq x_j \ for \ all \ i \neq j \}$ but provides a more efficient propagation algorithm (graph matching).



$Variables: \{x, y, z\}$

$Domain: \ x: \{0,1\}, y: \{0,1\}, z: \{0,1\}$

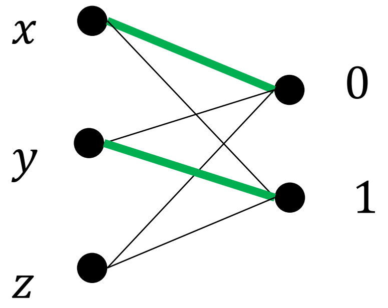
$Constraints: x \neq y, y \neq z, y \neq z$

Matching: Subset of edges s.t. no common endpoint exists for any pair of edges.

Maximum matching: A matching that cannot be augmented by any edge.

Global constraint - alldifferent

$alldifferent(x_1, x_2, \ldots, x_n)$ semantically equivalent to $\{ x_i \neq x_j \; for \; all \; i \neq j \}$ but provides a more efficient propagation algorithm (graph matching).



$Variables: \{x, y, z\}$

$Domain: \; x: \{0,1\}, y: \{0,1\}, z: \{0,1\}$

$Constraints: x \neq y, y \neq z, y \neq z$

Matching: Subset of edges s.t. no common endpoint exists for any pair of edges.

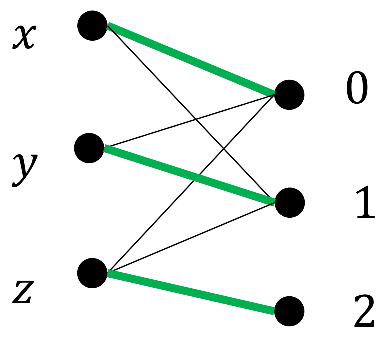
Maximum matching: A matching that cannot be augmented by any edge.

Solution of alldifferent: Maximum matching covering a set of variables.

Infeasible. The cardinality of maximum matching (2) is smaller than the number of variables (3)

Global constraint - alldifferent

Besides detecting infeasibility earlier, can assign values earlier



$z: \{\sout{0,1},2\}$

$Variables: \{x, y, z\}$

$Domain: \; x: \{0,1\}, y: \{0,1\}, z: \{0,1,2\}$

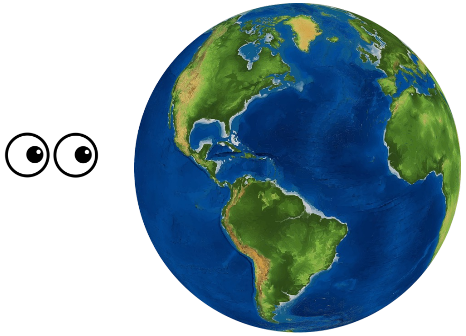
$Constraints: x \neq y, y \neq z, y \neq z$

Global constraint - alldifferent

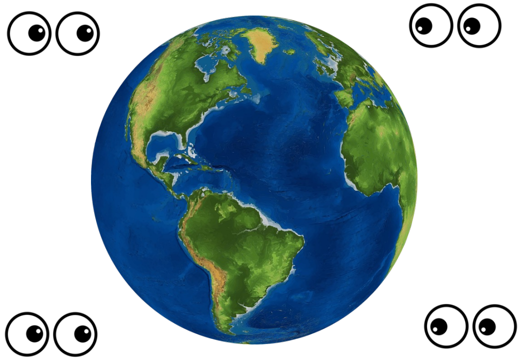
Exercise: Try alldifferent in N-queens and check the efficiency.

```
include "alldifferent.mzn";

int: n=200;
array[1..n] of var 1..n: queens_alldiff;

constraint alldifferent(queens_alldiff);
constraint alldifferent([queens_alldiff[i]+i | i in 1..n]);
constraint alldifferent([queens_alldiff[i]-i | i in 1..n]);

solve satisfy;
```

Satellite image selection problem (SIMS)



