THE UNIVERSITY *of* EDINBURGH

A multi-objective optimisation algorithm for locating humanitarian facilities

Kit Searle*[a]

[a]School of Mathematics and Maxwell Institute for Mathematical Sciences, University of Edinburgh, United Kingdom

19 May 2025

Introduction

- MSF operates in rural areas
- Significant challenges in accessibility
- Individuals often have to travel hours or days to reach essential infrastructure
- Where to build more facilities to maximise impact?

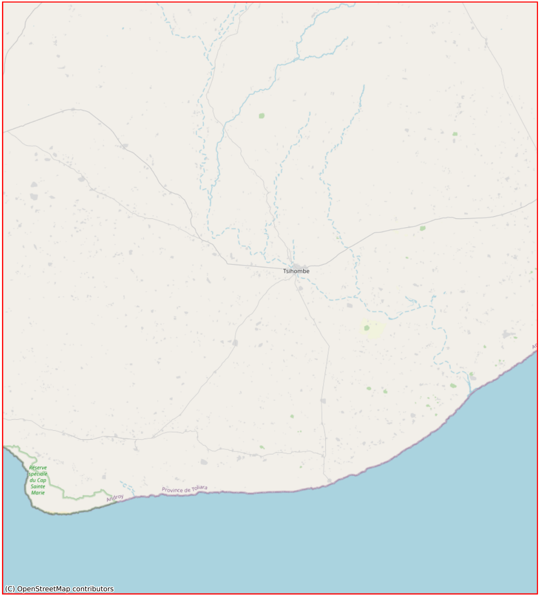
Introduction

- MSF operates in rural areas
- Significant challenges in accessibility
- Individuals often have to travel hours or days to reach essential infrastructure
- Where to build more facilities to maximise impact?

Introduction

- MSF operates in rural areas
- Significant challenges in accessibility
- Individuals often have to travel hours or days to reach essential infrastructure
- Where to build more facilities to maximise impact?

Introduction




Tiles (C) Esri -- Source: Esri, i-cubed, USDA, USGS, AEX, GeoEye, Getmapping, Aerogrid, IGN, IGP, UPR-EGP, and the GIS User Community

Introduction

Agenda

- Project progress
- Problem characteristics
- Solution approach
- Computational results
- Next steps

Project progress

Work package 1: Understand MSF accessibility tool

Work package 2: Develop offline optimisation algorithm

Work package 3: Integrate algorithm into MSF accessibility tool

Project progress

Work package 1: Understand MSF accessibility tool

- MSF model returns accessibility time for a given point in space
- Can easily be mapped to a reduction in accessibility hours for a new candidate location
- Input is therefore a set of geotiff objects, one for each candidate location

Work package 2: Develop offline optimisation algorithm

Work package 3: Integrate algorithm into MSF accessibility tool

Project progress

Work package 1: Understand MSF accessibility tool

- MSF model returns accessibility time for a given point in space
- Can easily be mapped to a reduction in accessibility hours for a new candidate location
- Input is therefore a set of geotiff objects, one for each candidate location

Work package 2: Develop offline optimisation algorithm

- We need fast solutions
- Evolutionary search algorithm
- Multiple objectives

Work package 3: Integrate algorithm into MSF accessibility tool

Project progress

Work package 1: Understand MSF accessibility tool

- MSF model returns accessibility time for a given point in space
- Can easily be mapped to a reduction in accessibility hours for a new candidate location
- Input is therefore a set of geotiff objects, one for each candidate location

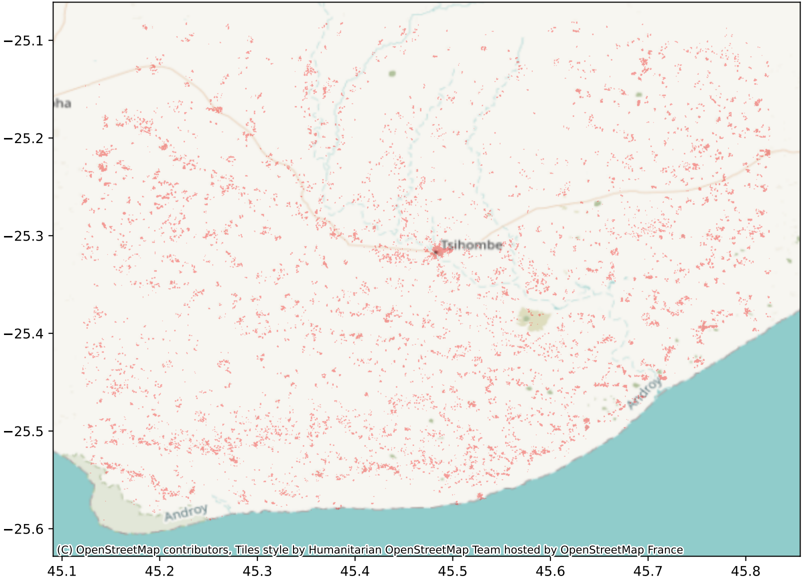
Work package 2: Develop offline optimisation algorithm

- We need fast solutions
- Evolutionary search algorithm
- Multiple objectives

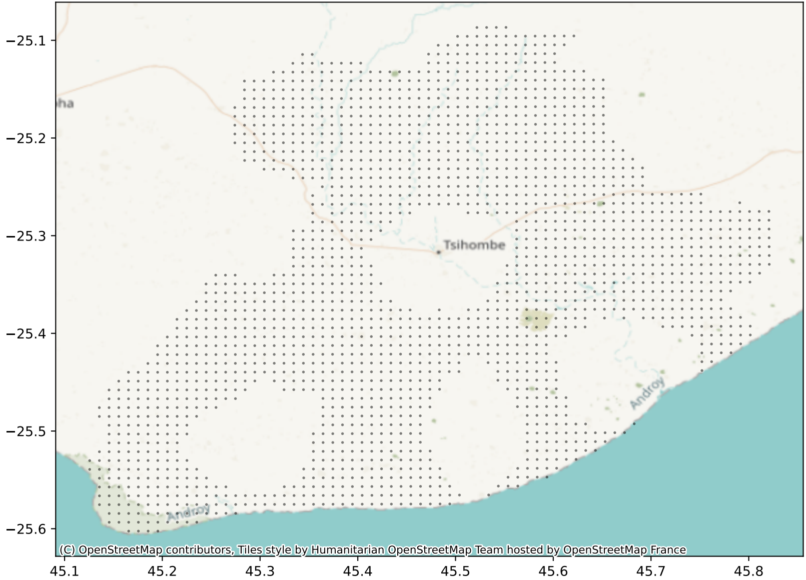
Work package 3: Integrate algorithm into MSF accessibility tool

