Combinatorial optimization @ Google™

CO@Work 2020, Pawel Lichocki, 25.09.2020

https://developers.google.com/optimization

Introduction

Deep dive

Benefits

Challenges

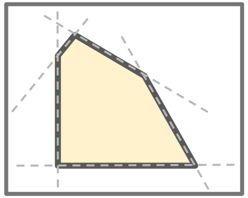
Introduction

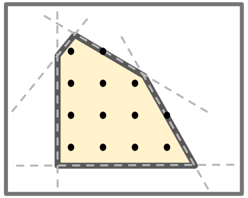
Deep dive

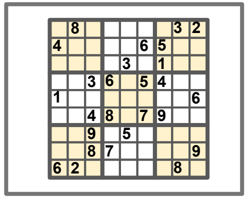
Benefits

Challenges

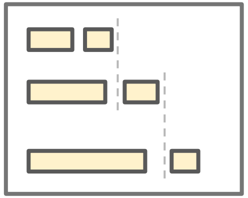
Combinatorial optimization

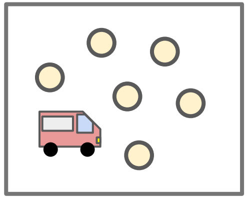


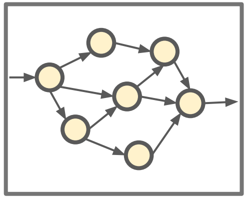




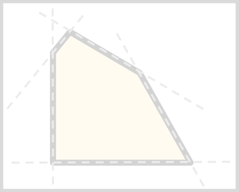
	8						3	2
4					6	5		
				3		1		
		3	6		5	4		
1								6
		4	8		7	9		
		9		5				
		8	7					9
6	2						8	

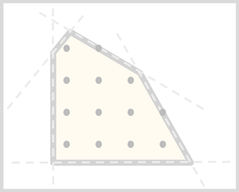


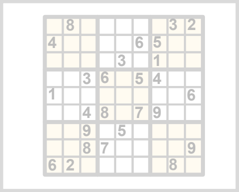


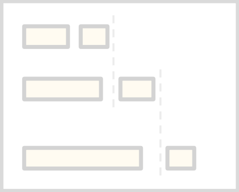


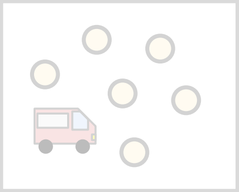
Solvers

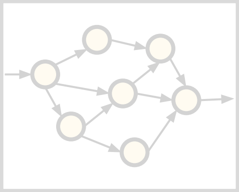












LP

MIP

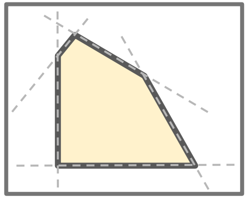
SAT

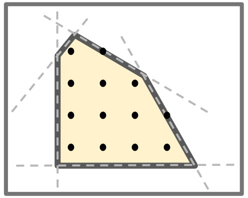
CP

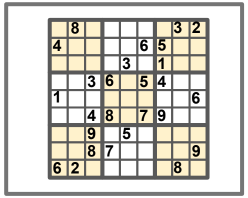
VRP

Graph

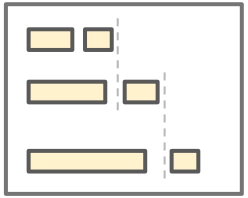
OR-tools

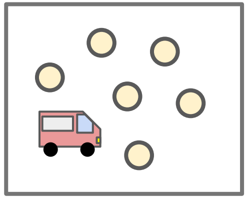


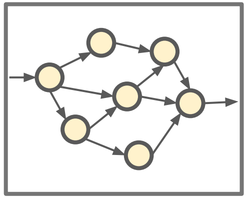




	8						3	2
4					6	5		
				3		1		
		3	6		5	4		
1								6
		4	8		7	9		
		9		5				
		8	7					9
6	2						8	







