Large-scale multi-objective optimisation for sustainable waste management using Evolutionary Algorithms

A meta-heuristic approach to sustainable waste management

Master's Thesis submitted to the
Faculty of Informatics of the *Università della Svizzera Italiana*
in partial fulfillment of the requirements for the degree of
Master of Science in Informatics
Artificial Intelligence

presented by
Felix Carl Rouben Edzard Linus Boelter

under the supervision of
Prof. Luca Gambardella
co-supervised by
Umberto Jr. Mele

September 2022

I certify that except where due acknowledgement has been given, the work presented in this thesis is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; and the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program.

Felix Carl Rouben Edzard Linus Boelter
Lugano, September 19, 2022 September 2022

“The line between order and disorder lies in logistics.”

Sun Tzu

Abstract

By 2050, our waste output will be approximately 3.4 billion tons. This waste generation will have detrimental effects on our planet and every inhabitant of it. Furthermore, recycling rates in developing countries are falling to all-time lows. Inefficiently planned waste management systems that address these issues burden the environment, economy and society. Researchers propose waste-management supply chain methods to solve these issues by transforming the planning problem into a multi-objective mixed integer linear programming (MILP) problem. The approach to solving this problem utilises methods that find an optimised solution using exponential-time increasing algorithms dependent on the input size. In addition, multi-objective problems create trade-offs between the different objectives, inducing a set of promising solutions called a Pareto front, which the current approaches do not exploit.
This Thesis proposes a method that aims to improve the current solution methods by incorporating multi-objective evolutionary algorithms, which decrease the time taken for large-scale problems and obtain a set of solutions, each with different trade-offs across the objectives, resulting in diverse solutions. Finally, we convey an improvement to the current methods by differentially displaying the results of the current methods and our method.

Contents

Figures

Tables

List of Algorithms

Chapter 1

Background & Motivation

Global warming is the most pressing threat humanity will face in our lifetime. We currently produce 700 million tons of waste OECD [2019]. Waste production will only grow. By 2050, our waste output will equal 3.4 billion tons, a 70 per cent increase compared to today. Tiseo [2022]
How do governments manage all that waste? Germany is considered best-in-class for recycling, with a 67.1 per cent recycling rate. South Korea, Austria and the Netherlands follow in a head-to-head recycling race Statista [2022]. However, not every country manages to hit these targets. E.g. the United States only works to recycle around 5 per cent of its plastic waste Volcovici [2022]. The main challenge lies in developing nations, whose total recycling rates are often under 5 per cent. Statista [2022]

It is essential to develop sophisticated recycling systems around the world. Otherwise, it could lead to dire consequences. One of the most well-known consequences is called microplastics. These are tiny plastic particles that result from the breakdown of larger plastics. These microscopic plastics wreak havoc on the environment and can harm animals and seedlings National-Geographic-Society [2022]. Moreover, recycling may create an economic opportunity to reuse materials rather than produce them again. Furthermore, recycling leads to a reduction in energy usage and may eliminate resource scarcity.

This Thesis aims to investigate waste management optimisation research and build on the latest findings in the field. In more detail, we will optimise the location and size of sorting, incinerator and landfill facilities. This optimisation depends on parameters such as distance, cost, land usage, the amount of municipal solid waste, and disability-adjusted life years (DALY), which is a metric to measure the number of years lost due to death of disease. Murray [1994] Using these parameters, we optimise in a multi-objective manner by using evolutionary algorithms to obtain a Pareto front of solutions. Furthermore, we use a baseline to differentially convey our experiments and

the difference between the two approaches.

The following sections will introduce fundamental concepts and definitions utilised in the subsequent chapters. We will then give an overview of the literature reviewed, including waste management optimisation, the exact solution algorithms, and the evolutionary algorithms we will use in this Thesis. After, we introduce the methodology section, which gives insight into the methods we used to obtain the results to validate this Thesis's claims. In chapter 4, we will give information on the implementation of the baseline model and our model. We then display the empirical evidence to support our claims and briefly discuss the results. This Thesis concludes by indicating our approach's limitations and suggesting future work.

1.1 Definitions

The following section will detail the proposed definitions for the main concepts investigated in this thesis to state all further claims and research as transparently and accurately as possible.

1.1.1 Linear Programming (LP)

Linear programming is an optimisation technique that maximises or minimises a linear objective function subject to linear equality and linear inequality constraints. A canonical form expresses a linear program as,

$$\begin{array}{ll} \text{Find a vector} & \mathbf{x} \\ \text{that maximises} & \mathbf{c^T x} \\ \text{subject to} & \mathbf{Ax} \leq \mathbf{b} \\ \text{and} & \mathbf{x} \geq \mathbf{0} \end{array}$$

1.1.2 Mixed-Integer Linear Program (MILP)

A mixed-integer linear program (MILP) is a problem with a linear objective function, f. Bounds and strictly linear constraints, with some variables of x requiring to be integers, while other variables are non-integers. Furthermore, the MILP problem is an NP-Complete problem, meaning that no efficient solution algorithm yet exists that can solve the problem in polynomially-increasing time based on the number of inputs. The canonical form of a MILP problem is,

$$\begin{aligned} \min \quad & \mathbf{c^T x} \\ \text{s.t.} \quad & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \\ & \mathbf{x_i} \in \mathbb{Z}, \quad \forall i \in \mathcal{I} \end{aligned}$$

1.1.3 Single Objective

We define a general single objective optimization problem as minimizing or maximizing a function subject to constraints which are less than or equal to 0. Given a function f, the best value $f(x^*)$ is a global minimum if and only if it is less than or equal to any other $f(x)$. The goal of finding the minimum solution is called a global optimization problem for single objective problems. Bäck [1996]

1.1.4 Multi-objective

Multi-objective optimisation problems use decision variables. These are vectors consisting of chosen numerical values in an optimisation problem, denoted as x_j, where $j = \{1, .., n\}$. The multi-objective optimization problem uses these decision variables to optimise multiple objective functions simultaneously. The term "optimise" means finding a solution with a trade-off between the objective functions. Coello Coello et al. [2007] writes that “finding the global optimum of a general MOP problem is an NP-complete problem” [pp. 7–8].

A MOP consists of k objective functions, $m + p$ total constraints on the objective functions and n decision variables. The k objective functions with $m + p$ constraints can be linear or non-linear and continuous or discrete. Coello Coello et al. [2007] defines a general MOP as a “minimisation (or maximisation), of the components of a vector $F(x)$ where $F(x) = (f_1(x), ..., f_k(x))$ and x is a n-dimensional vector, subject to constraints $g_i(x) \leq 0$, where $i = 1, ..., m$ and $h_j(x) = 0$, where $j = 1, ..., p$” [pp. 7–8].

1.1.5 Pareto Terminology

When a problem handles several objective functions to optimise the notion of "optimum" changes, the aim shifts from finding a single optimal solution to finding suitable compromises between the objective functions called the Pareto Optimum.
The vector $\mathbf{x}^* \in \Omega$ is said to be Pareto optimal if there exists no other feasible vector $\mathbf{x} \in \Omega$ which dominates $\mathbf{x}^* \in \Omega$, where Ω is the feasible region of $\mathbf{x}$. Pareto domination implies a decrease in a criterion which does not cause an increase in at least another criterion [Coello Coello et al., 2007, pp. 10–11].

Definition 1.1.1 (Pareto Dominance (Coello Coello et al. [2007])): *A vector* $\mathbf{u} = (u_1, ..., u_k)$ ***dominates*** *another vector* $\mathbf{v} = (v_1, ..., v_k)$ *(written as* $\mathbf{u} \preceq \mathbf{v}$*), where* k *is the number of objective functions. If and only if* $\mathbf{u}$ *is partially less than* $\mathbf{v}$ *i.e.* $\forall i \in \{1, ..., k\}, u_i \leq v_i \wedge \exists i \in \{1, ..., k\} : u_i < v_i$. □

Definition 1.1.2 (Pareto Optimal Set (Coello Coello et al. [2007])): *For a multi-objective problem,* $F(\mathbf{x})$*, the* ***Pareto Optimal Set****,* $\mathcal{P}^*$*, is*

$$\mathcal{P}^* = \{\mathbf{x} \in \Omega \mid \neg\exists\, \mathbf{x}' \in \Omega : F(\mathbf{x}') \preceq F(\mathbf{x})\}. \tag{1.1}$$

□

Definition 1.1.3 (Pareto Front (Coello Coello et al. [2007])): *Given a multi-objective problem,* $F(\mathbf{x})$*, and a Pareto Optimal Set,* $\mathcal{P}^*$*, let the* ***Pareto Front****,* $\mathcal{PF}^*$*, be*

$$\mathcal{PF}^* = \{\mathbf{u} = F(\mathbf{x}) \mid \mathbf{x} \in \mathcal{P}^*\} \tag{1.2}$$

□

1.1.6 Evolutionary Algorithms

This section defines basic evolutionary algorithm terms and concepts. We define an *individual* or *chromosome* as a solution to some problem. A list of *parameters* or *genes* represents these that take on values (alleles) from some genetic alphabet. The term for a set of individuals is a population. Each individual's parameters of the population are the input for an objective function. Like in nature, Evolutionary Operators (EVOPs) act on the population to obtain solutions with higher fitness [Coello Coello et al., 2007, pp. 24–25]. The three significant EVOPs used in EAs are *mutation*, *crossover*, and *selection*. The mutation operator is used to obtain a new solution by making a minor random modification to a chromosome. It maintains and introduces diversity into the population. Fig. 1.1 shows a common mutation, known as a *bitwise mutation*, where '1' gets changed to '0' and vice-versa depending on a probability applied on every bit. The crossover operator is a process of reproducing new chromosomes from the parent chromosomes. Fig. 1.2 shows a single-point crossover, where each parent is cut and recombined with a piece from each other. The selection operator selects the parents for use in the crossover operator to generate new offspring. Fig. 1.3 shows a method known as tournament selection, which involves running several "tournaments" among a few individuals chosen randomly from the population. The individuals with the highest fitness, which won the tournament, are selected for crossover.