2014
192 EO satellites



2021
971 EO satellites

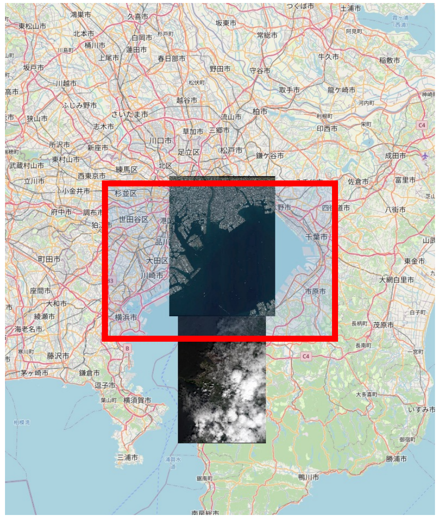
>100 TB of satellite imagery per **day**

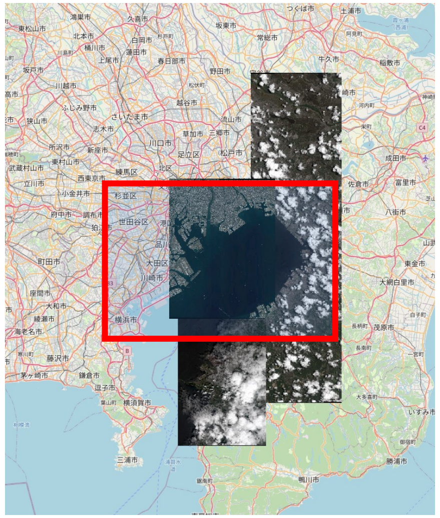
Satellite images: Planet Labs, Inc.

Satellite image selection problem (SIMS)

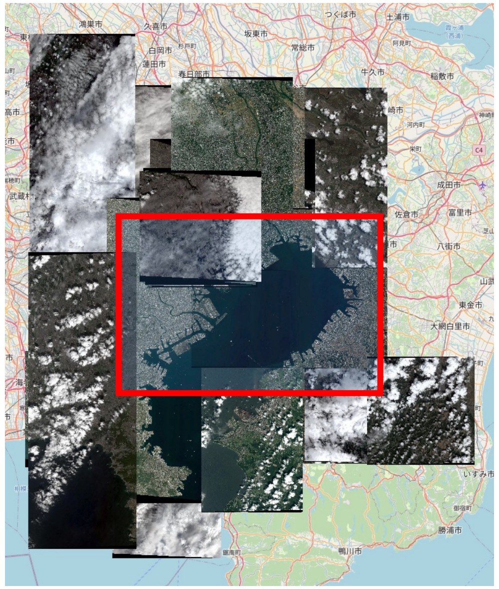
To cover large areas we need several images