- Still to come

Problem characteristics



Problem characteristics



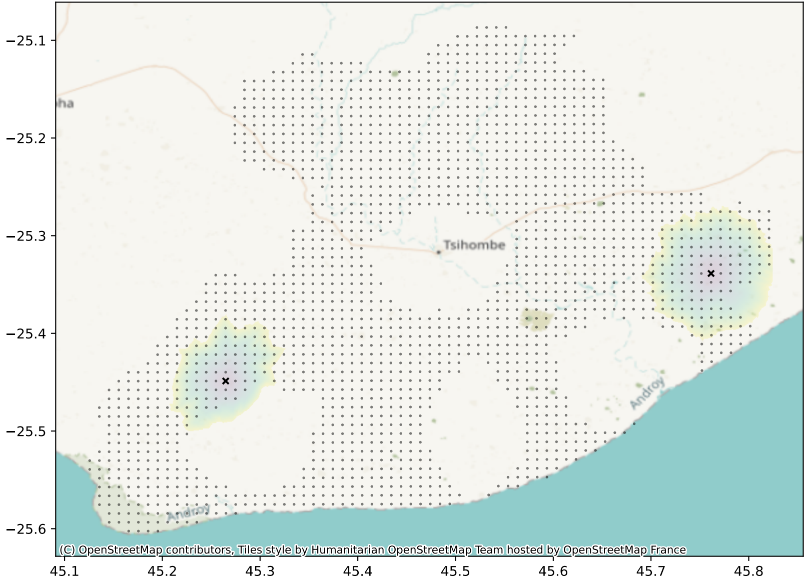
Problem characteristics

Problem definition

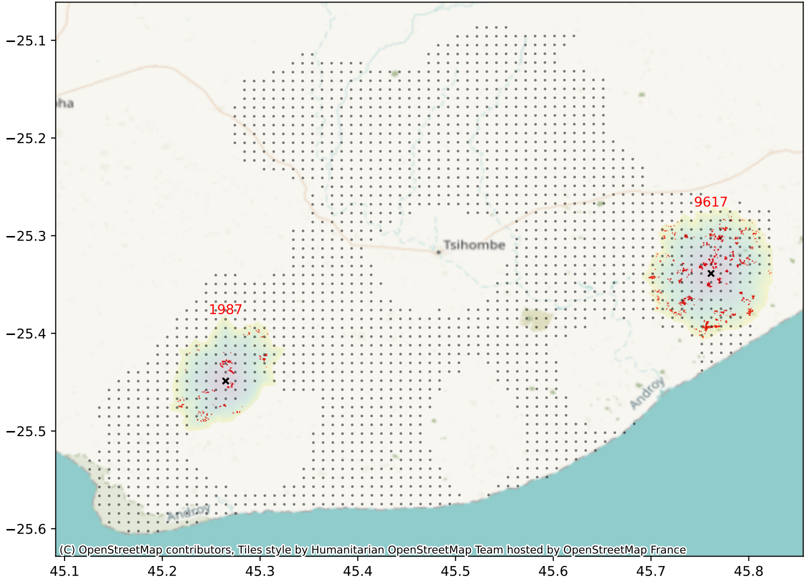
Build between $n_{\min}$ and $n_{\max}$ facilities such that

- Maximise the reduction in access hours for all dwellings
- Maximise the number of covered communities
- Minimise the number of facilities

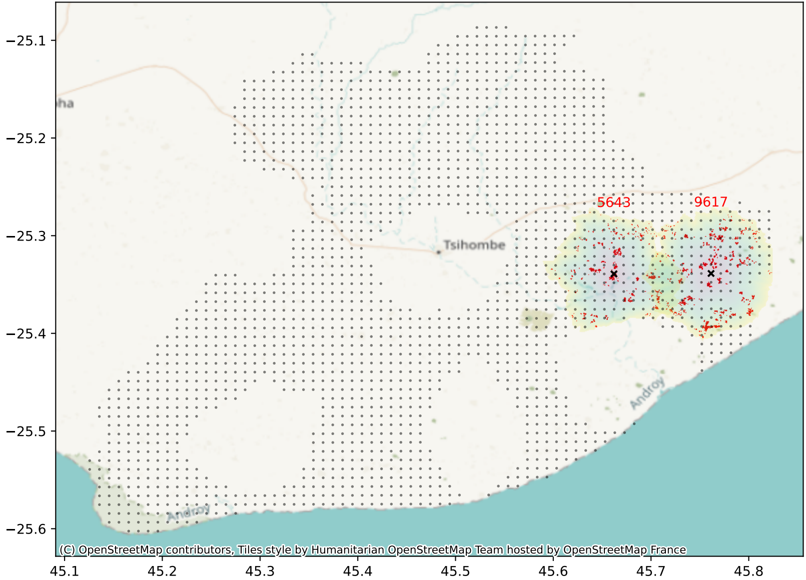
Objective function 1



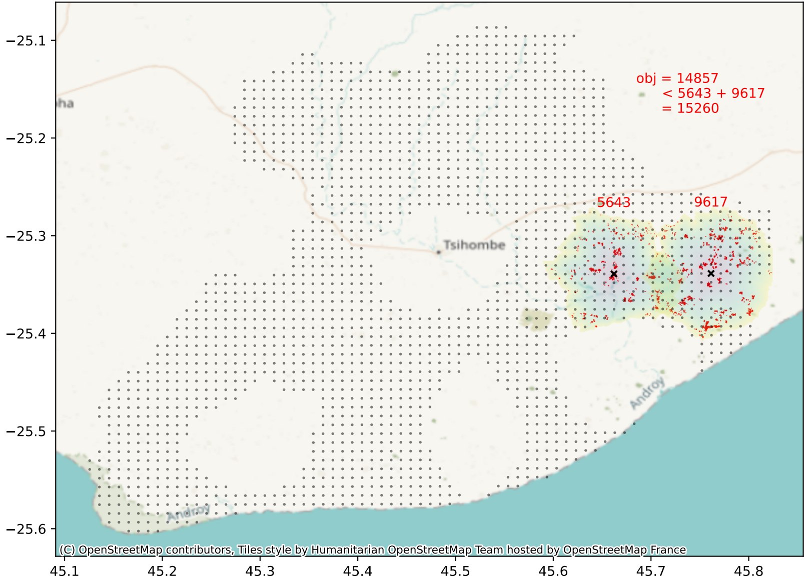
Objective function 1



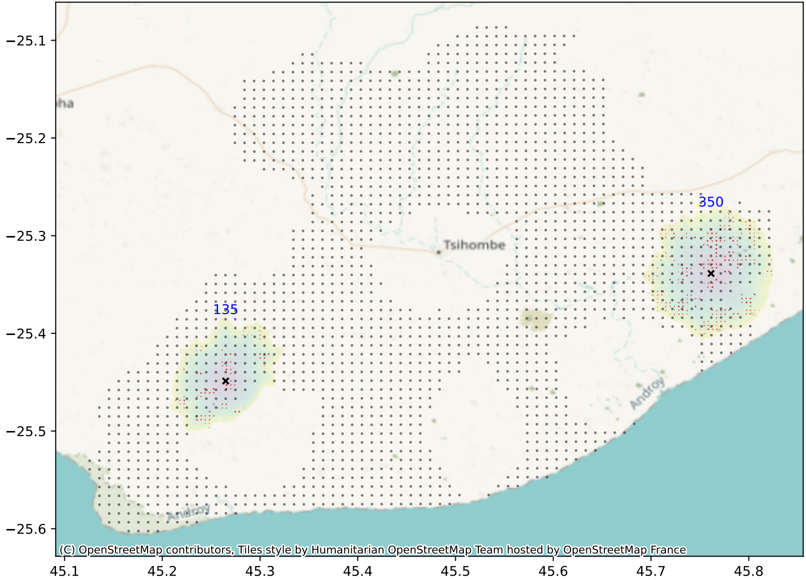
Objective function 1



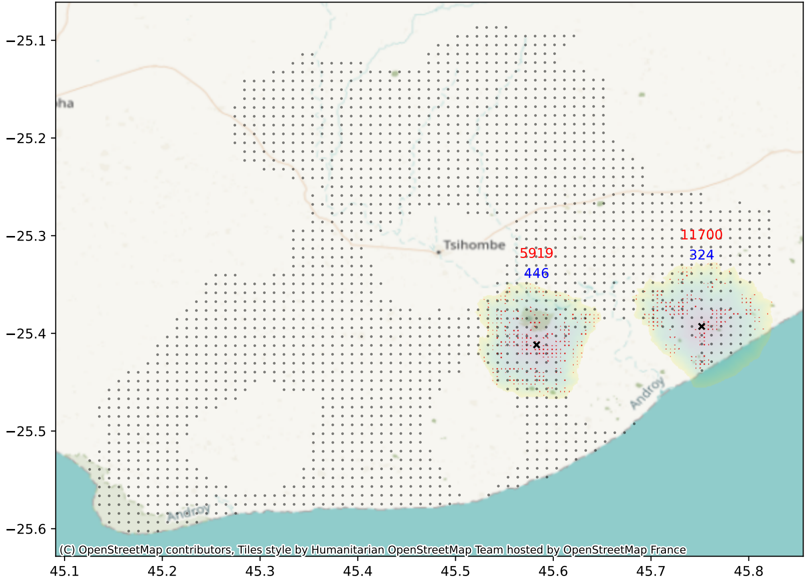
Objective function 1



Objective function 2



Objective function 2



Pre-requisites: Multi-objective optimisation

A general optimisation model

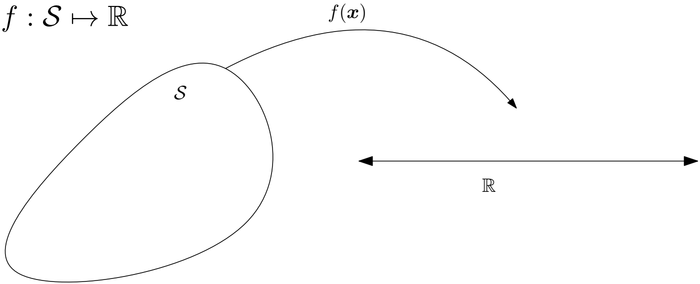
$$\begin{aligned} \min \quad & z = f(\boldsymbol{x}) \\ \text{s.t.} \quad & \boldsymbol{x} \in \mathcal{S} \end{aligned}$$