Algorithm 1: Evolutionary Algorithm Outline

1 $t \leftarrow 0$;
2 Initialize population $P(t = 0)$;
3 **while** *not termination-criterion* **do**
4 Parent-selection: $P'(t) \leftarrow select(P(t))$;
5 Crossover: $O(t) \leftarrow crossover(P'(t))$;
6 Mutation: $O'(t) \leftarrow mutate(O(t))$;
7 Selection: $P(t+1) \leftarrow select(O'(t) \cup P(t))$;
8 $t \leftarrow t+1$;

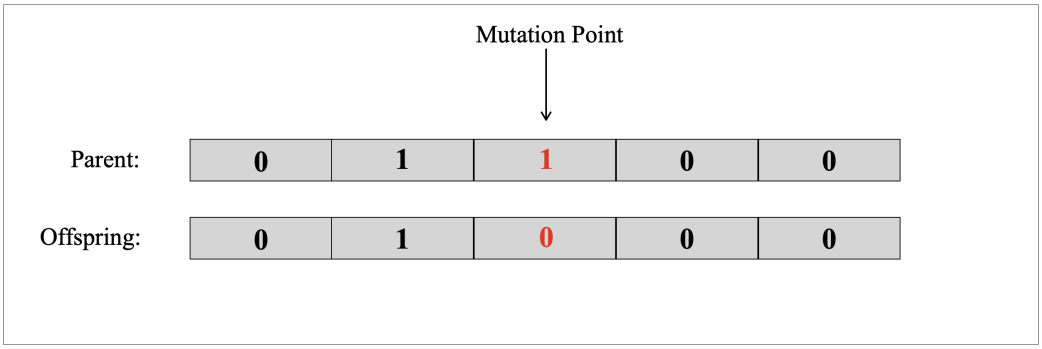


Figure 1.1. Example of a bitwise mutation

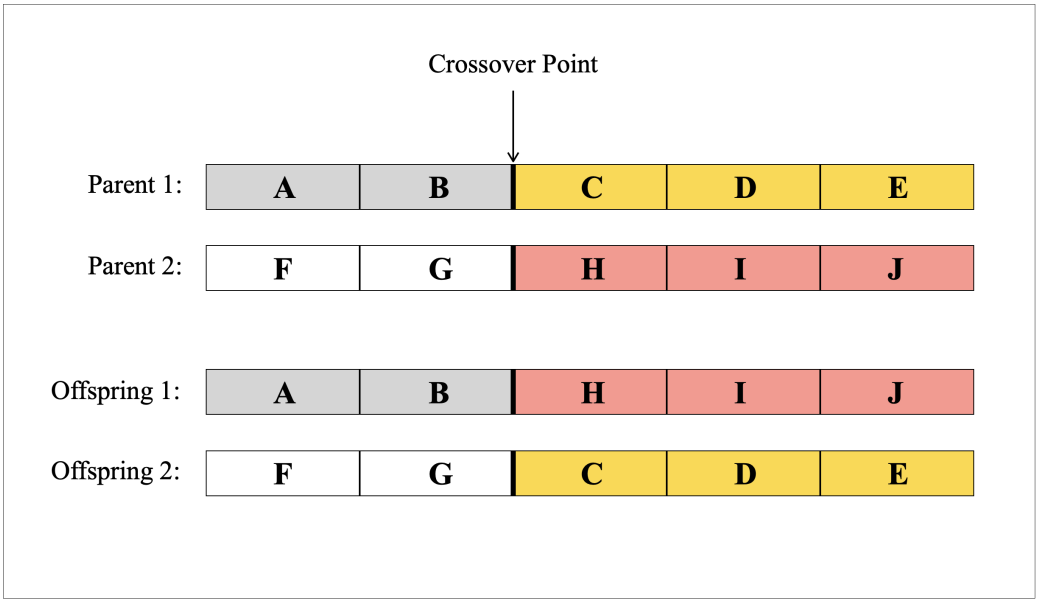


Figure 1.2. Example of a single-point crossover

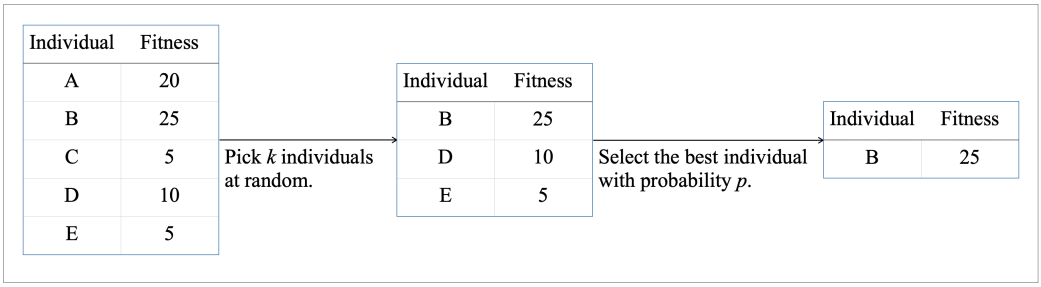


Figure 1.3. Example of the tournament selection procedure

1.2 Conclusion

We started this chapter by providing the motivation of this Thesis, looking at different types of present statistics for the topic of recycling. We then gave some background knowledge and definitions of linear and mixed-integer linear programming. Moreover, we introduced single and multi-objective problems. We also introduced the Pareto terminology needed to understand the subsequent sections, looking into Pareto dominance, optimal set, and front. Finally, to understand the following chapters, we illustrated the basic procedure of an evolutionary algorithm and gave fundamental mutation and crossover operators.

Chapter 2

Literature Review

The following sections aim to inspect the literature and research in the sphere of waste management optimisation, exact solution algorithms, and evolutionary algorithms.

2.1 Waste Management Optimisation

Researchers have sparingly investigated sustainable waste management optimization. Brandenburg et al. [2014] give insight into recent supply chain management (SCM) literature. They write that most papers often choose model types and techniques as multi-criteria decision making (MCDM) and mathematical programming, with very few papers focusing on genetic algorithms, dynamic programming, and neural networks as solution techniques.

Furthermore, researchers are investigating the sphere of vehicle routing. A 2019 paper proposes a novel way to find the location of waste sorting centres. It finds the optimal routes using a vehicle routing problem model and tries to find the waste sorting locations while investigating the economic, environmental and social aspects. Farahbakhsh and Forghani [2019]

Olapiriyakul et al. [2019] advance the field of sustainable waste management by proposing a multi-objective model to create a cost-effective waste management supply chain to help waste disposal for rapidly urbanising cities and developing countries. It focuses on three primary sustainability issues: environmental, social, and economic. In the context of the environmental issue, the authors specifically consider the land-use stress caused by municipal solid waste management (MSWM) by including a land-use equality objective to obtain a fairly distributed network design across the area. For the economic issue, they consider facility and transportation costs. Furthermore, they introduce a measure for the impact of the MSWM on human health based on the disability-

adjusted life years (DALYs) metric, commonly used by the World Health Organization (WHO) as a measure of the global burden of disease.
The optimization model considers a 3-echelon supply chain where solid wastes are in collection centres, then moved by a small truck to the sorting facilities, and finally sent by a large truck to either an incinerator or landfill facility. The model can make decisions on the locations and sizes of the facilities in tier 2 and tier 3 of the supply chain (i.e. sorting and landfill/incinerator facilities). The main objective is to determine the locations, sizes, and routes to minimize cost, land use, and the impact on public health. Olapiriyakul et al. [2019] describe the single-objective and multi-objective problems as Mixed Integer Linear Programming (MILP) minimisation problems.

Sets and Indices

a) I is the set of collection centers, indexed by i;

b) J is the set of sorting facility locations, indexed by j;

c) K is the set of incinerator facility locations, indexed by k;

d) K' is the set of landfill facility locations, indexed by k';

e) $L(j)$, $L(k)$, and $L(k')$ are the set of available sizes at locations j, k, and k', indexed by l.

Parameters

a) t_{ij}, t_{jk}, and $t_{jk'}$ are the transportation costs on links (i, j), (j, k), and (j, k'), respectively;

b) c_{jl}, c_{kl}, and $c_{k'l}$ are the fixed costs to open facilities of size l;

c) C_{jl}, C_{kl}, and $C_{k'l}$ are the storage capacities of facilities of size l;

d) $\overline{C}_{ij}$, $\overline{C}_{jk}$, and $\overline{C}_{jk'}$ define the maximum amount of waste a single trip can carry over links (i,j), (j,k), and (j,k'), respectively;

e) o_j, o_k, and $o_{k'}$ are unit operation costs to manage the flow of solid waste, for each facility;

f) s_{jl}, s_{kl}, $s_{k'l}$ are land-use stress ratios for facilities of size l;

g) D_i is the amount of solid waste available at collection center i;

h) p_{ij}, p_{jk}, and $p_{jk'}$ are the number of people living nearby links (i, j), (j, k), and (j, k'), respectively;

i) p_{jl}, p_{kl}, and $p_{k'l}$ are the number of people living near a facility of size l;

j) d_{ij}, d_{jk}, and $d_{jk'}$ are the DALYs per person due to transportation activities on links (i, j), (j, k), and (j, k'), respectively;

k) d_{jl}, d_{kl}, and $d_{k'l}$ are the DALYs per person due to size l facility operations.