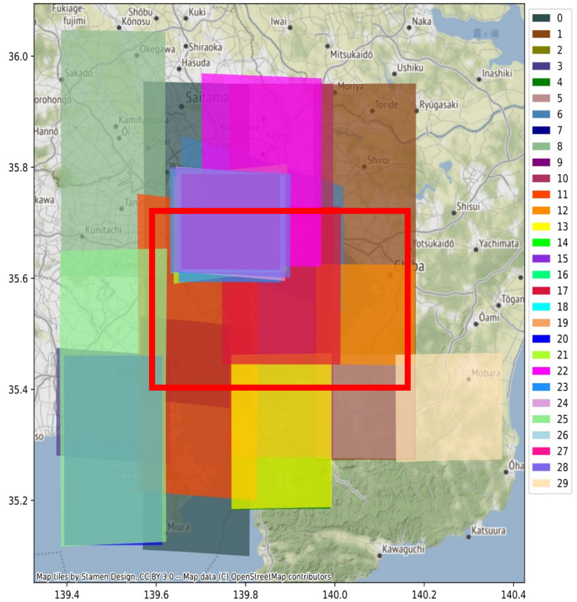
Mosaic



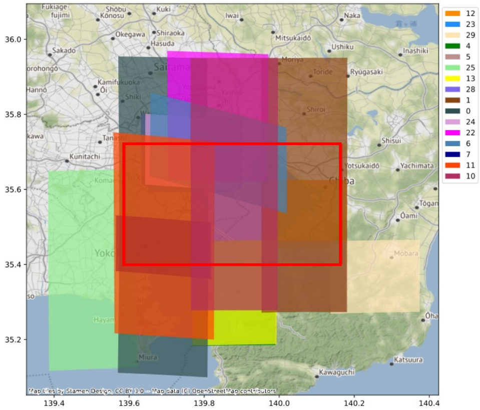


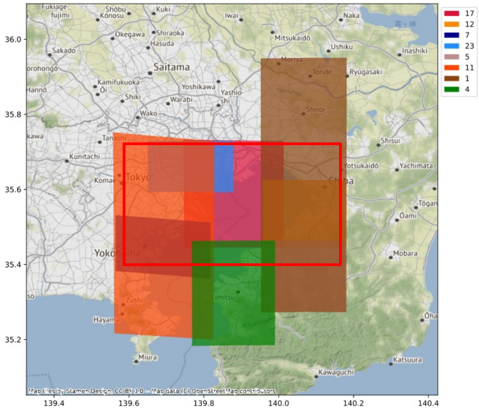
Satellite image selection problem (SIMS)

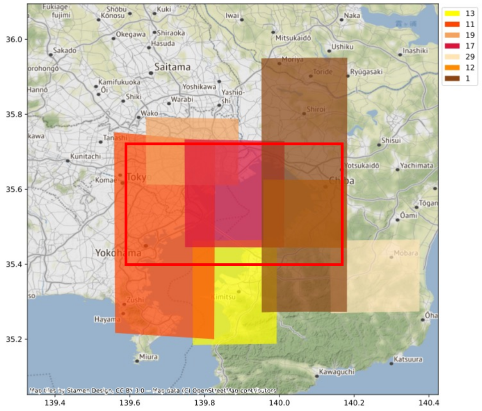


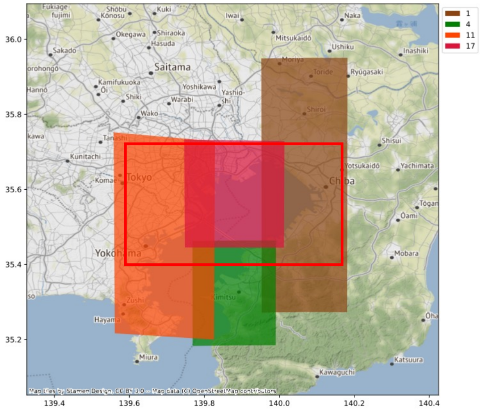


Which combination?
NP-Hard
Enumeration: 2^n

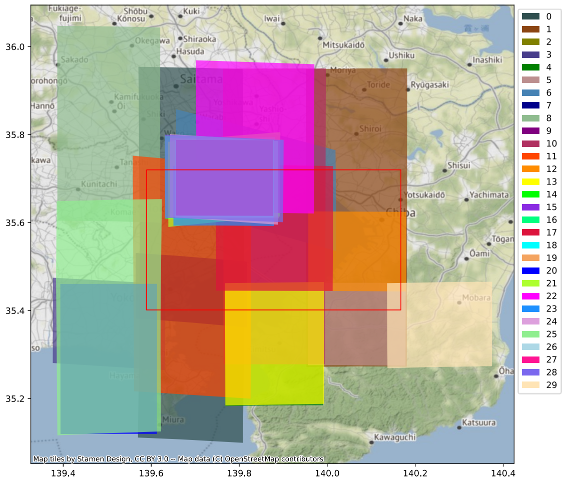






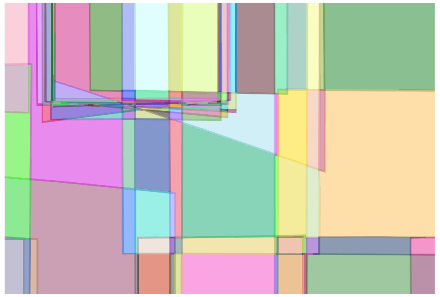


Satellite image selection problem (SIMS)



Remove the area of images outside AOI

Find all intersections

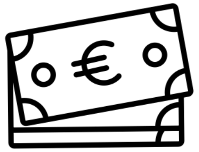


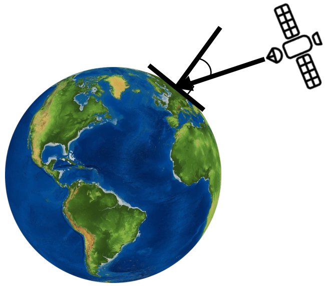
The cover constraint and cost can be modeled as the classical *weighted set cover* problem