- $\boldsymbol{x} = [x_1, \dots, x_n]^T$ are our decision variables or decision vector
- $\mathcal{S}$ is our feasible region
- $f : \mathcal{S} \mapsto \mathbb{R}$

A general optimisation model

$$\begin{aligned} \min \quad & z = f(\boldsymbol{x}) \\ \text{s.t.} \quad & \boldsymbol{x} \in \mathcal{S} \end{aligned}$$

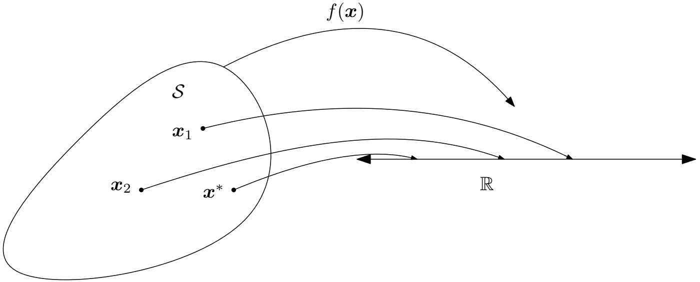
- $\boldsymbol{x} = [x_1, \ldots, x_n]^T$ are our decision variables or decision vector
- $\mathcal{S}$ is our feasible region
- $f : \mathcal{S} \mapsto \mathbb{R}$



Pre-requisites: Multi-objective optimisation

A global solution

A vector $\boldsymbol{x}^* \in \mathcal{S}$ such that $z^* = f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})\ \forall \boldsymbol{x} \in \mathcal{S}$



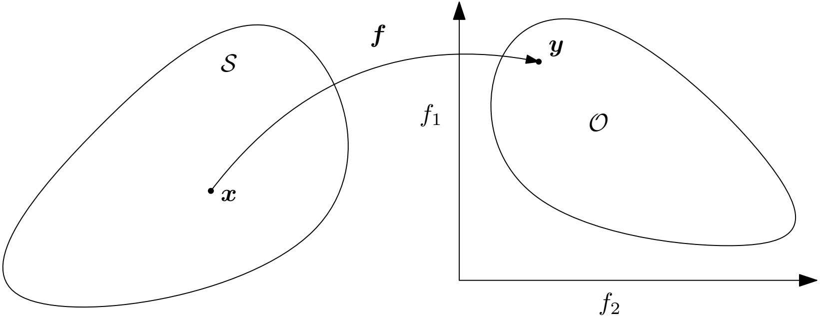
Pre-requisites: Multi-objective optimisation

A general multi-objective optimisation problem

$$\begin{aligned} \min \quad & \boldsymbol{f}(\boldsymbol{x}) \\ \text{s.t.} \quad & \boldsymbol{x} \in \mathcal{S} \end{aligned}$$

- $\boldsymbol{x} = [x_1, \ldots, x_n]^T$ are our decision variables or decision vector
- $\boldsymbol{f}(\boldsymbol{x}) = [f_1(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x})]$ is the vector of objectives
- $\mathcal{S}$ is our feasible region or decision space
- $\boldsymbol{f} : \mathcal{S} \mapsto \mathcal{O}$
- $\mathcal{O}$ is our objective space
- For simplicity let $\boldsymbol{y} = [y_1, \ldots, y_k]^T$, where $y_i = f_i(\boldsymbol{x})$

Pre-requisites: Multi-objective optimisation



Decision space

Objective space

Pre-requisites: Multi-objective optimisation

Conflicting objectives

Pre-requisites: Multi-objective optimisation

Conflicting objectives

- In single objective optimisation we would have a single solution (or infinitely many but all with the same objective function value)

Pre-requisites: Multi-objective optimisation

Conflicting objectives

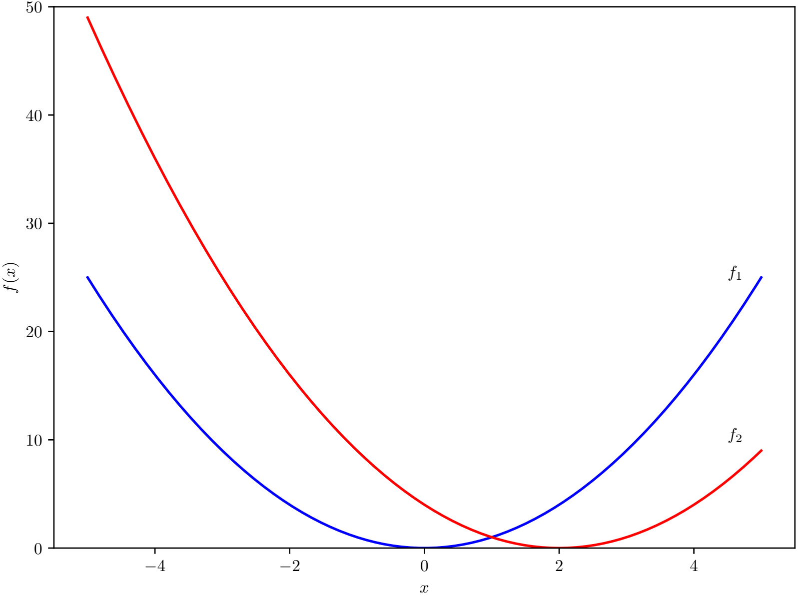
- In single objective optimisation we would have a single solution (or infinitely many but all with the same objective function value)
- Conflicting nature of objectives functions

Pre-requisites: Multi-objective optimisation

Conflicting objectives

- In single objective optimisation we would have a single solution (or infinitely many but all with the same objective function value)
- Conflicting nature of objectives functions
- The improvement in one objective function results in the degradation in another

Pre-requisites: Multi-objective optimisation



Pre-requisites: Multi-objective optimisation

How do we say if one feasible solution is better than another?

Pre-requisites: Multi-objective optimisation

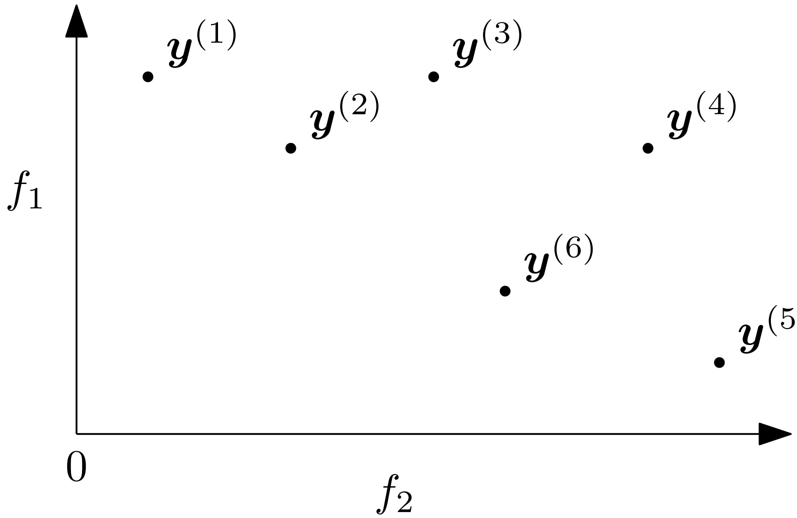
How do we say if one feasible solution is better than another?