LP

MIP

SAT

CP

VRP

Graph

Image stabilization

Datacenter optimization

Street view

Loon

Introduction

Deep dive

Benefits

Challenges

MIP model

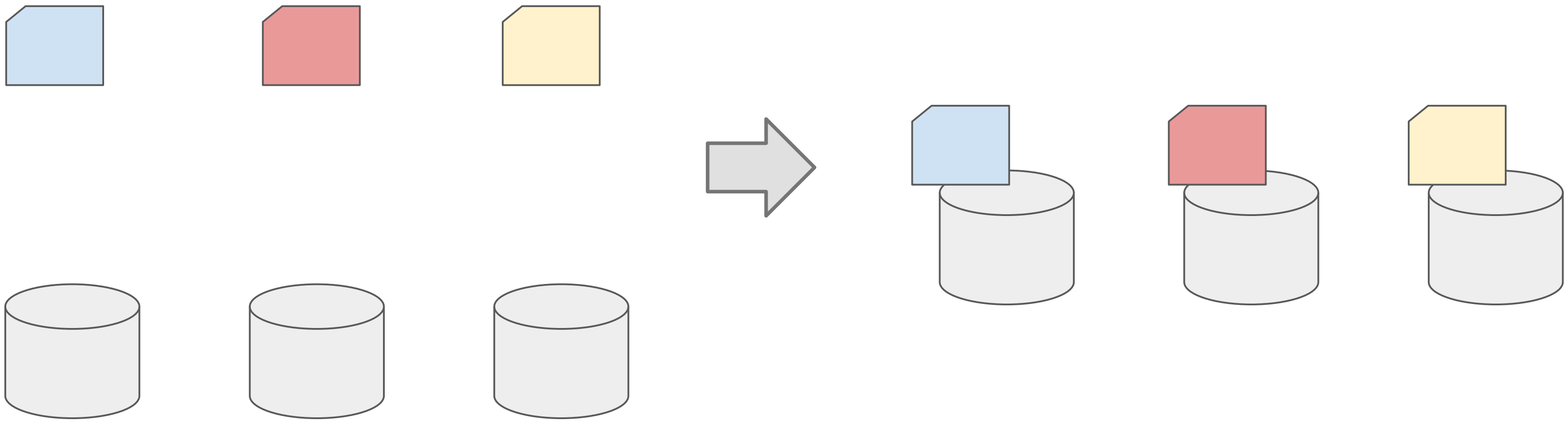
$$\text{min/max } c_0 + c^T x$$

$$lb_{ct} \leq Ax \leq ub_{ct}$$

$$lb_{var} \leq x \leq ub_{var}$$

$$x_j \in Z,\ j \in J$$

Place items



Indices

```
  Item i = 1..I
  Bin b = 1..B
  Resource r = 1..R
```

Variables

Constants

```
  double Required(i, r)
  double Available(b, r)
```

Constraints

Objective

Indices

```
  Item i = 1..I
  Bin b = 1..B
  Resource r = 1..R
```

Variables

```
  place(i, b) in {0, 1}
```

Constants

```
  double Required(i, r)
  double Available(b, r)
```

Constraints

Objective

Indices

```
  Item i = 1..I
  Bin b = 1..B
  Resource r = 1..R
```

Variables

```
  place(i, b) in {0, 1}
```

Constants

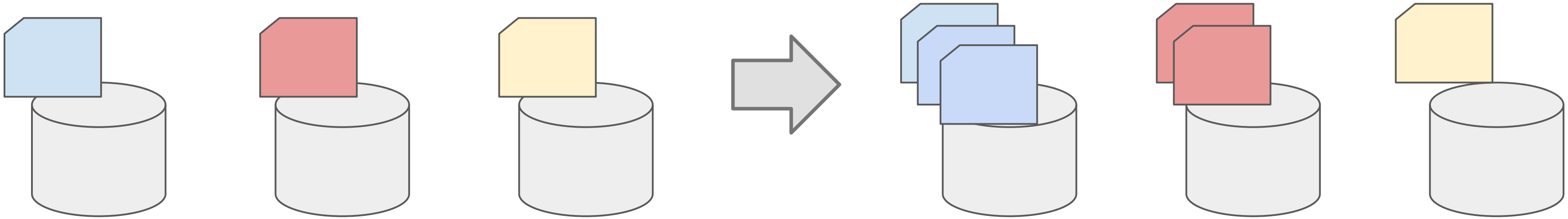
```
  double Required(i, r)
  double Available(b, r)
```

Constraints

```
  for item i = 1..I:
    $\sum_{b = 1..B}$ place(i, b) = 1

  for resource r = 1..R:
    for bin b = 1..B:
      $\sum_{i = 1..I}$ Required(i, r) * place(i, b) ≤ Available(b, r)
```

Objective

Redundancy



Indices

```
Item i = 1..I
Bin b = 1..B
Resource r = 1..R
```

Variables

```
place(i, b) in [0..Copies(i)]
```

Constants

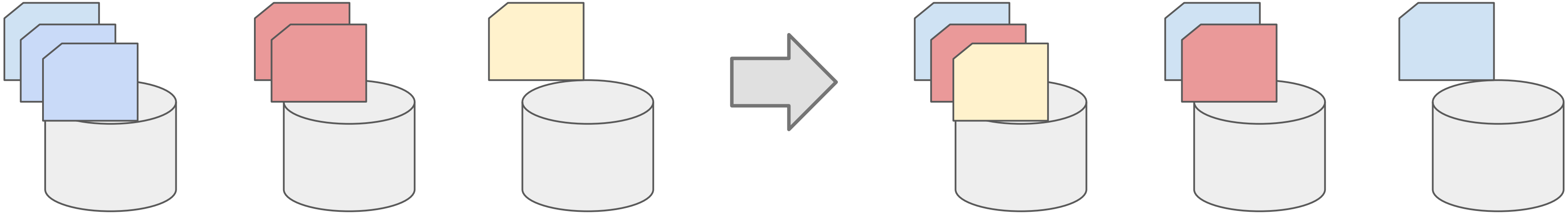
```
int Copies(i)
double Required(i, r)
double Available(b, r)
```

Constraints

```
for item i = 1..I:
  $\sum_{b = 1..B}$ place(i, b) = Copies(i)

for resource r = 1..R:
  for bin b = 1..B:
    $\sum_{i = 1..I}$ Required(i, r) * place(i, b) $\leq$ Available(b, r)
```

Objective

Fault tolerance



Indices

```
Item i = 1..I
Bin b = 1..B
Resource r = 1..R
```

Variables

```
place(i, b) in {0, 1}
```

Constants

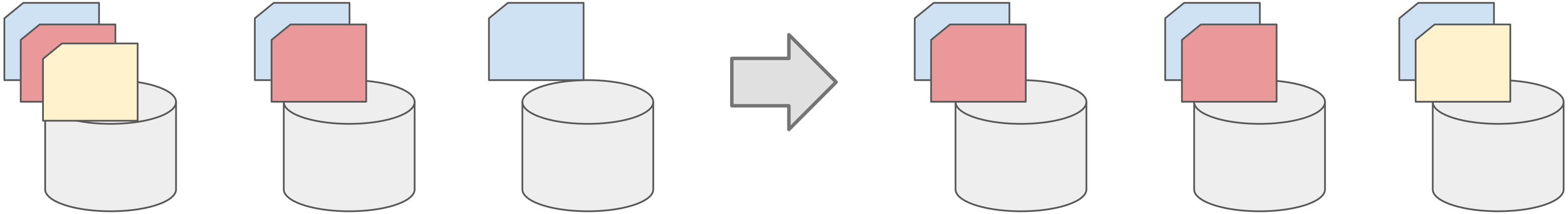
```
int Copies(i)
double Required(i, r)
double Available(b, r)
```

Constraints

```
for item i = 1..I:
  $\sum_{b = 1..B}$ place(i, b) = Copies(i)

for resource r = 1..R:
  for bin b = 1..B:
    $\sum_{i = 1..I}$ Required(i, r) * place(i, b) ≤ Available(b, r)
```