Decision Variables

a) y_{jl}, y_{kl}, and $y_{k'l}$ are binary facility location variables equal to 1 when sorting, incinerator, and landfill facilities of size l are open at their respective locations, j, k, and k';

b) x_{ij}, x_{jk}, and $x_{jk'}$ are the number of vehicle trips on links (i, j), (j, k), and (j, k'), respectively;

c) f_{ij}, f_{jk}, and $f_{jk'}$ indicates the amount of solid waste transported on links (i, j), (j, k), and (j, k'), respectively.

Objective Functions

The MILP model comprises three objective functions: the cost objective function F_c, the land-usage objective function F_u, and the health-impact objective function F_h.

$$\begin{aligned} F_c = &\sum_{j\in J}\sum_{l\in L(j)} c_{jl}y_{jl} + \sum_{k\in K}\sum_{l\in L(k)} c_{kl}y_{kl} + \sum_{k'\in K'}\sum_{l\in L(k')} c_{k'l}y_{k'l} \\ &+ \sum_{i\in I}\sum_{j\in J}(t_{ij}+o_j)x_{ij} + \sum_{j\in J}\sum_{k\in K}(t_{jk}+o_k)x_{jk} \\ &+ \sum_{j\in J}\sum_{k'\in K'}(t_{jk'}+o_{k'})x_{jk'} \end{aligned} \tag{2.1}$$

The cost objective, F_c, comprises fixed costs to open sorting, incinerator, and landfill facilities and the operational cost of transporting and managing solid waste flow throughout the network.

$$F_u = \sum_{j\in J}\sum_{l\in L(j)} s_{jl}y_{jl} + \sum_{k\in K}\sum_{l\in L(k)} s_{kl}y_{kl} + \sum_{k'\in K'}\sum_{l\in L(k')} s_{k'l}y_{k'l} \tag{2.2}$$

The land-usage objective, F_u, is the sum of all land-use ratios throughout the network. The parameters s_{jl}, s_{kl}, and $s_{k'l}$ denote the total land used and available land ratios. [Olapiriyakul et al., 2019, p. 7] use the following ratio to compute the parameters s_{jl}, s_{kl}, and $s_{k'l}$.

$$\frac{\text{Direct + Indirect land use, with a facility of size } l}{\text{Total land available}} \tag{2.3}$$

The third objective function is the health-impact objective, F_h, which evaluates the impact of transportation and facilities on the population.

$$\begin{aligned} F_h = &\sum_{j\in J}\sum_{l\in L(j)} p_{jl}d_{jl}y_{jl} + \sum_{k\in K}\sum_{l\in L(k)} p_{kl}d_{kl}y_{kl} + \sum_{k'\in K'}\sum_{l\in L(k')} p_{k'l}d_{k'l}y_{k'l} \\ &+ \sum_{i\in I}\sum_{j\in J} p_{ij}d_{ij}x_{ij} + \sum_{j\in J}\sum_{k\in K} p_{jk}d_{jk}x_{jk} + \sum_{j\in J}\sum_{k'\in K'} p_{jk'}d_{jk'}x_{jk'} \end{aligned} \tag{2.4}$$

Constraints

$$\sum_{j \in J} f_{ij} = D_i, \qquad \forall i \in I \tag{2.5}$$

$$\sum_{i \in I} f_{ij} = \sum_{k \in K \cup K'} f_{jk}, \qquad \forall j \in J \tag{2.6}$$

$$\sum_{i \in I} f_{ij} \leq \sum_{l \in L(j)} C_{jl} y_{jl}, \qquad \forall j \in J \tag{2.7}$$

$$\sum_{j \in J} f_{jk} \leq \sum_{l \in L(k)} C_{kl} y_{kl}, \qquad \forall k \in K \tag{2.8}$$

$$\sum_{j \in J} f_{jk'} \leq \sum_{l \in L(k')} C_{k'l} y_{k'l}, \qquad \forall k' \in K' \tag{2.9}$$

$$f_{ij} \leq \overline{C}_{ij} x_{ij}, \qquad \forall i \in I, j \in J \tag{2.10}$$

$$f_{jk} \leq \overline{C}_{jk} x_{jk}, \qquad \forall j \in J, k \in K \tag{2.11}$$

$$f_{jk'} \leq \overline{C}_{jk'} x_{jk'}, \qquad \forall j \in J, k' \in K' \tag{2.12}$$

$$\sum_{l \in L(j)} y_{jl} \leq 1, \qquad \forall j \in J \tag{2.13}$$

$$\sum_{l \in L(k)} y_{kl} \leq 1, \qquad \forall k \in K \tag{2.14}$$

$$\sum_{l \in L(k')} y_{k'l} \leq 1, \qquad \forall k' \in K' \tag{2.15}$$

$$y_{jl}, y_{kl}, y_{k'l} \in \{0, 1\}, \qquad \forall j, k, k', l \tag{2.16}$$

$$x_{ij}, x_{jk}, x_{jk'} \in \mathbb{Z}^+, \qquad \forall i, j, k, k' \tag{2.17}$$

$$f_{ij}, f_{jk}, f_{jk'} \geq 0, \qquad \forall i, j, k, k' \tag{2.18}$$

Equation (2.5) declares that the outflow of waste from any collection centre i must be equal to the demand D_i. Constraint (2.6) is a flow balance restriction, which enforces that the inflow at any sorting facility j is forwarded to the incinerators k and landfills k'. Constraints (2.7) - (2.9) are capacity constraints for all network facilities. These enforce that the flow into a facility is at most equal to the facility's capacity. The inequalities (2.10) - (2.12) constrain the vehicle transportation capacity for every link (i, j), (j, k), and (j, k'), respectively. Constraints (2.13) - (2.15) enforce that each open facility can have at most one facility size selected. Finally, equations (2.16) - (2.18) define the decision variables' domains, where (2.16) are binary values, (2.17) are positive integer values, and (2.18) are real numbers that are at least 0.

To allow for a multi-objective MILP optimisation, [Olapiriyakul et al., 2019, pp. 5 — 6] use a min-max approach which minimises the deviations from the ideal results. The deviations use the optimal values F_c^*, F_u^*, and F_h^*, obtained by solving their respec-

tive single-objective problem. Moreover, F_c^{max}, F_u^{max}, and F_h^{max} are the upper bounds obtained by the worst values for each function across the single-objective problems. Normalisation according to the ideal target occurs for each function to compute the deviation values.

$$\sigma_c = \frac{F_c - F_c^*}{F_c^{max} - F_c^*} \tag{2.19}$$

$$\sigma_u = \frac{F_u - F_u^*}{F_u^{max} - F_u^*} \tag{2.20}$$

$$\sigma_h = \frac{F_h - F_h^*}{F_h^{max} - F_h^*} \tag{2.21}$$

Olapiriyakul et al. [2019] add a new continuous variable, z, which minimises the deviation levels of the objective functions. The multi-objective formulation of the problem is thus a min-max on the deviations of the objective functions as follows.

$$\min \; z \tag{2.22}$$

$$\text{s.t.} \; \sigma_c \leq z \tag{2.23}$$

$$\sigma_u \leq z \tag{2.24}$$

$$\sigma_h \leq z \tag{2.25}$$

$$(2.5) - (2.18) \tag{2.26}$$

2.2 Exact Solution Algorithms

This section overviews the exact algorithmic approaches in the literature to solving MILP and combinatorial optimisation problems.

2.2.1 Branch and Bound

Land and Doig [1960] pioneered the famous Branch and Bound algorithm, a search strategy based on a binary enumeration tree. The algorithm consists of two primary operations: *branching* that recursively splits the search space S into smaller spaces, then minimising the objective function $f(x)$, and *bounding*, which attempts to prune the search space, thus eliminating candidate solutions that do not contain an optimal solution.

Definition 2.2.1 (Branch and Bound Operations): *Let I be an instance of a problem, and the set of candidate solutions of an instance I be S_I. Then the two operations to the branch and bound algorithm are:*

a) $branch(I)$*, which produces two or more disjoint subsets of S_I;*

b) $bound(I)$*, which computes the lower bound on the value of candidate solutions in I.* $bound(I) \leq f(x), \ \forall x \in S_I$.

□

Algorithm 2: Generic Branch and Bound

Input : A problem **p**, a objective function **f**, and a lower bounding function **LB**.
Output: The optimal solution $\mathbf{x}^*$.

```
1  UB ← ∞;
2  x* ← ∅;
3  Initialise a candidate queue based on the problem Q ← Q(p);
4  while Q ≠ ∅ do
5      Node ← Q.pop();
6      if Node N represents a single candidate solution x and f(x) < B then
7          x* ← x;
8          UB ← f(x*);
9      else
10         foreach child N_i ∈ Node N do
11             if LB(N_i) ≤ UB then
12                 Q ← Q.enqueue(N_i);
```

2.2.2 Branch and Cut

To extend the branch and bound algorithm to integer linear problems (ILPs) or MILPs, a method known as branch and cut combines the branch and bound algorithm with cutting planes. Gomory [1960] proposed the *cutting plane method*, which includes a *cut* as an inequality constraint that tightens a relaxed problem's solution space without cutting out any candidate solution in S. Cutting planes can find an optimal solution to ILPs in a finite number of iterations. However, it is ineffective as many rounds of cuts are needed to progress towards a solution. For this reason, the generation of cuts is combined with branch and bound, resulting in the *branch and cut* approach.
We can relax an integer linear program by removing the integrality constraint of each variable. For example, if we take a binary integer program, then all the constraints are of the form:

$$x_i \in \{0, 1\}. \tag{2.27}$$

The relaxed form of the problem converts these constraints into linear constraints:

$$0 \leq x_i \leq 1. \tag{2.28}$$

This relaxation helps to find a bound on an integer program's solution. Coupled with branch and bound, this provides an exact solution to complex optimisation problems. Algorithm 3 shows the pseudocode for the branch and cut algorithm for a minimisation problem.