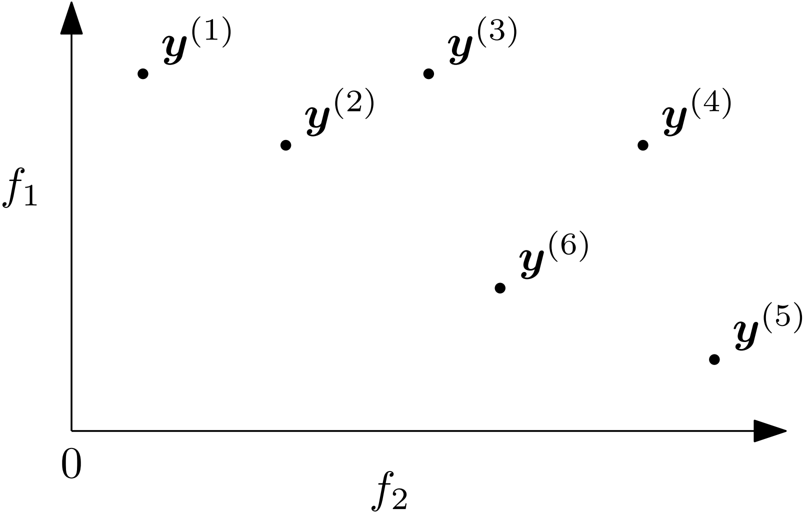
Definition 1 (Dominance)

An objective vector $\boldsymbol{y}^{(1)} = \boldsymbol{f}(\boldsymbol{x}^{(1)})$ to a multi-objective optimisation problem is said to dominate another objective vector $\boldsymbol{y}^{(2)} = \boldsymbol{f}(\boldsymbol{x}^{(2)})$ if and only if

1. $\boldsymbol{y}^{(1)}$ is no worse than $\boldsymbol{y}^{(2)}$ in all objective functions, and
2. $\boldsymbol{y}^{(1)}$ is at least strictly better than $\boldsymbol{y}^{(2)}$ in at least one objective function

then we say that $\boldsymbol{y}^{(1)} \prec \boldsymbol{y}^{(2)}$


$\boldsymbol{y}^{(1)}$
$\boldsymbol{y}^{(3)}$
$\boldsymbol{y}^{(2)}$
$\boldsymbol{y}^{(4)}$
f_1
$\boldsymbol{y}^{(6)}$
$\boldsymbol{y}^{(5)}$
0
f_2

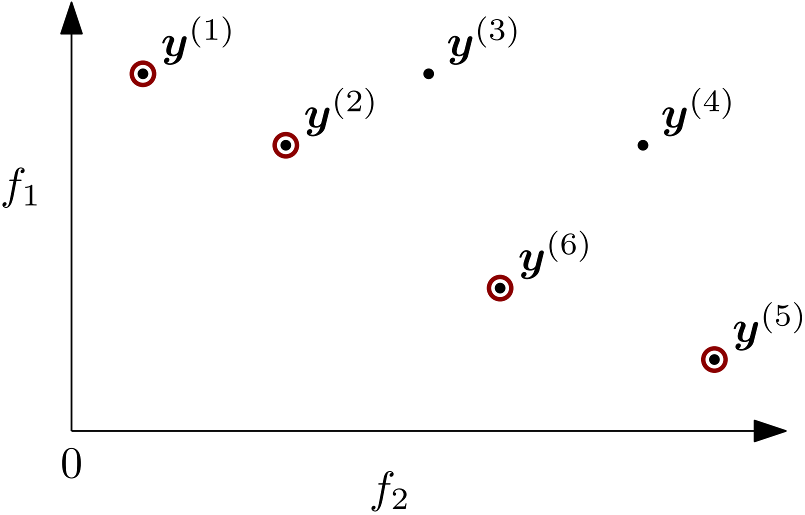


$$\boldsymbol{y}^{(1)} \prec \boldsymbol{y}^{(3)},\ \boldsymbol{y}^{(1)} \nprec \boldsymbol{y}^{(2)},\ \boldsymbol{y}^{(1)} \nprec \boldsymbol{y}^{(4)},\ \ldots$$

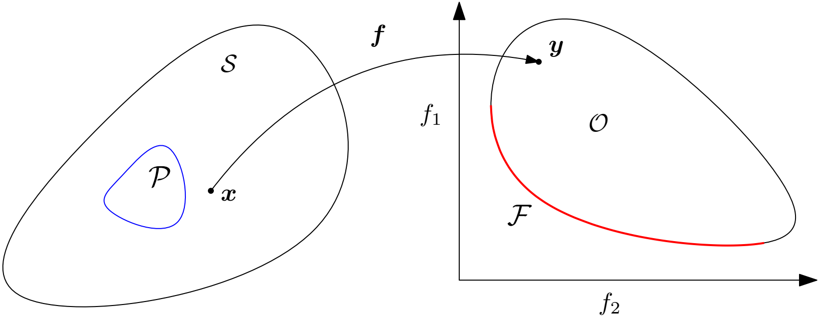
Pre-requisites: Multi-objective optimisation

Definition 2 (Pareto optimality)

A solution $\boldsymbol{x}^* \in \mathcal{S}$ is Pareto optimal if it's corresponding objective vector $\boldsymbol{y}^* = \boldsymbol{f}(\boldsymbol{x}^*)$ not dominated by any other solution vector $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})\ \forall \boldsymbol{x} \in \mathcal{S}$.


$y^{(1)}$
$y^{(2)}$
$y^{(3)}$
$y^{(4)}$
$y^{(6)}$
$y^{(5)}$
f_1
f_2
0

Multi-objective optimisation



Decision space

Objective space

Pre-requisites: NSGA II

Overview

- Genetic Algorithms (GAs) are search heuristics inspired by the process of natural selection
- Used to solve multi objective optimization problems
- Key concepts include: chromosome, population, non dominated rank, crowding distance, selection, crossover, and hypervolume

Pre-requisites: NSGA II

Chromosomes

- A chromosome is a solution to the problem
- Represented as a fixed-length vector of values

5	10	6	1	3	4

Pre-requisites: NSGA II

Population

- A population is a set of chromosomes
- Each element is a solution to the problem

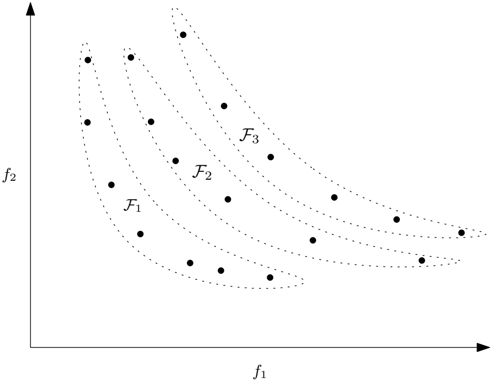
5	10	6	1	3	4
6	12	4	15	9	1
4	2	8	16	11	7
3	9	17	15	19	2

⋮

Pre-requisites: NSGA II

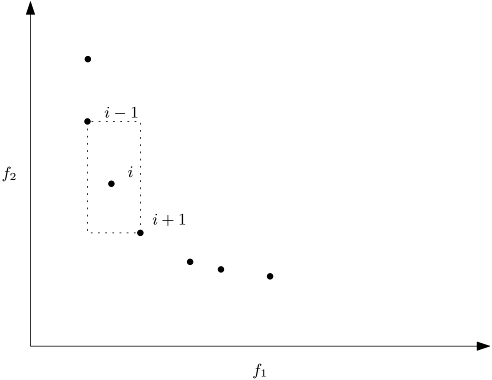
Non-dominated rank

- Determines "which" Pareto front a chromosome is in
- Used to measure solution quality



Crowding distance

- Determine how close solutions are to a given chromosome
- Only consider chromosomes with equal non-dominated rank



Pre-requisites: NSGA II

Selection

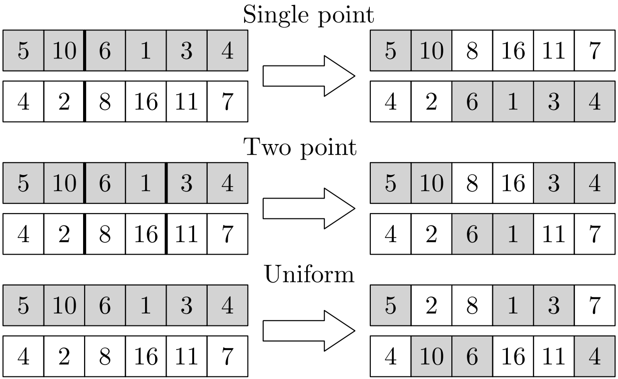
- Choose two individuals to reproduce
- Usually based on non-dominated rank or crowding distance