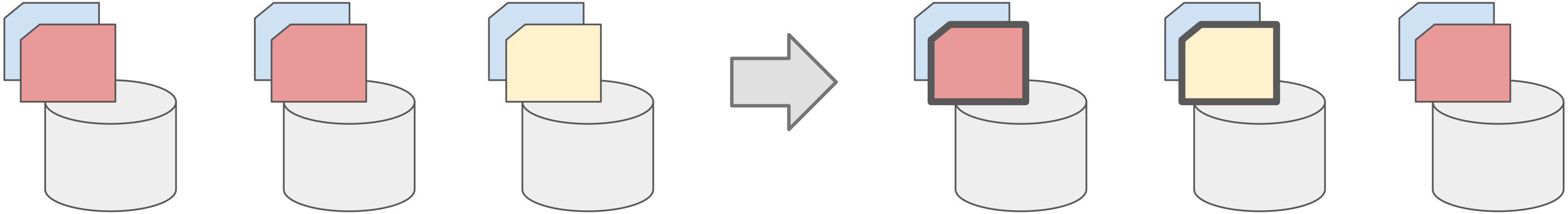
Objective

Balance



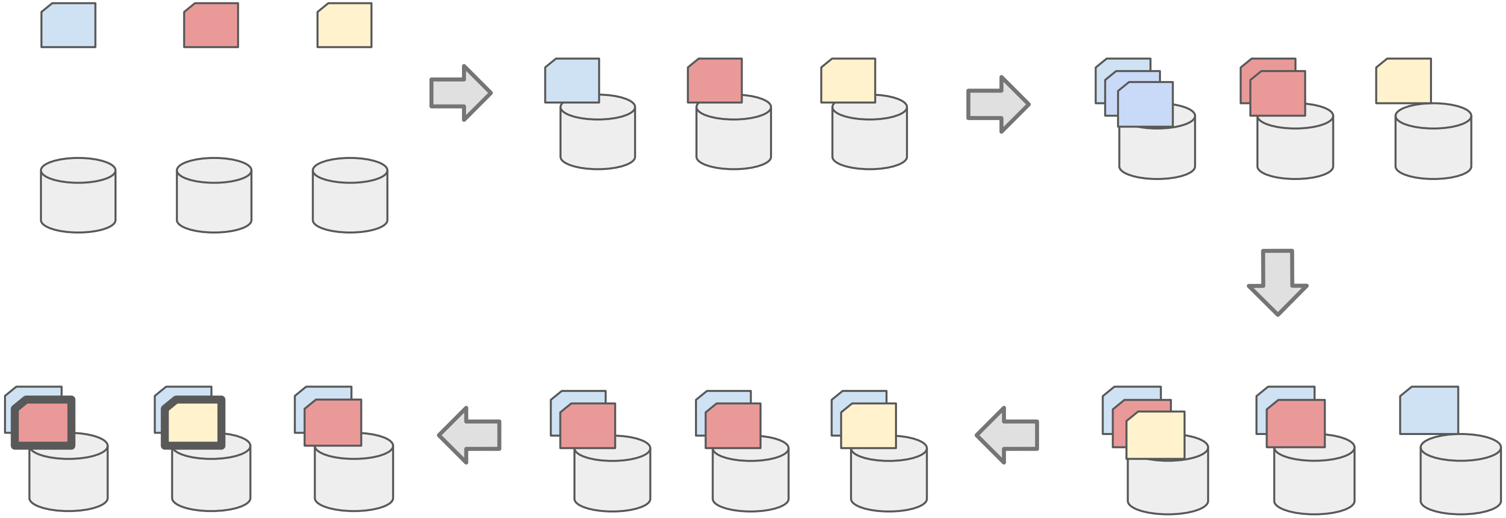
```
Indices
  Item i = 1..I
  Bin b = 1..B
  Resource r = 1..R

Variables
  place(i, b) in {0, 1}
  surplus(b) in [0, +inf)

Constants
  int Copies(i)
  double Required(i, r)
  double Available(b, r)

Constraints
  for item i = 1..I:
    $\sum_{b = 1..B}$ place(i, b) = Copies(i)

  for resource r = 1..R:
    for bin b = 1..B:
      $\sum_{i = 1..I}$ Required(i, r) * place(i, b) ≤ Available(b, r)

  for bin b = 1..B:
    $\sum_{i = 1..I}$ place(i, b) - $\sum_{i = 1..I}$Copies(i) / B ≤ surplus(b)

Objective
  min $\sum_{b = 1..B}$surplus(b)
```

Minimize churn



```
Indices
  Item i = 1..I
  Bin b = 1..B
  Resource r = 1..R

Variables
  place(i, b) in {0, 1}
  surplus(b) in [0, +inf)

Constants
  int Copies(i)
  double Required(i, r)
  double Available(b, r)
  int MaxChange
  bool Placed(i, b)

Constraints
  for item i = 1..I:
    $\sum_{b = 1..B}$ place(i, b) = Copies(i)

  for resource r = 1..R:
    for bin b = 1..B:
      $\sum_{i = 1..I}$ Required(i, r) * place(i, b) ≤ Available(b, r)

  for bin b = 1..B:
    $\sum_{i = 1..I}$ place(i, b) - $\sum_{i = 1..I}$Copies(i) / B ≤ surplus(b)

  $\sum_{b = 1..B}\sum_{i = 1..I}$Placed(i, b) * (1 - place(i, b)) ≤ MaxChange

Objective
  min $\sum_{b = 1..B}$surplus(b)
```

Multi-dimensional multi-packing with redundancy, fault tolerance, balancing, and reducing churn