Algorithm 3: Generic Branch and Cut

Input : An integer linear programming problem **p**, an objective function **f**.
Output: The optimal solution $\mathbf{x}^*$.

```
1  Add the initial problem to a queue Q:
2  Q ← Q.enqueue(p);
3  v* ← ∞;
4  x* ← ∅;
5  while Q ≠ ∅ do
6      Obtain a problem from the queue Q:
7      P ← Q.dequeue(Q);
8      Solve the linear programming relaxation of P:
9      R ← relax(P);
10     x ← solve_L P(R);
11     if ¬x.isfeasible() or v* ≥ f(x) then
12         Go back to step 5.
13     if x.isinteger() then
14         x* ← x;
15         v* ← f(x);
16         Go back to step 5.
17     Search for violated cutting planes:
18     VCP ← ViolatedCuttingPlanes(x)
19     if VCP ≠ ∅ then
20         foreach Cutting plane CP ∈ VCP do
21             Q ← Q.enqueue(relax(P), CP);
22         Go back to step 8.
23     Branch to partition the problem into new problems:
24     N ← branch(P);
25     foreach child N_i ∈ Branch N do
26         Q ← Q.enqueue(N_i);
27 return x*
```

2.3 Heuristic Solution Algorithms

This section overviews the heuristic algorithmic approaches in the literature to solving MILPs and multi-objective combinatorial optimisation problems.

2.3.1 Multi-objective optimisation with NSGA

NSGA (Non-dominated Sorting Genetic Algorithm) is one example of a famous multi-objective evolutionary algorithm. It uses dominance ranking to move the population to the Pareto front in a multi-objective problem. To determine the dominance rank of an individual, we count how many individuals are dominating this individual. The result of a dominance ranking procedure, as shown in fig. 2.1, is a partially ordered list used for sorting the points before utilising the selection operator. [Coello Coello et al., 2007, pp. 79–81]

In addition, it also uses a technique named fitness sharing (or niching), seen in fig. 2.2, where one counts how many solutions are in a neighbourhood of solutions and proportionally decreases the fitness of the solutions in the same neighbourhood. It introduces a neighbourhood radius, σ_{share}, to control the size of a neighbourhood. Fitness sharing promotes the creation of solutions in the least populated regions of the search space. [Coello Coello et al., 2007, p. 81]

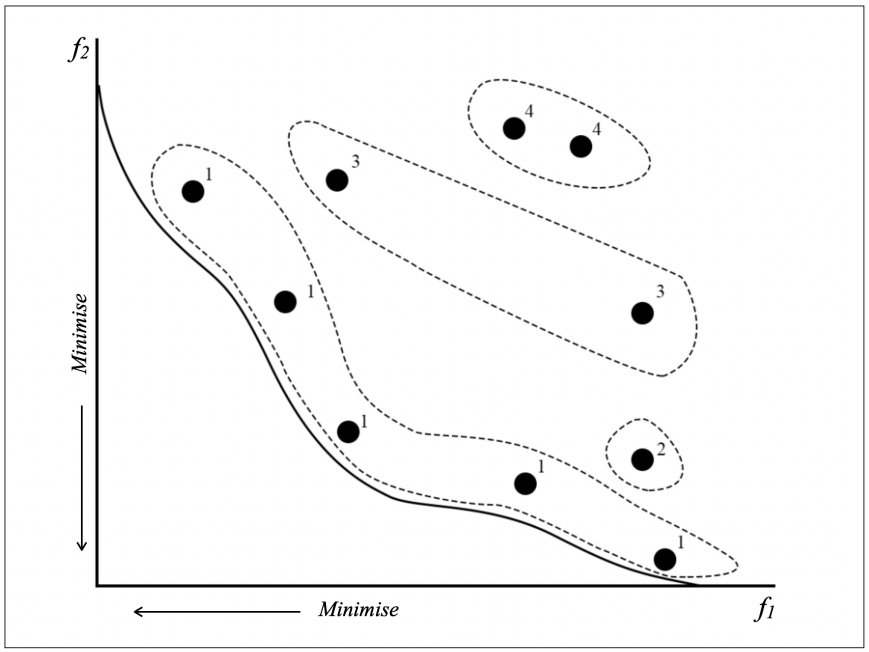


Figure 2.1. Dominance ranking example with the grouping of equal ranks for sorting. (Adapted from [Coello Coello et al., 2007, p. 80])

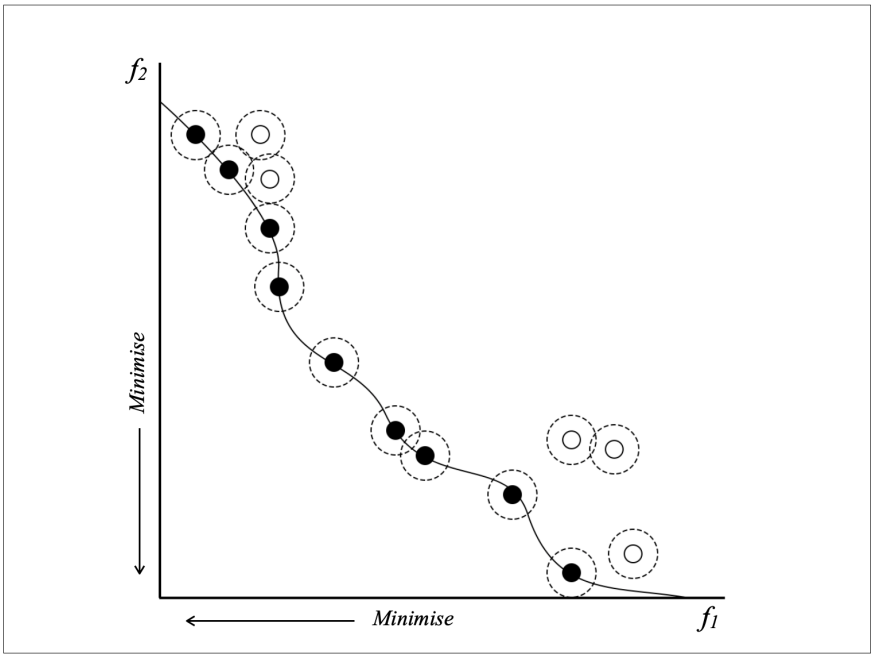


Figure 2.2. An illustration of the fitness sharing scheme. (Adapted from [Coello Coello et al., 2007, p. 82])

NSGA uses these techniques in conjunction with the population being ranked based on dominance and classified into a hierarchy of subpopulations based on the order of domination.

NSGA then evaluates the similarity between the members of these subgroups using the fitness sharing approach to promote population diversity. An advantage of this approach is that the individuals in the first front obtain more copies than the rest of the population due to their higher fitness values, which allows for a convergence of the population towards the known Pareto front regions. [Coello Coello et al., 2007, pp. 91–93]

Nonetheless, a disadvantage of NSGA is that it uses the fitness sharing mechanism. Deb et al. [2002] write that there are two issues with the sharing approach.

1. "The performance of the sharing function method in maintaining a spread of solutions depends largely on the chosen σ_{share} value." [Deb et al., 2002, p. 184]

2. "Since each solution must be compared with all other solutions in the population, the overall complexity of the sharing function approach is $\mathcal{O}(N^2)$." [Deb et al., 2002, p. 185]

NSGA-II

NSGA-II addresses this bottleneck by using a crowding sort technique using the crowding distance. The crowding distance measures the density of solutions surrounding a particular solution in the population. In fig. 2.3, we can see an illustration of the crowding distance. To calculate the crowding distance of a particular solution, i, the following equation is applied, where f_k^{max} and f_k^{min} are the maximum and minimum values of the kth objective function.

$$D_i = \sum_{k \in K} \frac{f_k(i+1) - f_k(i-1)}{f_k^{max} - f_k^{min}} \tag{2.29}$$

NSGA-II uses crowding distance sorting, where sorting occurs on the population in ascending order of magnitude before calculating the crowding distance. According to [Deb et al., 2002, p. 185], this approach's algorithmic complexity is $\mathcal{O}(MN \log N)$, with M independent sortings of at most N solutions.