Universe = Union of intersections (parts)
Images -> Sets with parts and weight = cost

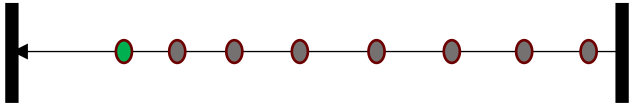
Satellite image selection problem (SIMS) - Model

Multi-objective problem:
- Cost
- Clouds
- Resolution
- Incidence angle



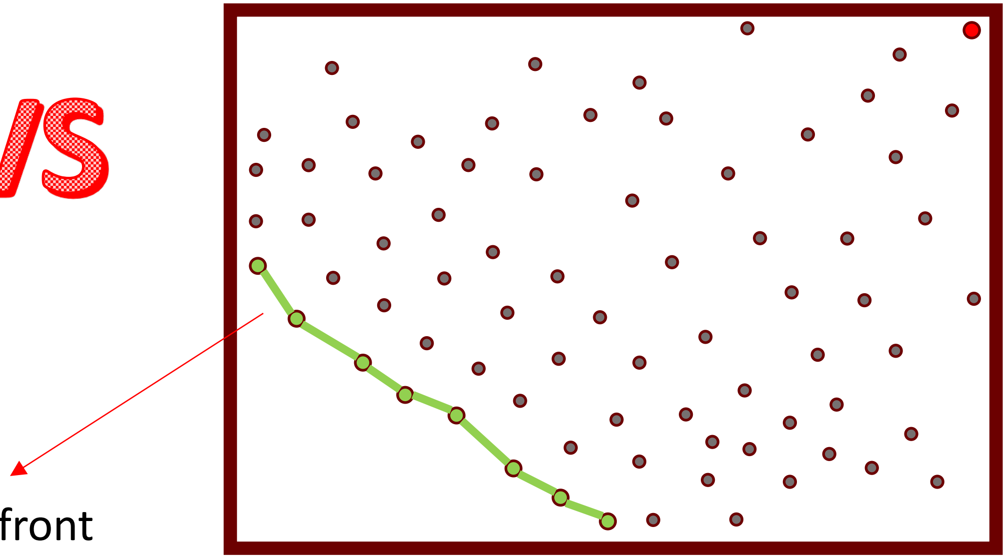


Single objective



VS

Multiobjective



Satellite image selection problem (SIMS) - Model

$Variables: \{taken_i | i = 1, .. n\}$

$Domain: taken_i: \{false, true\}$

$Constraints: cover$

$Objectives: \; cost, resolution, incidence$

Cover constraint:

$$\bigvee_{i:u \in Img_i} taken_i = \text{true}, \qquad \text{for all } u \in Universe$$

```
constraint forall(u in UNIVERSE)(exists(i in IMAGES)(taken[i] /\ u in images[i]));
```

Satellite image selection problem (SIMS) - Model

Cost:

$$min \sum_{i \in Img} cost_i * taken_i$$

```
var int: total_cost = sum(i in IMAGES)(costs[i] * taken[i]);
```

Satellite image selection problem (SIMS) - Model

Resolution:

$$min \sum_{u \in Universe} min\,\{ R_i \mid u \in P_i, taken_i = \text{true}\}$$

```
var int: max_resolution = sum(u in UNIVERSE)(min(i in IMAGES where u in images[i] /\ taken[i])(resolution[i]));
```

Incidence angle:

$$min\{ max\{ taken_i * Inc_i \mid i \ \in Img\}\}$$

```
var int: max_incidence = max(i in IMAGES)(taken[i] * incidence_angle[i]);
```

SnT

Parallel Computing and Optimisation Group

Contact:

Manuel Combarro Simón
Doctoral researcher
manuel.combarrosimon@uni.lu

Pierre Talbot
Research associate
pierre.talbot@uni.lu

Connect with us

@SnT_uni_lu

SnT, Interdisciplinary Centre for Security, Reliability and Trust