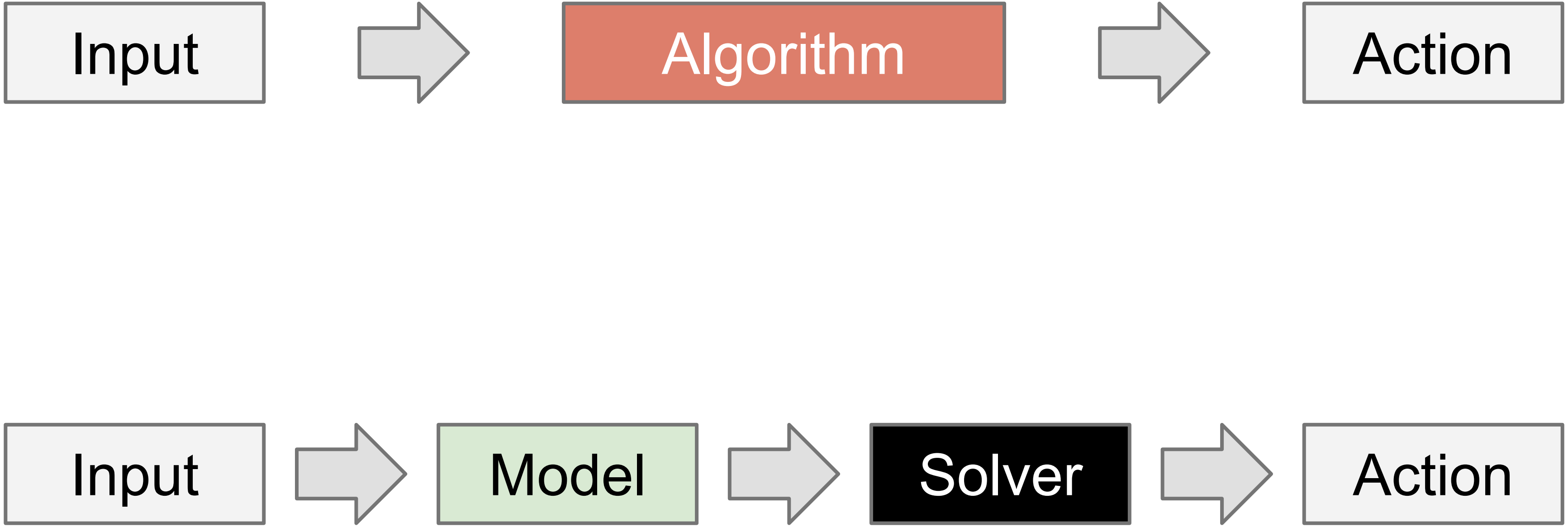
Introduction

Deep dive

Benefits

Challenges

Imperative vs Declarative



Minimalistic

$$\min/\max\ c_0 + c^T x$$

$$lb_{ct} \le Ax \le ub_{ct}$$

$$lb_{var} \le x \le ub_{var}$$

$$x_j \in Z \quad j \in J$$

Modular

Constraints

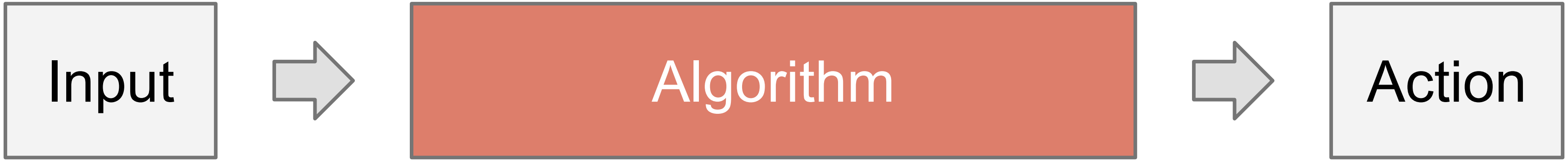
```
for item i = 1..I:
  $\sum_{b = 1..B}$ place(i, b) = Copies(i)
```

```
for resource r = 1..R:
  for bin b = 1..B:
    $\sum_{i = 1..I}$ Required(i, r) * place(i, b) ≤ Available(b, r)
```

```
for bin b = 1..B:
  $\sum_{i = 1..I}$ place(i, b) - $\sum_{i = 1..I}$Copies(i) / B ≤ surplus(b)
  surplus(b) ≤ max_surplus
```

```
$\sum_{b = 1..B}\sum_{i = 1..I}$Placed(i, b) * (1 - place(i, b)) ≤ MaxChange
```