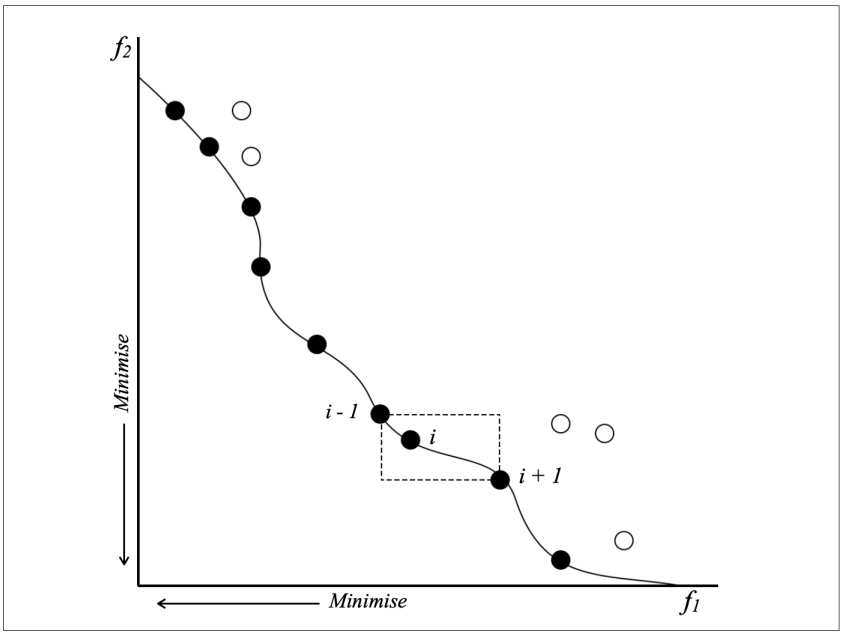


Figure 2.3. Illustration of the crowding distance. (Adapted from [Coello Coello et al., 2007, p. 83])

In fig. 2.4, NSGA-II builds a population of individuals and ranks and sorts each individual according to the non-domination level. It then uses the evolutionary operators to create new offspring. Afterwards, it applies fitness sharing by adding a crowding distance to every individual of the population, which is more efficient than computing the fitness sharing count.

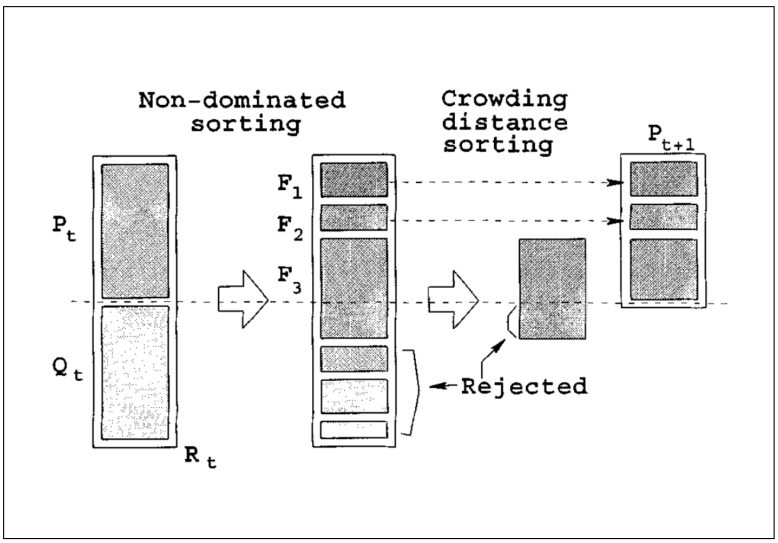


Figure 2.4. The procedure of NSGA-II. [Deb et al., 2002, p. 186]

Crowding distance sorting decides which individuals get added to the new population to keep a diverse front of solutions in the population and avoid local minimums (or maximums). The individuals with higher crowding distances are selected first and filled until it reaches the desired number of parents. The individuals who did not make it into the parent population get rejected. The algorithmic complexity of NSGA-II, considering the non-dominated sorting, the crowding-distance assignment, and the crowding-distance sorting, is $\mathcal{O}(MN^2)$ [Deb et al., 2002, p. 186]. Algorithm 4 illustrates the NSGA-II algorithm from an adaptation of the pseudocode from [Coello Coello et al., 2007, p. 93] and [Deb et al., 2002, p. 186].

Algorithm 4: NSGA-II Algorithm

Input: Population size N, Total generations g.

Initialise population P with N number of random individuals;
for $t = 1$ *to* g **do**
    Combine parent and offspring population:
        $R_t \leftarrow P_t \cup Q_t$;
    $F \leftarrow$ Apply the non-dominated sorting procedure on R_t
    $P_{t+1} \leftarrow \emptyset$;
    $j \leftarrow 1$;
    while $|P_{t+1}| + |F_j| \leq N$ **do**
        Calculate the crowding distance of F_j
        Include the jth non-dominated front into the parent population P_{t+1}:
            $P_{t+1} \leftarrow P_{t+1} \cup F_j$;
        $j \leftarrow j + 1$;
    Sort F_j in descending order
    Add the first $(N - |P_{t+1}|)$ elements of F_j to P_{t+1}:
        $P_{t+1} \leftarrow P_{t+1} \cup F_j[1 : (N - |P_{t+1}|)]$;
    Create next generation Q_{t+1}:
        Binary tournament selection
        Crossover and Mutation

2.3.2 Generating well-spaced reference points on a unit simplex

Most evolutionary many-objective optimisation (EMaO) algorithms start with a predefined set of reference points on a unit simplex. These points operate as guides to search for a single Pareto optimal solution across each reference direction. The approach by Das and Dennis [1998] is the most common method of creating a set of uniformly spaced reference points on a normalised hyperplane. This approach starts by initialising a set of points on a K-dimensional unit simplex to generate reference points subject to,

$$\sum_{i=1}^{K} z_i = 1, \quad \text{for } \mathbf{z} \in [0,1]^K \tag{2.30}$$

Subsequently, the construction of reference directions uses a vector which originates from the origin and connects to all initialised reference points. The output of this procedure is a K-dimensional unit simplex to all objective axes with an intercept of one on each axis. If an EMaO algorithm works well enough to find a single Pareto-optimal solution for one reference direction, then a well-spaced set of Pareto-optimal solutions is anticipated due to the unit simplex.

The parameter ρ, which indicates the number of gaps between two successive points along an objective axis, controls the number of points on the unit simplex. The total number of points, H, on a unit simplex are:

$$H = C_{\rho}^{K+\rho-1} \tag{2.31}$$

Eq. 2.31 shows a binomial coefficient for the number of points, represented in factorial notation as,

$$H = \frac{(K+\rho-1)!}{\rho!\,(K-1)!} \tag{2.32}$$

A problem arises with this approach when there are a preferred number of points. This problem occurs as there is a drastic increase in the number of points when increasing p by one. Another problem with the Das and Dennis [1998] approach occurs as K increases, the number of total points on the unit simplex increases severely. Moreover, this requires a vast population for the EMaO algorithms as the population size is usually the same as the number of points.

Blank et al. [2021] develop an approach to generating well-spaced points on a unit simplex that mends the abovementioned problems. They introduce a novel method that uses a generalisation of potential energy called *Riesz s-Energy*.

Definition 2.3.1 (Riesz s-Energy Hardin and Saff [2005]): *Let* $\mathbf{x} = \{x_1, ..., x_N\}$ *denote a set of N distinct points in the d-dimensional unit sphere* $S^d \in \mathbb{R}^{d+1}$ *or a rectifiable* d'*-dimensional manifold embedded in* $\mathbb{R}^{d'}$*, where* $s > 0$ *is a fixed parameter, then the*

definition of the Riesz s-Energy is

$$E_s(\mathbf{x}) = \sum_{i=1}^{N} \sum_{\substack{j=1 \\ i \neq j}}^{N} \frac{1}{\|x_i - x_j\|^s} \tag{2.33}$$

□

[Blank et al., 2021, p. 53] set $s = K^2$ motivated by trial-and-error studies. The goal of the energy method is to find the $\mathbf{x}$-matrix of size $\mathbb{R}^{N \times K}$ that minimises the total energy E_s subject to $\sum_{k=1}^{K} x_k = 1$. Blank et al. [2021] use the Adam gradient optimiser to minimise this problem. As E_s has a considerable magnitude, the authors take the logarithm of E_s and then compute the partial derivative of $F_E = \log E_s$ w.r.t x_k as follows.

$$\frac{\partial F_E}{\partial x_k^{(i)}} = -\frac{s}{E_s} \left[\sum_{\substack{j=1 \\ i \neq j}}^{N} \frac{x_k^{(i)} - x_k^{(j)}}{\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^{s+2}} \right] \tag{2.34}$$

Finally, the authors project the gradients onto the unit simplex to ensure that all points stay on the unit simplex. Figure 2.5 displays the distribution of 91 circle and 92 cross points on the unit simplex. The 92 points are not possible with the method from Das and Dennis [1998]. Interestingly, the lower right corner points get readjusted to accommodate the extra 92nd point. This method effectively finds a distribution of points that are still very well-spaced even though, to our knowledge, no well-spaced distribution of 92 points exists.