5	10	6	1	3	4
6	12	4	15	9	1
4	2	8	16	11	7
3	9	17	15	19	2

⋮

Crossover

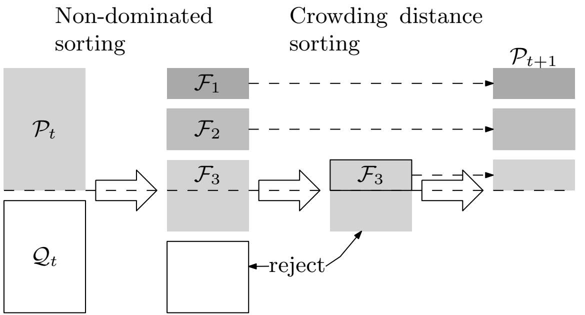
- Crossover combines two parents to produce new offspring
- Common types: single-point, multi-point, uniform crossover



Pre-requisites: NSGA II

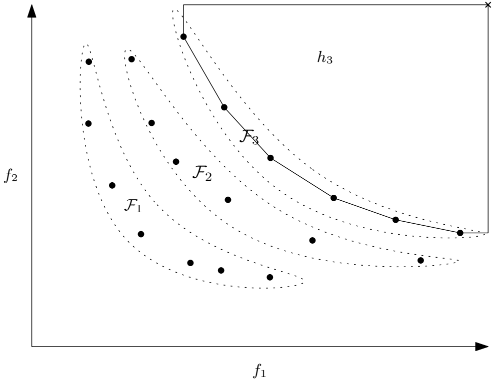
Outline

1. Initialise population $\mathcal{P}_t$
2. Create new temporary population $\mathcal{Q}_t$
3. Perform non dominated sorting
4. Update based on non-dominated rank on crowding distance
5. Update population $\mathcal{P}_{t+1}$; return to 2



Algorithim performance: Hypervolume

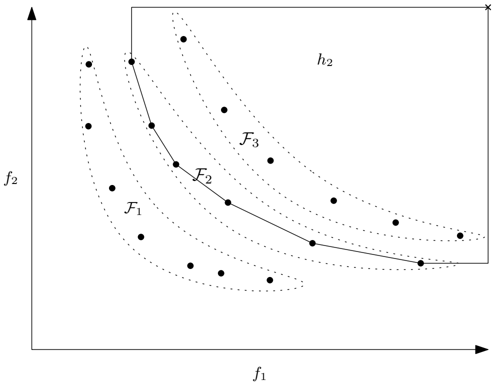
- The are in objective dominated by the Pareto Front
- Computed with respect to a reference point



Pre-requisites: NSGA II

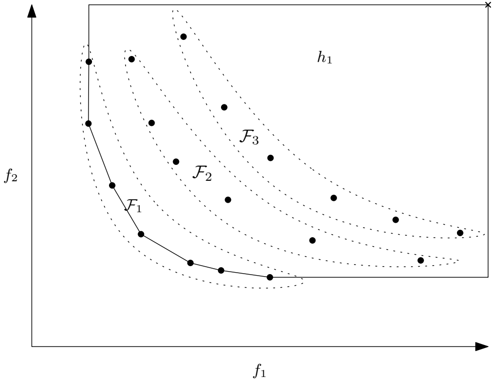
Algorithim performance: Hypervolume

- The are in objective dominated by the Pareto Front
- Computed with respect to a reference point



Algorithim performance: Hypervolume

- The are in objective dominated by the Pareto Front
- Computed with respect to a reference point



Solution approach

Genetic algorithm

- Solved at multiple resolutions
- Variable length chromosomes
  - $\boldsymbol{s}_1 = [100,\ 250,\ 650]$ $\boldsymbol{s}_2 = [580,\ 360,\ 1,\ 200]$
- Additional local search operators

Solution approach

Outline

1. Fetch candidates at lowest resolution
2. Create population at current resolution
3. NSGA II
4. Local search
5. Increase resolution; return to 2

Solution approach

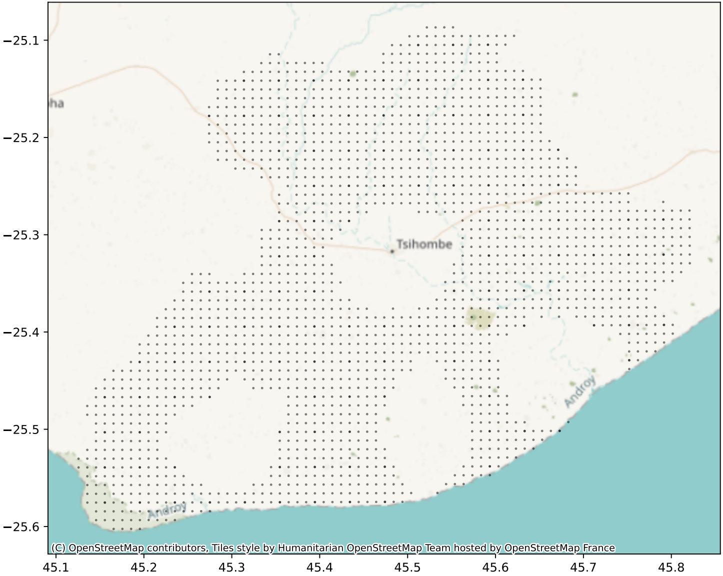
Creating the population

- If the population is the empty set randomly generate arrays
- If we already have a population carry forward the first k fronts, randomly generate the rest

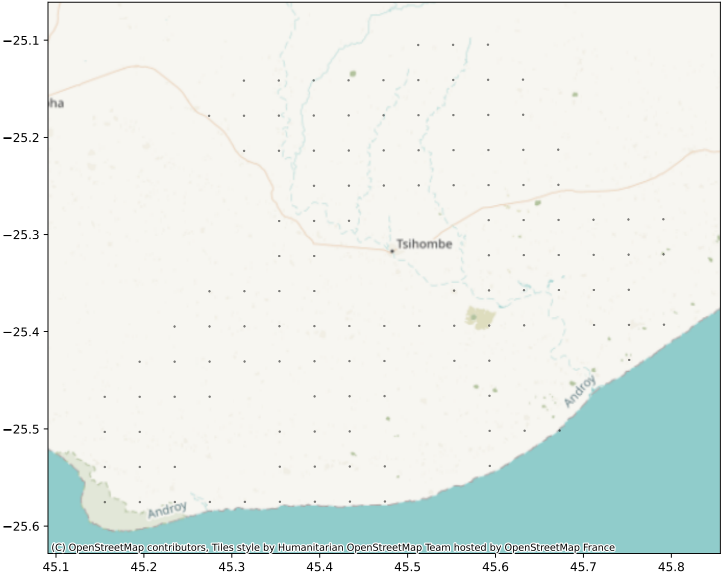
Local search

- A 1-1 interchange
- For every solution on the front
  - Switch out a near by facility
  - if the new solution is non-dominated add it to the population