Encapsulated

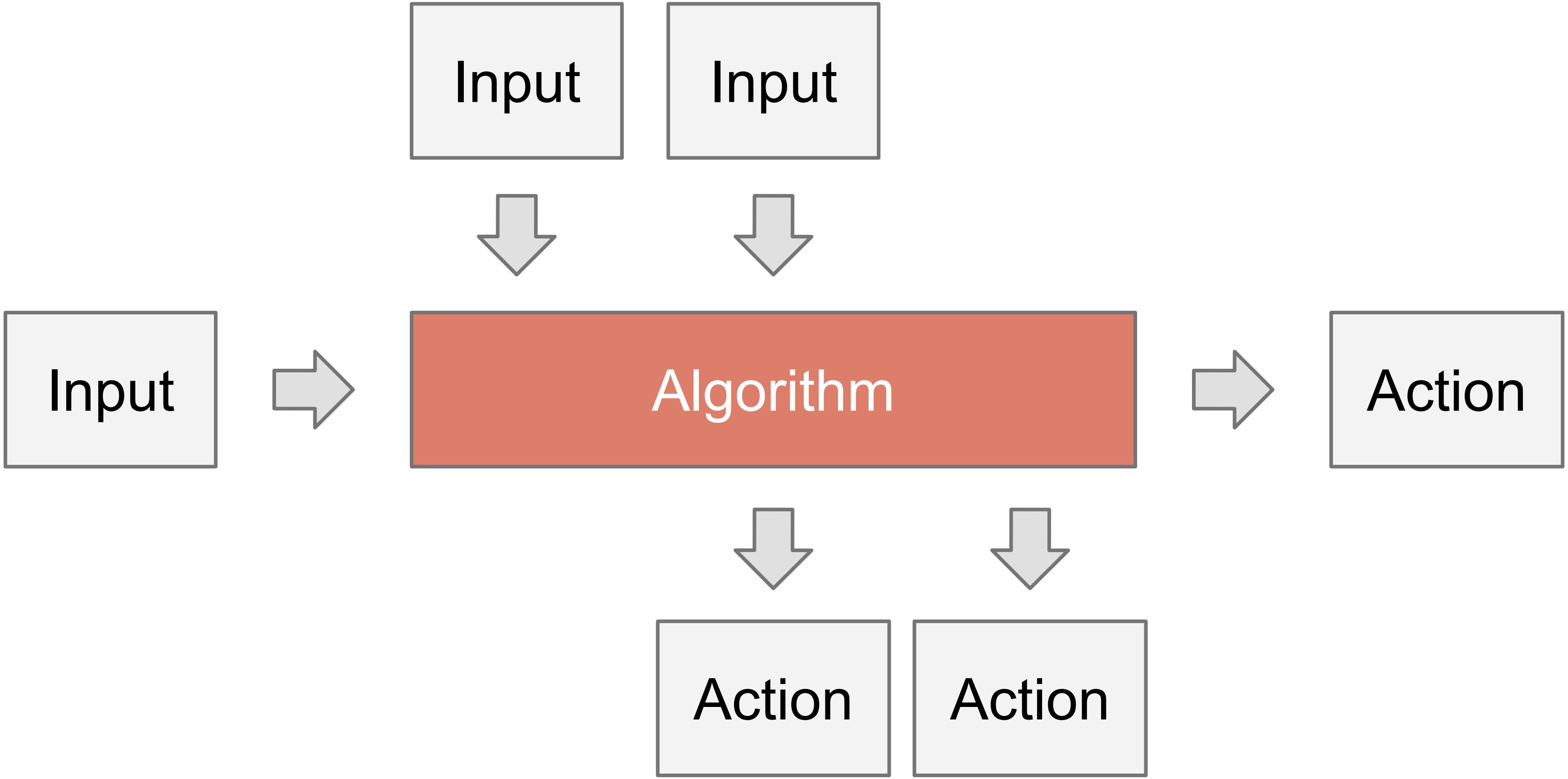


Input

Algorithm

Action

Encapsulated



Input

Input

Input

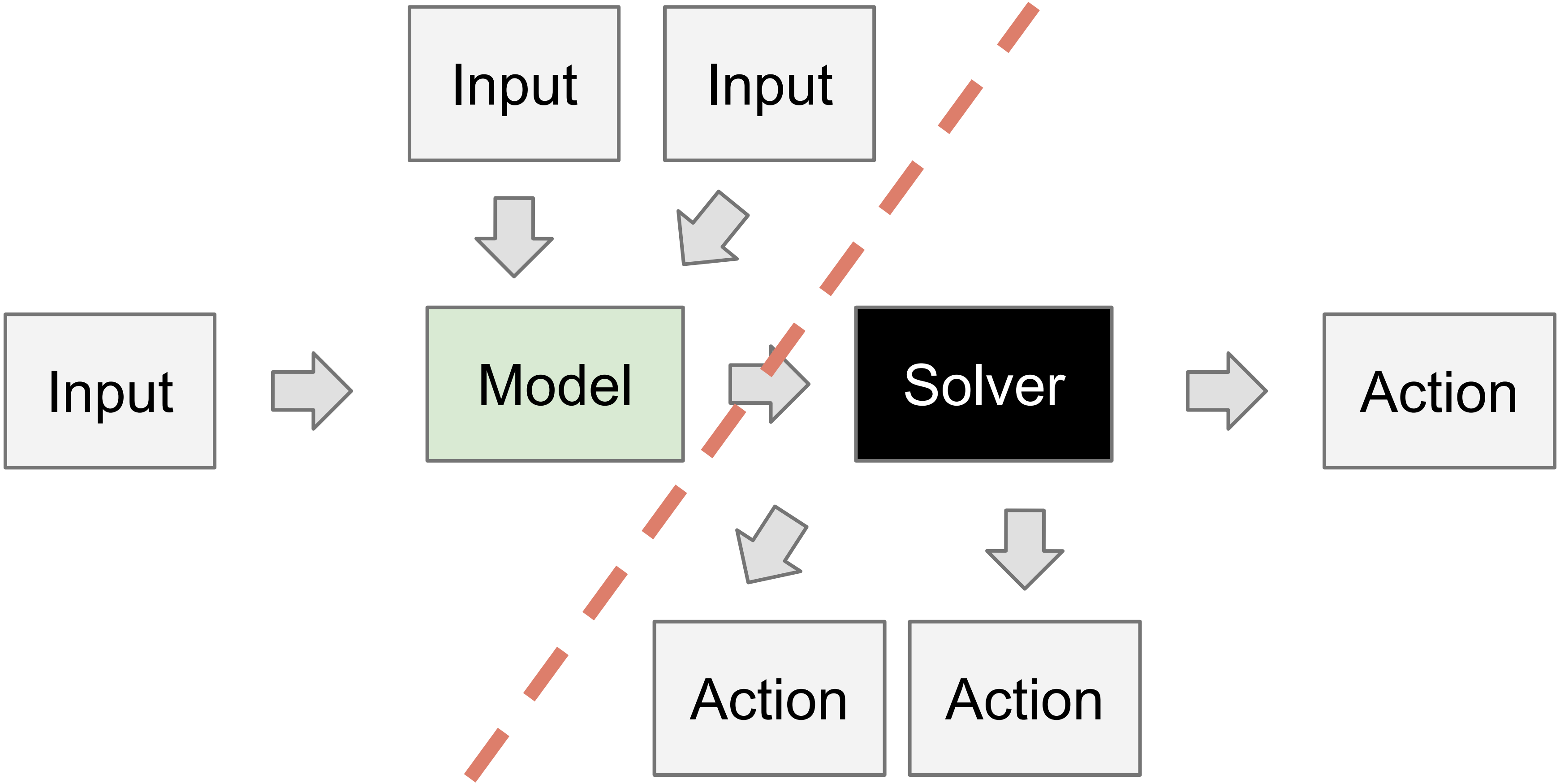
Algorithm

Action

Action

Action

Encapsulated



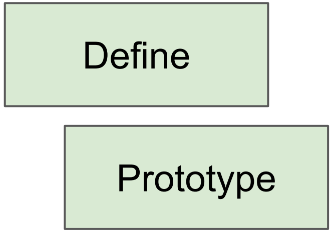
Introduction

Deep dive

Benefits

Challenges

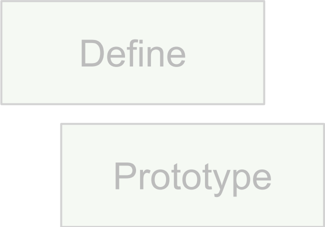
Meet



Define

Prototype

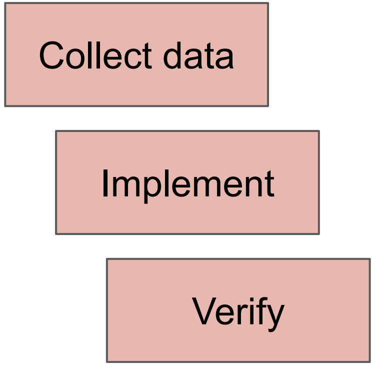