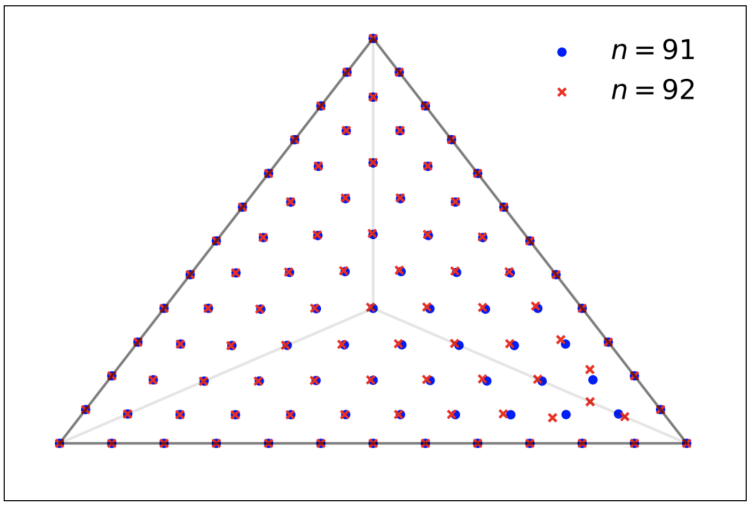


Figure 2.5. Points obtained by Riesz's s-energy method with 91 circle points and 92 cross points. (Taken from [Blank et al., 2021, p. 57]

2.3.3 Many-objective optimisation using NSGA-III

NSGA-III

NSGA-II works best on problems requiring two to three objectives and struggles in handling many objectives. NSGA-III is an extension of NSGA-II that can handle three or more objectives. It achieves this by using a predefined set of reference points, ensuring diversity in the solutions obtained. Usually, the reference points are predefined in a structured manner using the approaches found in section 2.3.2. NSGA-III uses the same non-dominated sorting approach found in NSGA and NSGA-II. As outright computing the crowding distance takes too much computational time on problems with many objective functions, the reference points help to find good representative, non-dominated solutions.
First, NSGA-III creates a new population, S, that fills up the underrepresented reference points using the non-dominated sorted solutions. If the last accepted non-dominated solution, F_l, is only partially added to S, NSGA-III selects only the solutions that maximise the diversity of the solution's front. It does this by using an alternative to the crowding operator shown in NSGA-II. The outline of the approach from [Deb and Jain, 2014, pp. 581–583] is shown in procedure 5. In addition, since NSGA-III assigns only one population member to each reference direction, there is no need for selection, as it would allow for competition between different reference directions. According to [Deb and Jain, 2014, p. 583] the algorithmic complexity of NSGA-III through this procedure is still $\mathcal{O}(MN^2)$.

Definition 2.3.2 (Achievement scalarising function): *Let* $\mathbf{w} = \{w_1, w_2, ..., w_K\}$*, be a non-negative weight vector with* $w_i \in (0, 1]$*, for* $i = \{1, ..., K\}$*,* $f(x) = \{f_1(x), f_2(x), ..., f_K(x)\}$ *a minimisation (or maximisation) multi-objective optimisation problem, and* $\mathbf{z}^* = \{z_1^*, z_2^*, ..., z_K^*\}$ *the ideal points for each objective function. Then the definition of an achievement scalarising function is,*

$$ASF(f(x) \mid \mathbf{w}, \mathbf{z}^*) = \max_{1 \leq i \leq K} \left\{ \frac{f_i(x) - z_i^*}{w_i} \right\} \tag{2.35}$$

□

Algorithm 5: Alternative crowding distance operator for NSGA-III

Input : The output of the non-dominated sorting **F**, the filled non-dominated sorting population **S**, and the structured reference points $\mathbf{Z^s}$

Output: The new population $\mathbf{P_{t+1}}$.

1. *Adaptive Normalisation of Population members*

   (a) Compute the ideal point by taking the minimum value for each objective function.

   $$z_k^{ideal} = \min_{\mathbf{s}\in S} f_k(\mathbf{s}), \quad k = \{1, ..., K\} \tag{2.36}$$

   (b) Translate all objective values in the population S by subtracting the ideal point from them.

   $$f_k'(\mathbf{s}) = f_k(\mathbf{s}) - z_k^{ideal}, \quad \forall \mathbf{s} \in S \tag{2.37}$$

   (c) Compute the extreme points $z^{k,extreme}$ in each objective axis $k = \{1, ..., K\}$ by finding the solution, $s \in S$, which minimises the corresponding *achievement scalarising function* defined in definition 2.3.2.

   $$z^{k,extreme} = f_k(\mathbf{s}) : \operatorname*{argmin}_{\mathbf{s}\in S} \left\{ ASF(f'(\mathbf{s}) \mid \mathbf{w}, \mathbf{z^{ideal}}) \right\} \tag{2.38}$$

   (d) Computation of the intercept, a_k, of the kth objective, uses the approach shown in fig. 2.6. In practice, the computation of the intercept, a_k, uses Gaussian Elimination. The normalisation on the objective function occurs using the following equation.

   $$f_k^n(\mathbf{s}) = \frac{f_k'(\mathbf{s})}{a_k}, \quad \text{for } k = \{1, ..., K\} \tag{2.39}$$

2. *Association Operation* to associate each population member with a reference point.

   (a) By joining reference points with the origin, create a reference line, **u**, for each reference point, $z \in Z^s$.

   (b) Calculate the perpendicular distance of each population member, $\mathbf{s} \in S$, to each reference line, $\mathbf{u} \in Z^s$.

   $$d^{\perp}(\mathbf{s}, \mathbf{u}) = \left\| \mathbf{s} - \frac{\mathbf{u}^T \mathbf{s}\mathbf{u}}{\|\mathbf{u}\|_2} \right\|_2 \tag{2.40}$$

   (c) Associate the population member in S to the reference point with the closest reference line.

   $$\pi(\mathbf{s}) = \mathbf{u} : \operatorname*{argmin}_{\mathbf{u}\in Z^s} d^{\perp}(\mathbf{s}, \mathbf{u}) \tag{2.41}$$

3. *Niche-Preservation Operation* ensures every reference point has at most one population member.

   (a) Count the number of population members, $P_{t+1} = S/F_l$, associated with each reference point. Let ρ_z denote the count at reference point $z \in Z^s$.

   (b) Find the reference point set with the minimum number of reference point associations, i.e. $Z^s_{\min} = \{z : \arg\min_{z\in Z^s} \rho_z\}$. In the case of multiple such reference points, choose a random $\bar{z} \in Z^s_{\min}$.

   (c) If there is no associated P_{t+1} member to the chosen reference point, $\bar{z}$, choose the member from the front F_l with the closest perpendicular distance to the reference line.

   (d) If one or more P_{t+1} members are associated with the chosen reference point, choose a random associated member from the front F_l.

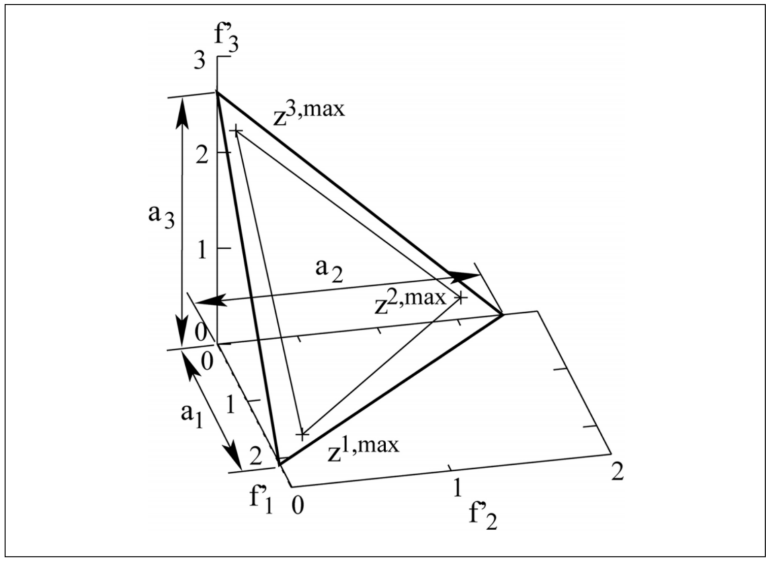


Figure 2.6. Procedure for computing intercepts and then forming the hyper-plane from extreme points. Shown for a three objective problem. [Deb and Jain, 2014, p. 582]

U-NSGA-III

[Seada and Deb, 2016, p. 361] extend the original NSGA-III algorithm to unify mono-objective, bi-objective, and many-objective problems. They do this by including a tournament selection procedure to select the parents P_t. For multi-objective problems (two or three objectives), if the population size N is greater than the number of reference directions H, the reference directions have multiple population members for each reference direction. The non-dominated sorting divides the population into multiple non-dominated fronts. The selection procedure introduced by U-NSGA-III will emphasise:

1. "Non-dominated solutions over dominated solutions" [Seada and Deb, 2016, p. 362]
2. "Solutions closer to reference directions over other non-dominated but distant solutions from the reference directions" [Seada and Deb, 2016, p. 363]

Compared with NSGA-II, in U-NSGA-III, the distribution of $(N-H)$ population members do not need a good diversity among them. Only H population members closest to H reference directions will have a well-spaced diversity. Although, since the user stated to find H solutions, the additional $(N-H)$ population members make the algorithm more efficient by finding exactly H Pareto-optimal points.
Nevertheless, when the chosen N is very close or equal to H, the algorithm may not obtain a solution for every reference direction in the early generations. However, since the

tournament selection introduces selection pressure, the algorithm emphasises finding a single population member for each reference direction.

Algorithm 6 shows the tournament selection procedure added to U-NSGA-III, based on [Seada and Deb, 2016, p. 362]. Furthermore, Algorithm 7 shows the full NSGA-III algorithm with optional unification, based on Deb and Jain [2014] and Seada and Deb [2016].

Algorithm 6: U-NSGA-III tournament selection procedure

Input : Two parents: p_1 and p_2.
Output: The selected individual p_s.

```
if p_1 and p_2 are feasible then
    if Rank of p_1 < Rank of p_2 then
        p_s ← p_1;
    else if Rank of p_2 < Rank of p_1 then
        p_s ← p_2;
    else if Perpendicular distance d^⊥(p_1) < Perpendicular distance d^⊥(p_2) then
        p_s ← p_1;
    else
        p_s ← p_2;
else
    p_s ← RandomSelection(p_1, p_2);
```

Algorithm 7: The combination of the NSGA-III and U-NSGA-III algorithms for a single generation t

Input : A set of reference points Z^s, the parent population P_t, a boolean to use the U-NSGA-III selection procedure U.
Output: The new population P_{t+1}.