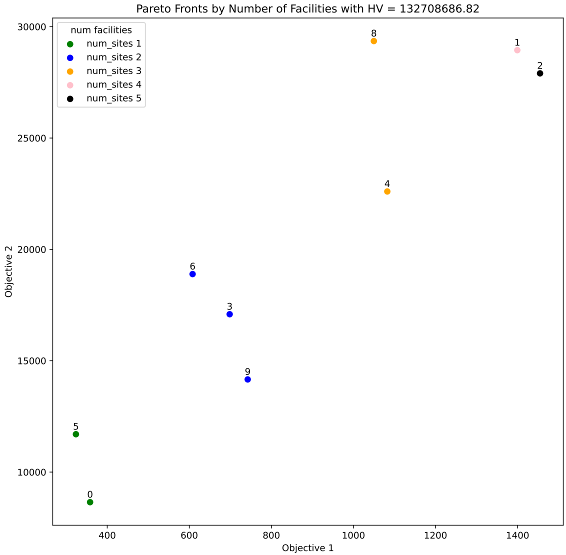
Solution approach: Example



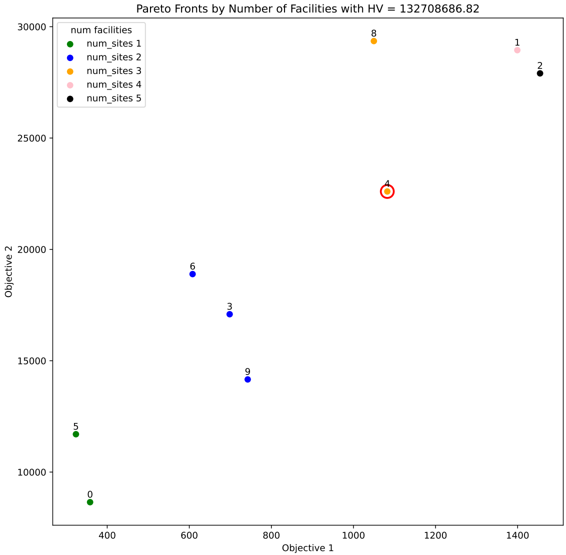
Solution approach: Example



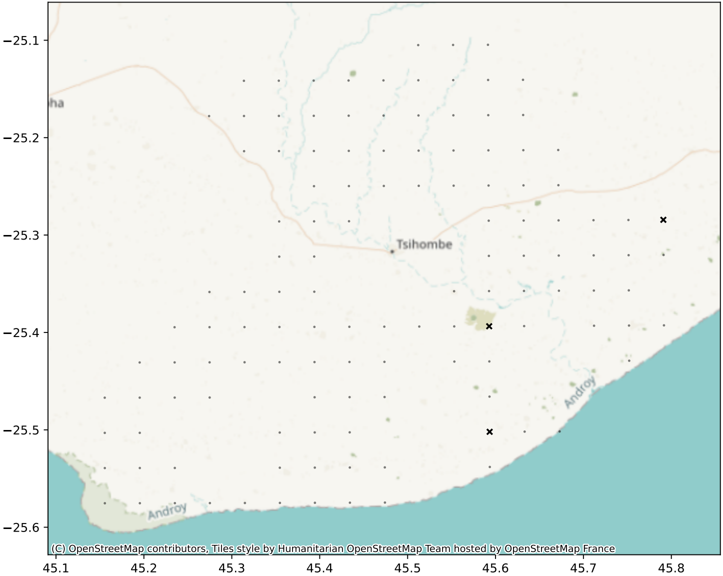
Solution approach: Example



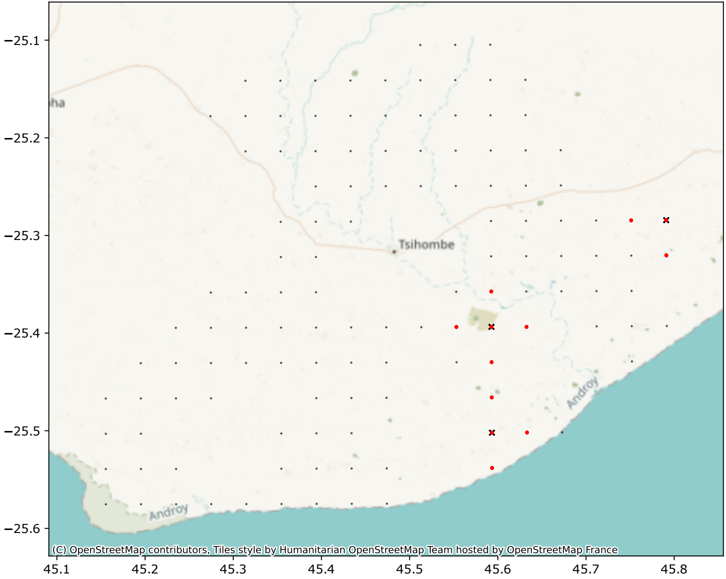
Solution approach: Example



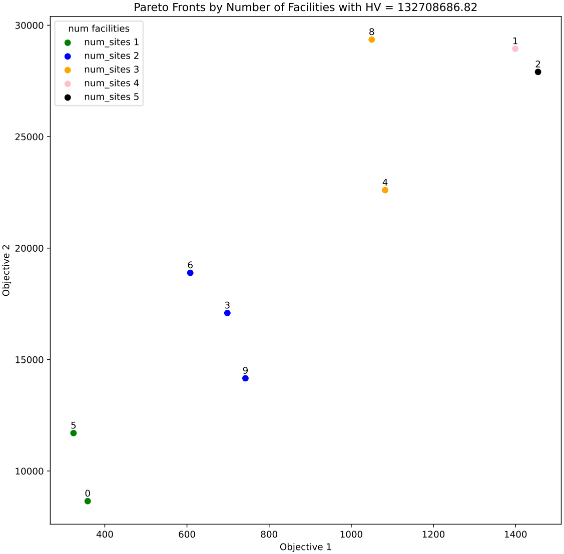
Solution approach: Example



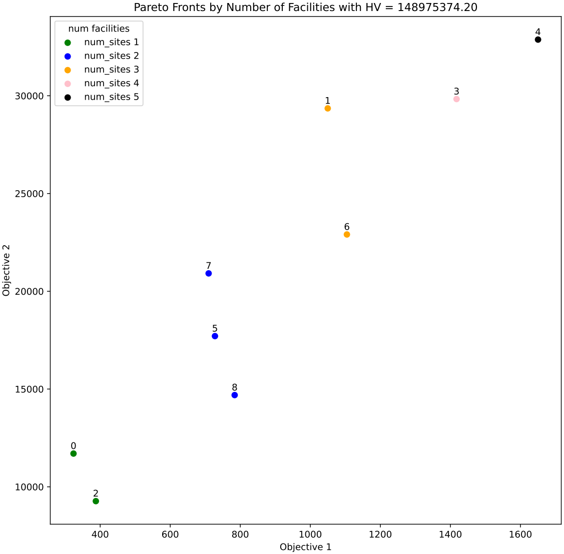
Solution approach: Example



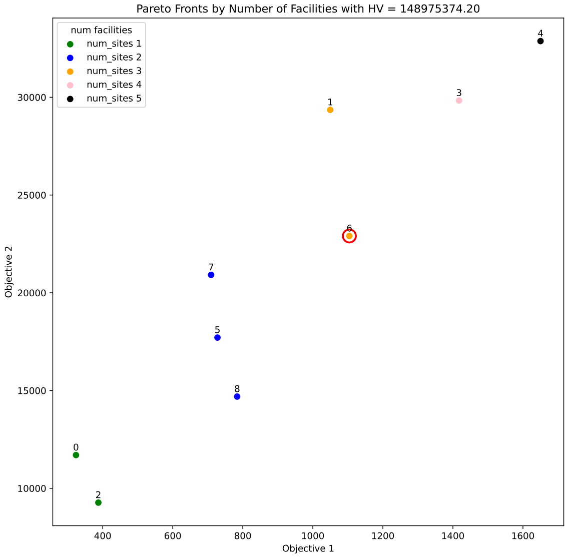
Solution approach: Example



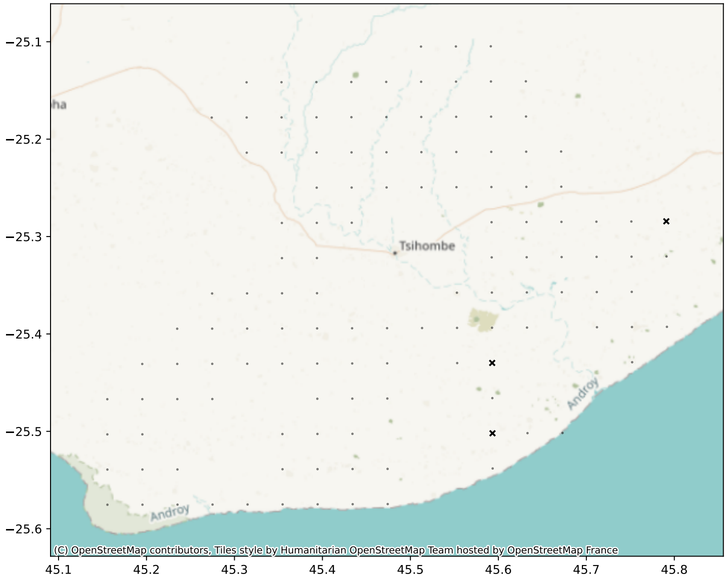
Solution approach: Example



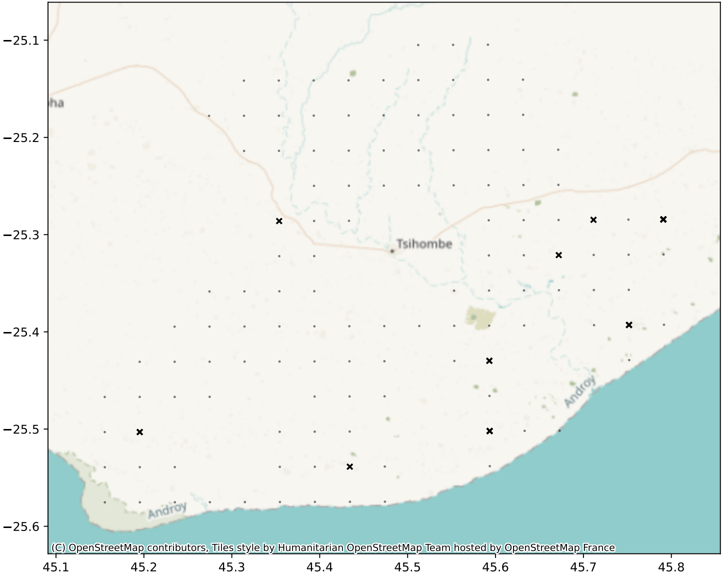
Solution approach: Example



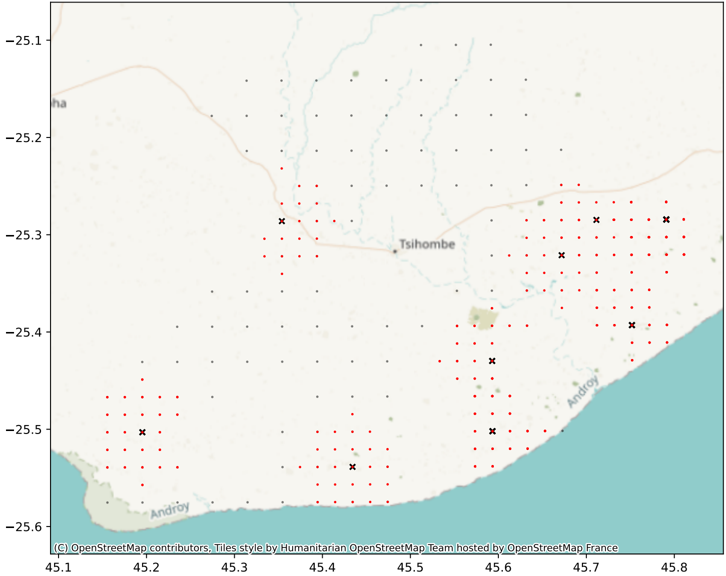
Solution approach: Example



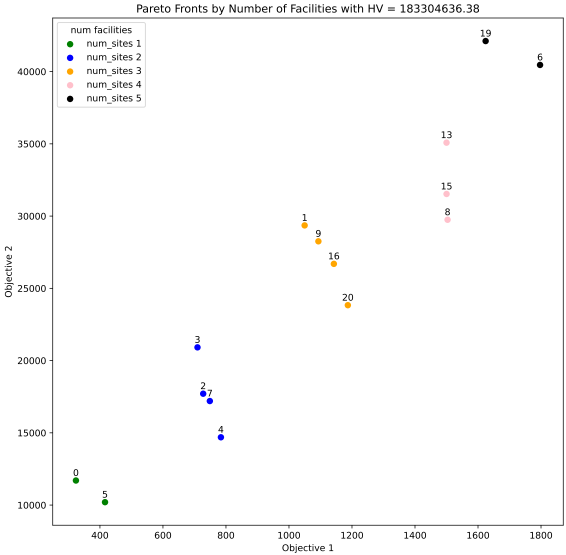