Meet



Define

Prototype

Solve



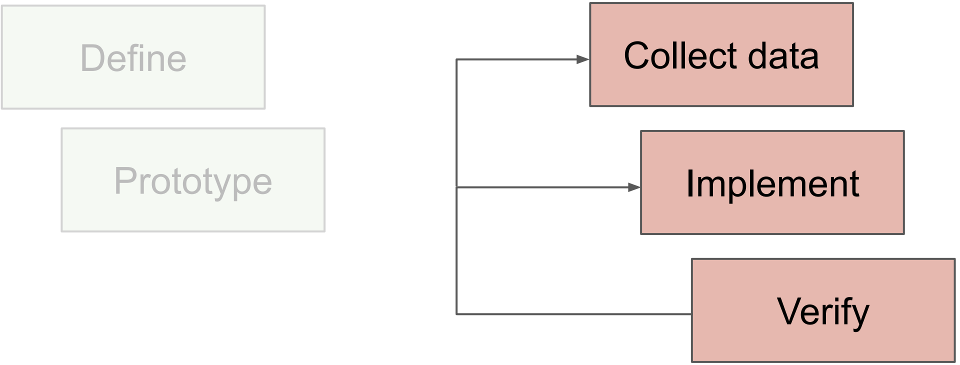
Collect data

Implement

Verify

Meet

Solve



Define

Prototype

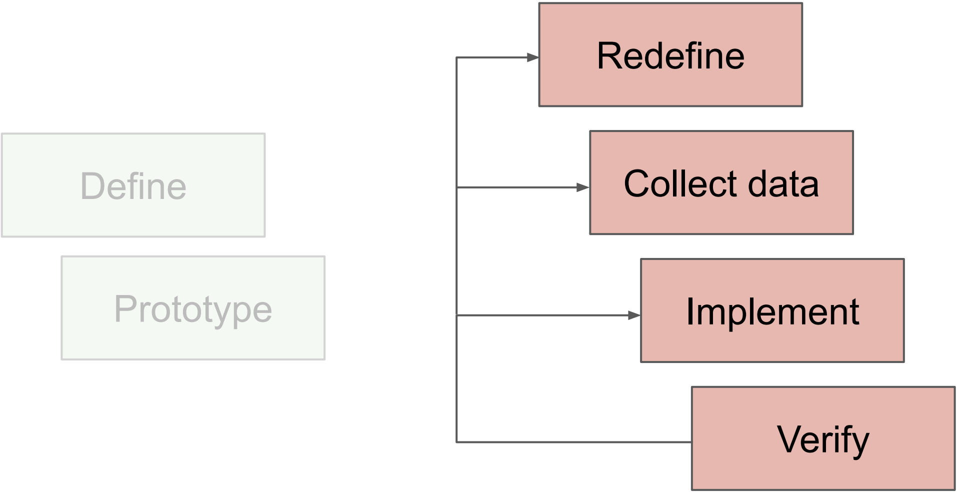
Collect data

Implement

Verify

Meet

Solve



Define

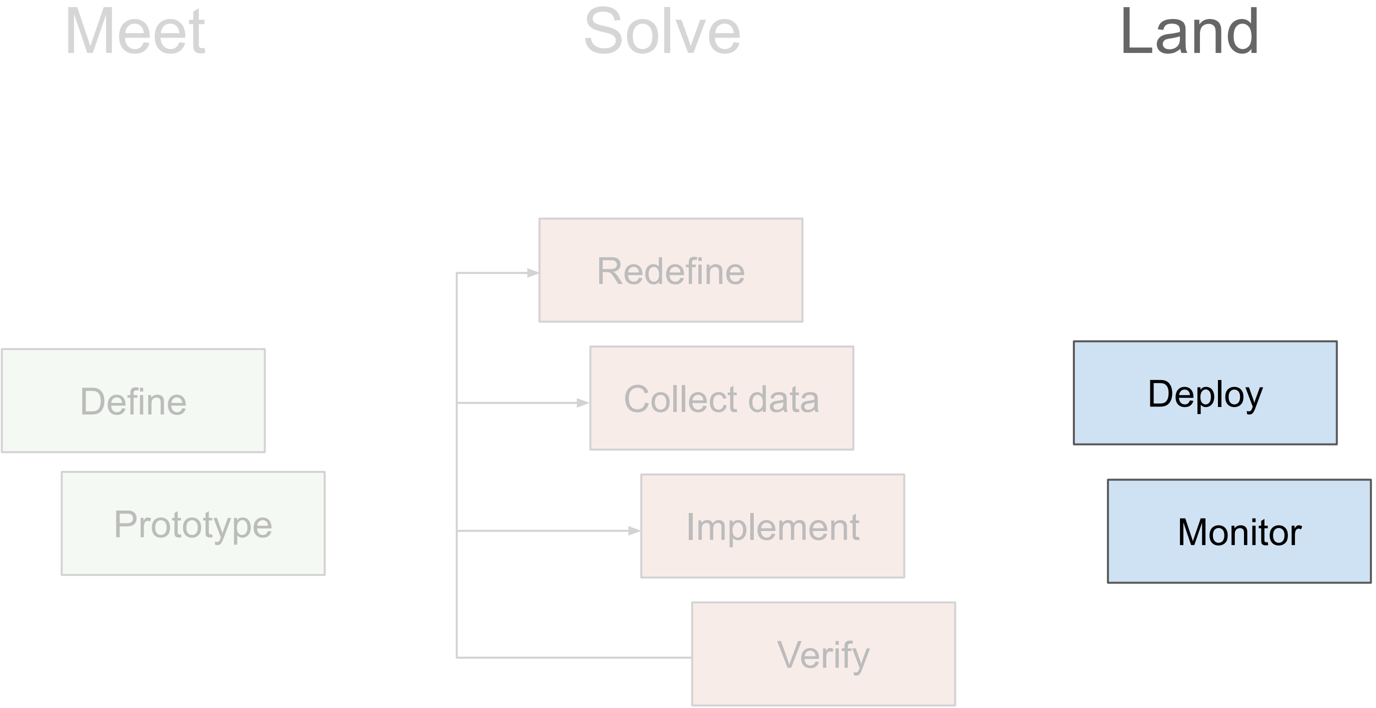
Prototype

Redefine

Collect data

Implement

Verify



Meet

Define

Prototype

Solve

Redefine

Collect data