$S_t \leftarrow \emptyset$;
$i \leftarrow 1$;
if $U = True$ **then**
    Apply the tournament selection described in Algorithm 6:
        $P'_t \leftarrow$ TournamentSelection(P_t);
    $Q_t \leftarrow$ Crossover+Mutation(P'_t);
else
    $Q_t \leftarrow$ Crossover+Mutation(P_t);
$R_t \leftarrow P_t \cup Q_t$;
$\mathbf{F} \leftarrow$ non-dominated-sort(R_t);
while $|S_t| \leq N$ **do**
    $S_t \leftarrow S_t \cup F_i$;
    $i \leftarrow i + 1$;
Last front included in S_t: $F_l \leftarrow F_i$;
if $|S_t| = N$ **then**
    $P_{t+1} \leftarrow S_t$;
    return P_{t+1}
else
    $P_{t+1} \leftarrow$ Apply the procedure given in Algorithm 5.
return P_{t+1}

2.3.4 Constrained multi-objective optimisation using a two-archive evolutionary algorithm (C-TAEA)

Li et al. [2019] focus on the design of evolutionary algorithms for constrained multi-objective optimisation problems (CMOP). The authors propose a two-archive evolutionary algorithm, C-TAEA, which focuses on the *convergence*, *diversity*, and *feasibility* of the solutions. The convergence-oriented archive (CA) tries to maintain the convergence and feasibility of the evolution process. In addition, it also provides selection pressure towards the Pareto-optimal front. Similarly, the diversity-oriented archive (DA) maintains convergence and diversity. In particular, it explores the areas of the feasible region that the CA has not exploited, which improves the population's diversity and helps jump over the local optima. Furthermore, the authors add a restricted mating selection mechanism that "selects the appropriate mating parents from the CA and DA seperately according to their evolution status" [Li et al., 2019, p. 304].

Li et al. [2019] introduce the association operator, which, similar to NSGA-III, calculates the perpendicular distance of each population member $s \in S$ to each reference line $u \in Z^s$. It then associates every individual of S to the reference point (or subregion) with the closest reference line. The CA first pushes the population towards a feasible region and then tries to balance the convergence and diversity inside of the feasible region. Algorithm 9 shows the procedure outlined in [Li et al., 2019, pp. 306-307]. Note that Li et al. [2019] propose a process to improve the population diversity and to identify the worst solution $\mathbf{x}^w$ (line 16 – 19 in Algorithm 9). First, calculate the distance between each solution $\mathbf{x} \in \pi_i$ and its nearest neighbour,

$$dist(\mathbf{x}) = \min_{\substack{\mathbf{x}' \in \pi_i \\ \mathbf{x} \neq \mathbf{x}'}} \left\{ \|\mathbf{x} - \mathbf{x}'\|_2 \right\} \tag{2.42}$$

Subsequently, the temporary archive S_d stores the solutions having the smallest distance and the definition of $\mathbf{x}^w$ is then,

$$\mathbf{x}^w = \underset{\mathbf{x} \in S_d}{\operatorname{argmax}} \left\{ g^{tch}(\mathbf{x} | \mathbf{z}^{\mathbf{ideal}}, \mathbf{w}^{\mathbf{i}}) \right\} \tag{2.43}$$

where

$$g^{tch}(\mathbf{x} | \mathbf{z}^{\mathbf{ideal}}, \mathbf{w}^{\mathbf{i}}) = \max_{1 \leq j \leq K} \left\{ \frac{|f_j(\mathbf{x} - z_j^{ideal}|}{w_j^i} \right\}, \text{ where } w^i \in (0, 1] \tag{2.44}$$

The DA, on the other hand, aims to provide as many diversified solutions as possible. Its update mechanism has two main characteristics [Li et al., 2019, p. 307]:

1. It does not consider the constraint violation.
2. It takes the population outputted by the CA and complements the behaviour of the CA by exploring the under-exploited areas.

The algorithm iteratively investigates each subregion, π_i, and decides the survival of the solutions in H^d. It makes these decisions by reviewing how many solutions exist in P^c at π_i. Moreover, if $iter$ solutions exist in P^c at π_i, then no solution in H^d will be considered to survive at π_i for this iteration. Otherwise, it chooses the best non-dominated solutions of H^d at π_i to survive [Li et al., 2019, p. 307]. Algorithm 8 shows the full procedure of the DA taken from [Li et al., 2019, p. 308].

For the offspring generation, the mating selection leverages the principal information from both archives. It combines both archives into one archive, H_m, and then checks the proportion of non-dominated solutions for each archive in H_m. If there is a higher proportion for archive CA, it means that the convergence status of the CA is better than the DA, and the first parent gets selected using a tournament selection procedure from the CA archive based on feasibility. Otherwise, the first parent gets selected the same way but from the DA archive. [Li et al., 2019, p. 307-308]

For the second mating parent, whether chosen from the CA or DA archives depends on the proportion of non-dominated individuals in the CA archive. The more non-dominated solutions the CA archive has, the higher the probability of the second parent coming from the CA archive and the lower the probability of it coming from the DA archive. [Li et al., 2019, p. 308]

The tournament selection procedure of the C-TAEA is feasibility-driven. Principally, if the randomly selected candidates are feasible, they are chosen based on Pareto dominance. However, if only one of the candidates is feasible, the feasible one is chosen. Otherwise, a random selection occurs between both candidates. [Li et al., 2019, p. 308]

Algorithm 8: DA update procedure for a single generation t

Input : The convergence archive population P_t^c, the diversity archive population P_t^d, the offspring population Q_t, and a set of reference directions Z^s

Output: The updated diversity archive population P_{t+1}^d

$S \leftarrow \emptyset,\ iter \leftarrow 1$;
Create the hybrid population H_d
  $H^d \leftarrow P_t^d \cup Q_t$;
$\pi_d \leftarrow Associate(H^d, Z^s)$;
$\pi_c \leftarrow Associate(P_t^c, Z^s)$;
while $|S| < N$ **do**
  for $i \leftarrow 1$ **to** N **do**
    if $|\pi_c^i| < iter$ **then**
      for $j \leftarrow 1$ **to** $iter - |\pi_c^i|$ **do**
        if $\pi_d^i \neq \emptyset$ **then**
          $O^i \leftarrow$ non-dominated-sorting$(H^d_{\pi_d^i})$;
          $\mathbf{x}^{best} \leftarrow \text{argmin}_{\mathbf{x} \in O^i} \left\{ g^{tch}(\mathbf{x} \mid \mathbf{z}^*, \mathbf{Z_c^s}) \right\}$;
          $\pi_d^i \leftarrow \pi_d^i \backslash x^{best}$;
          $S \leftarrow S \cup x^{best}$;
        else
          break
  $iter \leftarrow iter + 1$;
$P_{t+1}^d \leftarrow S$;
return P_{t+1}^d

Algorithm 9: CA update procedure for a single generation t

Input : The convergence archive population P_t^c, the offspring population Q_t, and a set of reference directions Z^s

Output: The updated convergence archive population P_{t+1}^c

$S \leftarrow \emptyset,\ S_c \leftarrow \emptyset, i \leftarrow 1$;
Create the hybrid population H_c
  $H_c \leftarrow P_t^c \cup Q_t$;
Add the feasible solutions in H_c to a temporary archive S_c
if $|S_c| = N$ **then**
  $P_{t+1}^c \leftarrow S_c$;
else if $|S_c| > N$ **then**
  $\mathbf{F} \leftarrow$ non-dominated-sort(S_c);
  while $|S| < N$ **do**
    $S \leftarrow S \cup F_i$;
    $i \leftarrow i + 1$;
  if $|S| > N$ **then**
    $\pi = Association(S,\ Z^s)$ as in Algorithm 5;
    while $|S| > N$ **do**
      Find the most crowded subregion (or reference point) π_i
      foreach $\mathbf{x} \in \pi_i$ **do**
        $\text{dist}(\mathbf{x}) \leftarrow \min_{\substack{\mathbf{x}' \in \pi_i \\ \mathbf{x} \neq \mathbf{x}'}} \|\mathbf{x} - \mathbf{x}'\|$;
      $S_d \leftarrow \text{argmin}_{\mathbf{x} \in \pi_i} \{dist(\mathbf{x})\}$;
      $\mathbf{x}^w \leftarrow \text{argmax}_{\mathbf{x} \in S_d} \left\{ g^{tch}(\mathbf{x} | \mathbf{z}^{\mathbf{ideal}}, \mathbf{Z_i^s}) \right\}$;
      $S \leftarrow S \backslash \mathbf{x}^w$
  $P_{t+1}^c \leftarrow S$
else
  $S_I \leftarrow H_c \backslash S_c$;
  $\mathbf{F} \leftarrow$ non-dominated-sort(S_I);
  while $|S_c| < N$ **do**
    $S \leftarrow S \cup F_i$;
    $i \leftarrow i + 1$;
  Remove large infeasible solutions from S. CV is the constraint violation.
  while $|S| > N$ **do**
    $\mathbf{x}^w \leftarrow \text{argmax}_{\mathbf{x} \in F_{i-1}} \{CV(\mathbf{x})\}$;
    $S \leftarrow S \backslash \mathbf{x}^w$;
  $P_{t+1}^c \leftarrow S$
return P_{t+1}^c

2.3.5 Adaptive Evolutionary Algorithm based on Non-Euclidean Geometry (AGE-MOEA)

Panichella [2019] proposes a different approach to the crowding distance operator first proposed in NSGA-II while keeping the original framework of NSGA-II. It implements a survival score that "combines diversity and proximity of the non-dominated fronts" [Panichella, 2019, p. 596]. First, the author proposes a fast heuristic that estimates the geometry of the first non-dominated front, F_1, in each generation and then computes the proximity as the "distance between each population member and the ideal point" [Panichella, 2019, p. 596].