Solution approach: Example



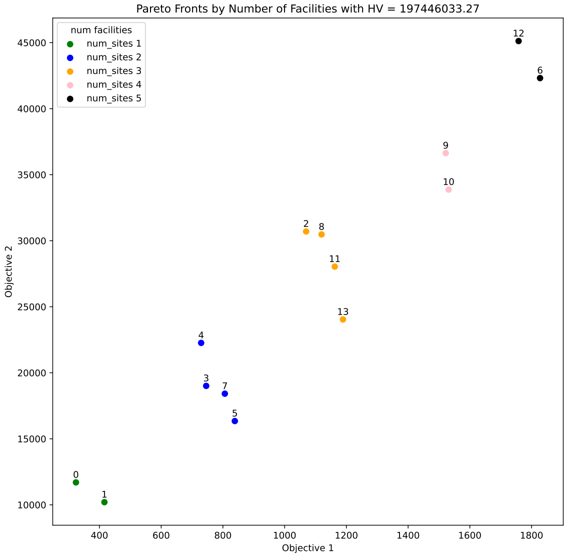
Solution approach: Example



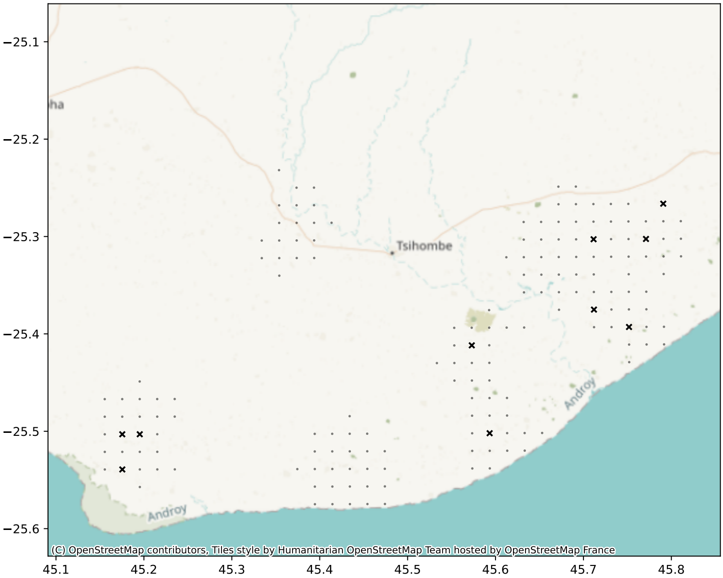
Solution approach: Example



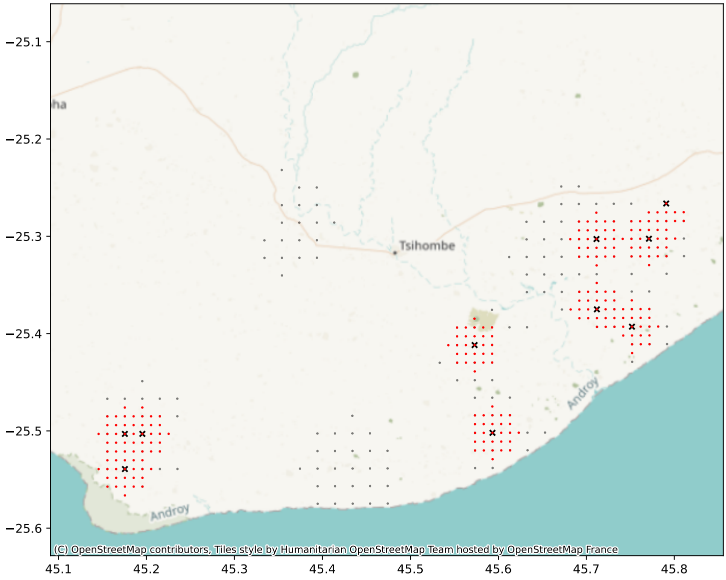
Solution approach: Example



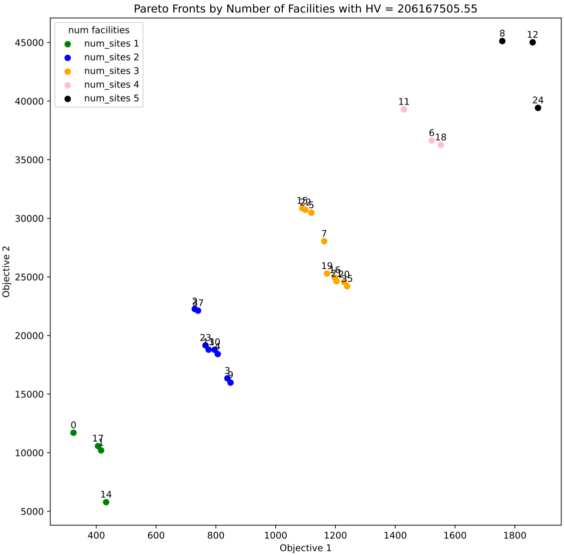
Solution approach: Example



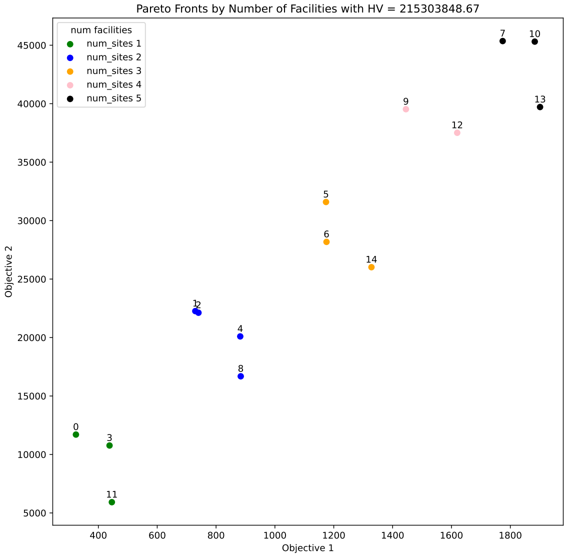
Solution approach: Example



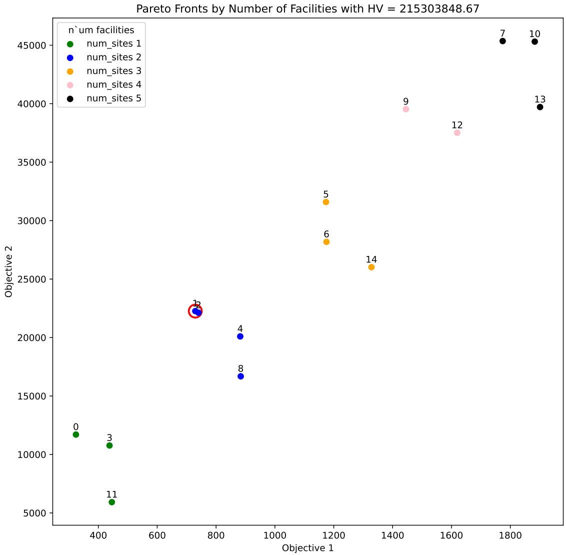
Solution approach: Example



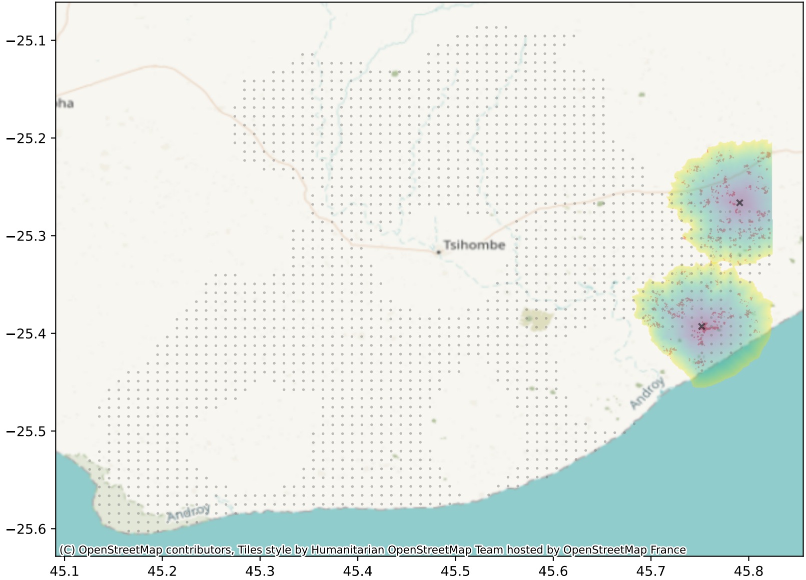
Solution approach: Example



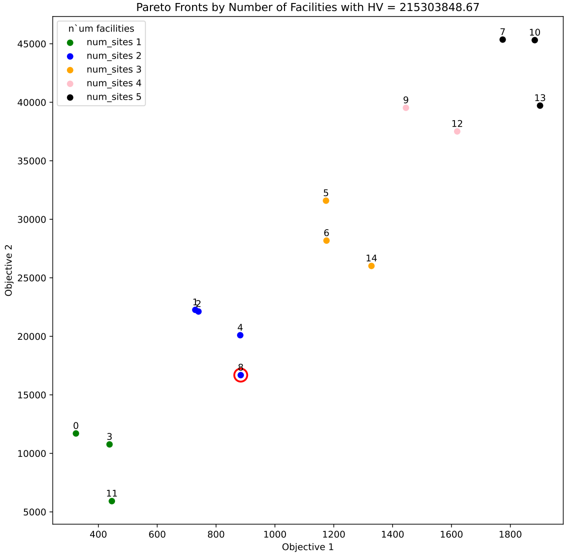
Solution approach: Example



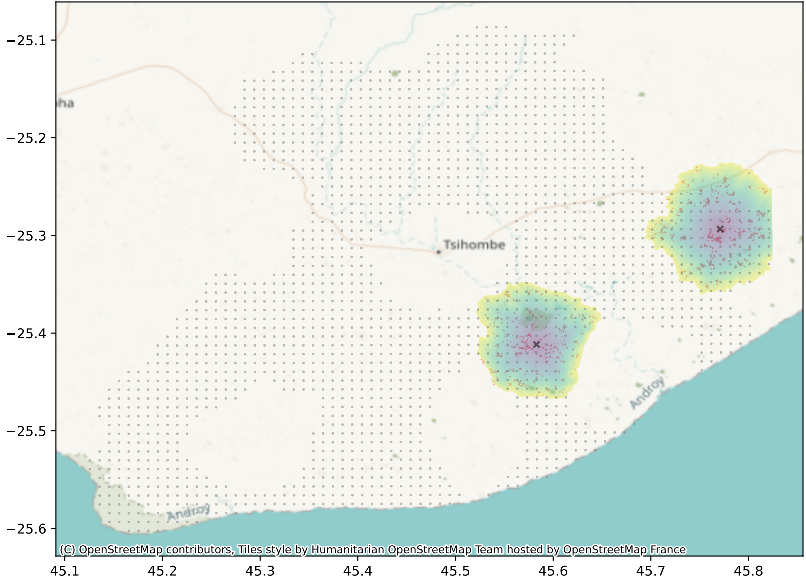
Solution approach: Example



Solution approach: Example



Solution approach: Example



Computational results

		full				no local search				one resolution			
		time		hv		time		hv		time		hv	
pop	gen	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
25	5	453	70	184	31	279	16	175	9	–	–	–	–
	10	605	95	210	9	521	16	192	7	–	–	–	–
	15	771	58	206	15	793	39	199	5	–	–	–	–
50	5	706	93	217	6	526	8	189	7	488	87	192	14
	10	886	88	217	6	1 055	15	203	3	881	130	209	2
	15	1 271	259	224	4	1 577	40	209	3	913	88	206	10
100	5	839	42	211	9	802	22	198	10	655	61	206	12
	10	1 934	520	222	4	1 588	38	211	4	1 158	81	213	10
	15	2 583	811	223	3	2 871	1 076	212	5	1 688	93	223	4

What's next?

- Include some parallelisation
- Some more local search operators
    - Split and merge
- Update the mutation operator
- Integrate the algorithm into the accessibility tool