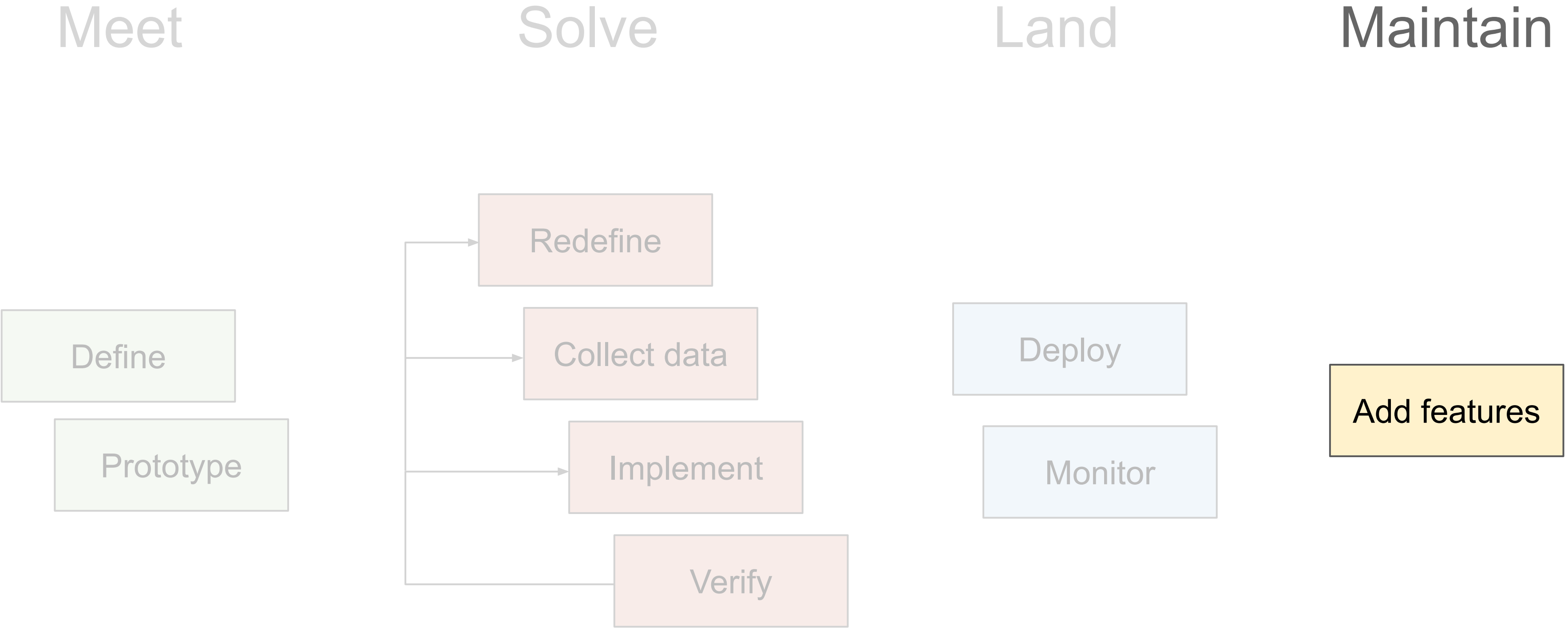
Implement

Verify

Land

Deploy

Monitor



Meet

Define

Prototype

Solve

Redefine

Collect data

Implement

Verify

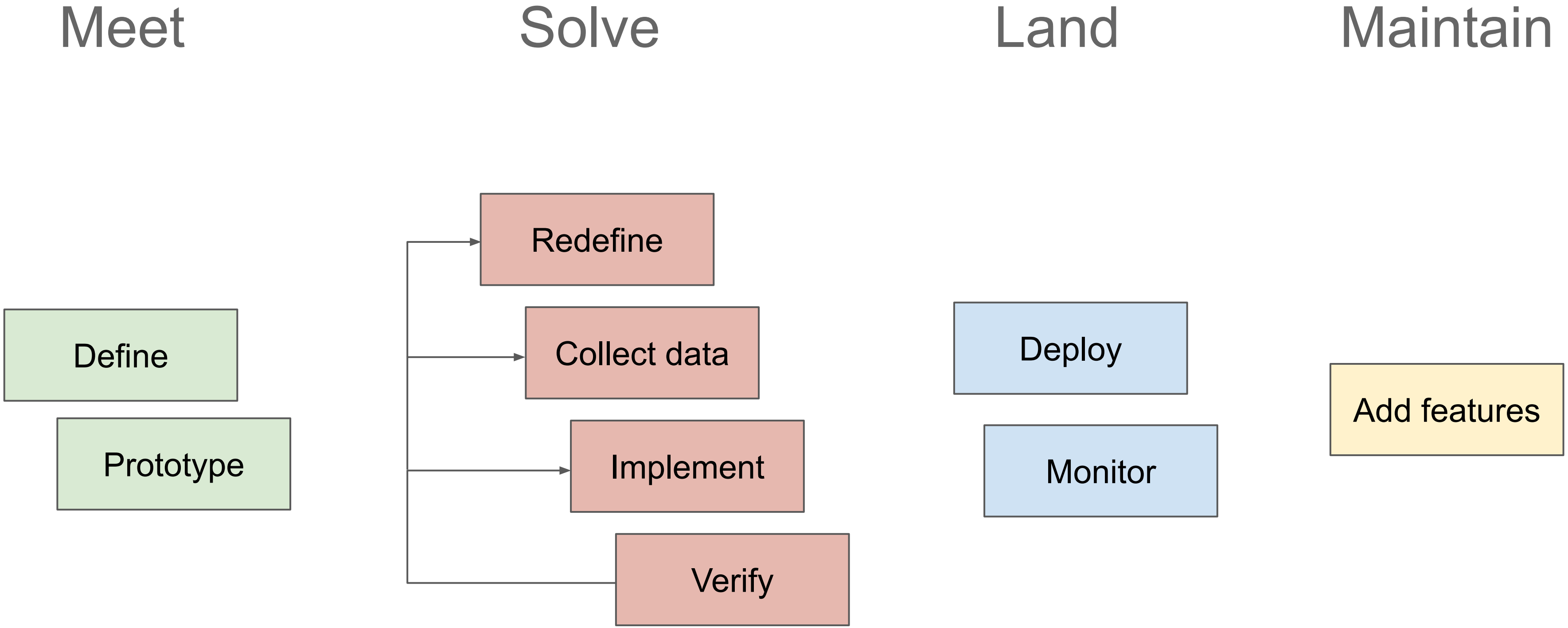
Land

Deploy

Monitor

Maintain

Add features



Meet

Define

Prototype

Solve

Redefine

Collect data

Implement

Verify

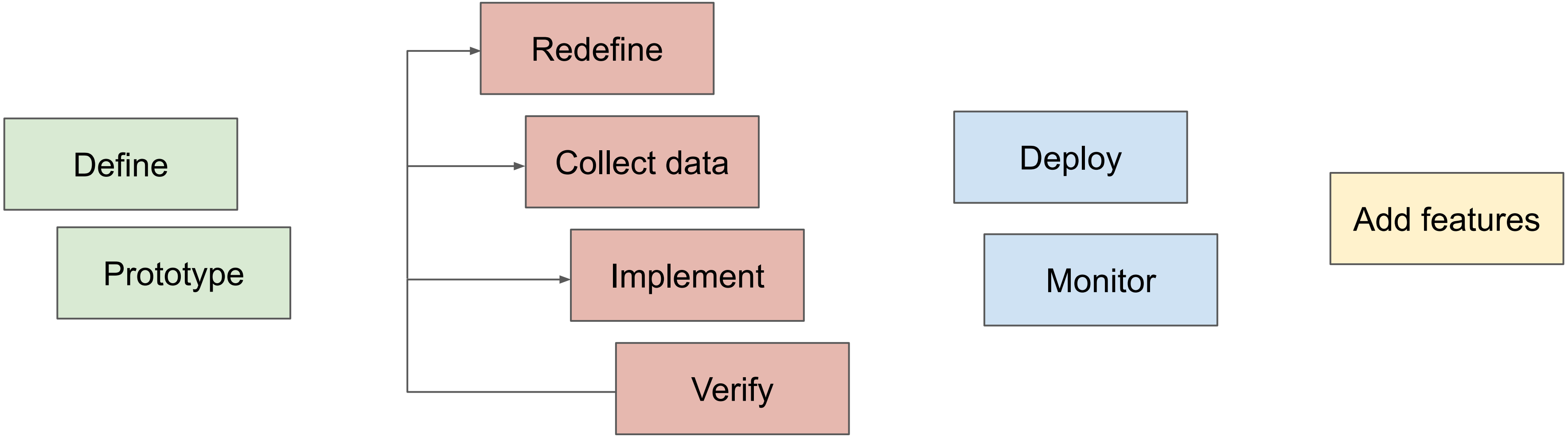
Land

Deploy

Monitor

Maintain

Add features



Meet

1 month

Define

Prototype

Solve

+1 year

Redefine

Collect data

Implement

Verify

Land

3 months

Deploy

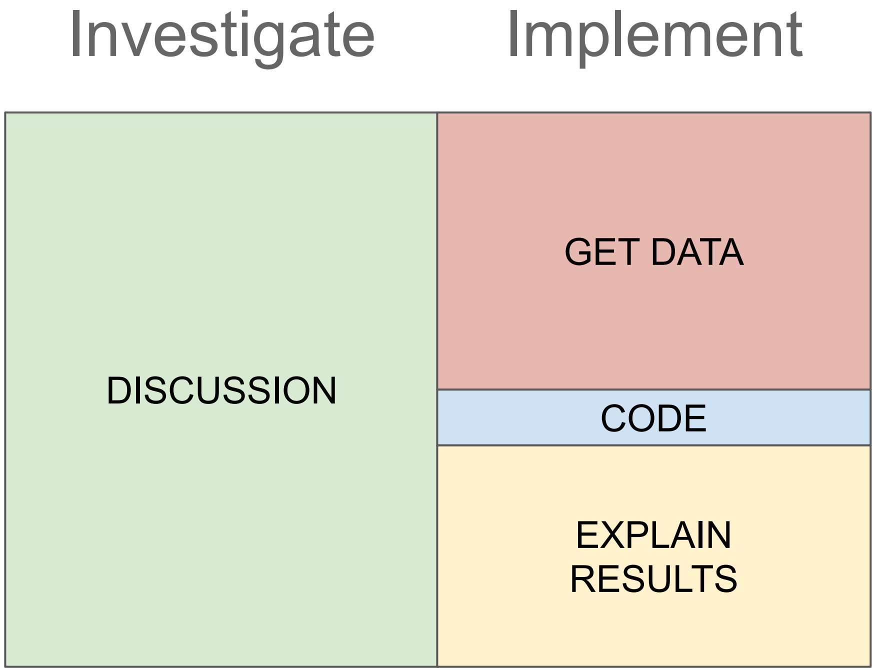
Monitor

Maintain

forever :-)

Add features

Time spent



Investigate

Implement

DISCUSSION

GET DATA

CODE

EXPLAIN RESULTS

Thank you!

CO@Work 2020, Pawel Lichocki, 25.09.2020

https://developers.google.com/optimization