The diversity calculation uses the distance between the population members. The distance to compute the proximity and diversity corresponds to the L_p norm, whose exponent p depends on the geometry of the optimal Pareto front. Since the optimal Pareto front is unknown, the computation of p depends on each generation's geometry of F_1.

Definition 2.3.3 (The L_p norm (Thompson [1996])): *In a K-dimensional space $\mathbb{R}^K$, let v be a vector of size K, i.e. $v = (v_1, ..., v_K)$. Then the L_p norm of v in the K-dimensional space is defined by,*

$$\|v\|_p = (v_1^p + ... + v_K^p)^{1/p}, \quad p \geq 1 \tag{2.45}$$

□

[Panichella, 2019, p. 597] uses the same normalisation technique found in NSGA-III. However, it occurs only on the first non-dominated front. Therefore, the objectives of F_1 take on values between $[0, 1]$, and the objectives for the other non-dominated fronts, $F_{d>1}$, can have values greater than one.

In order to determine the geometry of the first non-dominated front, F_1, we have to find the value p of the L_p norm such that the corresponding unit hypersurface best fits the normalised objectives of the front F_1. The optimal hypersurface has an L_p norm where all points in the normalised objectives are equally distant from the ideal points.

It turns out that the extreme points in the front correspond to the intersections of the front with the objective axes [Panichella, 2019, p. 598]. I.e. the intersection point of F_1 with axis f_i^n has the objective value $f_i^n = 1$, and all other objectives f_j^n have the objective value $f_j^n = 0 \ \ \forall j \neq i$. Thus the distance from the intersection point, a_k, to the ideal point, z_k^{min}, is always equal to one for any chosen p exponent [Panichella, 2019, p. 598]. The problem consists of solving the following system of non-linear equations.

$$\begin{cases} (f_1(S_1)^p + ... + f_K(S_1)^p)^{1/p} = 1 \\ \qquad\qquad . \ . \ . \\ (f_1(S_m)^p + ... + f_K(S_m)^p)^{1/p} = 1 \end{cases} \tag{2.46}$$

where m is the number of points in the front F_1.

Panichella [2019] approximates the value of p by considering the central point in the first non-dominated front, F_1. As all the objectives are normalised, the central point C is the point in F_1 with the lowest perpendicular distance to the vector $\vec{\beta}$ delimited by the nadir point $z^{max} = 1$ and the ideal point $z^{min} = 0$.

$$C = \underset{f^n(S)}{\operatorname{argmin}} \left\{ d^{\perp}(f^n(S), \vec{\beta}) \right\}, \quad \forall\, S \in F_1 \tag{2.47}$$

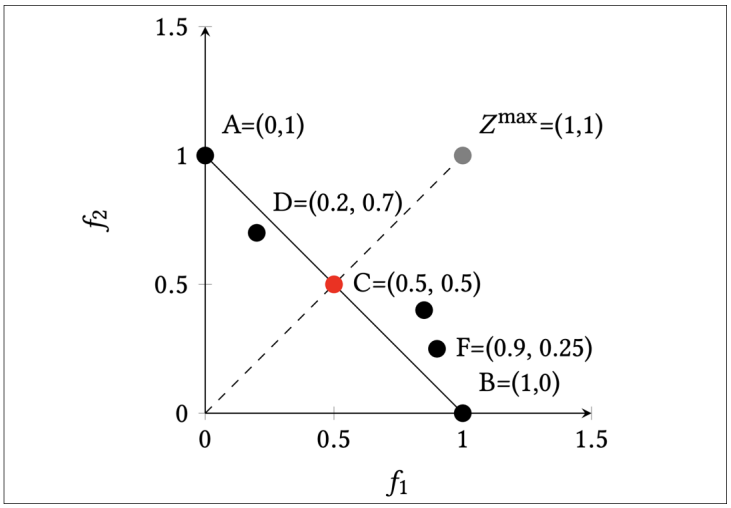


Figure 2.7. Example of the central point C of the normalised non-dominated front.

Figure 2.7 shows an example from [Panichella, 2019, p. 598] for calculating the central point C. The figure depicts two extreme points, $A = (0,1)$ and $B = (1,0)$, that lie on two objective axes, f_1^n and f_2^n, respectively. The central point C situates on the vector $\vec{\beta}$. Considering only the extreme points, A and B, and the central point, C, the problem consists of finding a p such that the L_p norm of C equals to one. I.e. $(C_1^p + C_2^p)^{1/p} = 1$. To find the value of p where C lies directly on the bisector of $\vec{\beta}$, we solve the following equation:

$$\begin{aligned} \left(\sum_{i=1}^{K} C_i^p \right) = 1 \quad & \Rightarrow K \cdot C_1^p = 1 \\ & \Rightarrow C_1^p = \frac{1}{K} \\ & \Rightarrow p = -\frac{\log(K)}{\log(C_1)} \end{aligned} \tag{2.48}$$

However, the central point in the first non-dominated front, F_1, can have a distance $d^{\perp}(C, \vec{\beta}) > 0$; therefore, the objective space coordinates may not be identical. [Panichella,

2019, p. 598] thus approximates the value of the exponent p using the following equation.

$$\begin{aligned} p &= -\frac{\log(K)}{\log(\frac{1}{K}\sum_{i=1}^{K} C_i)} \\ &= -\frac{\log(K)}{\log(K) - \log(\sum_{i=1}^{K} C_i)} \end{aligned} \tag{2.49}$$

To measure the diversity and proximity of $S \in F_1$, we compute the minimum distance compared to the other solutions in F_1 and the distance of the objective vector f^n to the ideal point, respectively. [Panichella, 2019, p. 599] writes that "a solution S with $proximity < 1$ dominates part of the unitary hypersurface of L_p", which point D shows in Figure 2.7, and "a solution S with $proximity > 1$ is more distant from the ideal point compared to points in the unit hypersurface of L_p", which point F shows in Figure 2.7.

The survival score of the solutions $S \in F_1$ merges the diversity and proximity, where they need to be maximised and minimised, respectively.

$$score(S) = \frac{diversity(S, F_1)}{proximity(S)} \tag{2.50}$$

where $diversity$ and $proximity$ are:

$$diversity(S, F_1) = \min_{S' \neq S \in F_1} \|f^n(S) - f(S')\|_p \tag{2.51}$$

$$proximity(S) \quad = \|f^n(S) - z^{ideal}\|_p = \|f^n(S)\|_p \tag{2.52}$$

2.4 Conclusion

In this chapter, we conveyed the literature related to this Thesis's goal. We started by showing the work done in the past in the scope of waste management, and we showed the main paper that we will improve to work with large-scale problems. Furthermore, we illustrated the novel model that Olapiriyakul et al. [2019] created and its constraints, which we will implement for the branch and cut and the evolutionary algorithm approach.

After that, we gave some insight into the different evolutionary algorithms we will use in the subsequent chapters. These include, NSGA-II (Section 2.3.1), NSGA-III (Section 2.3.3), U-NSGA-III (Section 2.3.3), C-TAEA (Section 2.3.4), and AGE-MOEA (Section 2.3.5). NSGA-II uses a crowding sort technique, making the original NSGA algorithm much more computationally efficient. Furthermore, we gave the procedure of the various algorithms by providing the pseudocode.

Chapter 3

Methodology

In this chapter, we extend the model proposed by Olapiriyakul et al. [2019] in section 2.1 by incorporating evolutionary algorithms to solve the proposed MILP problem faster, with a more diverse set of solutions, and on a larger scale. The following sections outline the data and methods used to support these claims.

3.1 Data

The data plays a vital role in supporting the claims made throughout this Thesis. There is currently a minimal amount of publicly available data on this topic. In order to create a dataset that is as close as possible to the real world, we used the data provided by Olapiriyakul et al. [2019] in their case study of Pathum Thani in Thailand, which provided us with health impact data and data on land-usage impacts of the various facilities. Moreover, the authors also gave data on the vehicle capacities and the facility capacities. However, it did not provide data on the facilities' construction and operational costs.

For the different facilities' construction costs, we used the data given by the incinerator in Giubiasco, Ticino, Switzerland, [Azienda cantonale dei rifiuti, 2010, p. 7]. It states that the waste-to-energy plant subsidy in Giubiasco, Switzerland was 40 million CHF in 2004, 42′050′290 CHF when adjusting for inflation for 2022. As the area of the incinerator plant is 40′000 m^2 square meters, the price per square meter is, thus, 1051.3 CHF.

For the landfill facilities, Elrabaya and Marchenko [2021] identified the total cost to landfill municipal waste in Sharjah, UAE, in 2021. They additionally consider the costs to construct, manage, excavate and consult the creation of an MSW landfill. The sum of these costs is 6′482′949 USD. Adjusting for inflation and converting to CHF, the total cost becomes 7′206′328.42 CHF. The MSW landfill occupies an area of 126′500 m^2

square meters. Thus the cost per square meter is 57 CHF. Unfortunately, there was no publicly available data for the cost of constructing sorting facilities. For this reason, we chose to use the cost per square meter of a landfill facility and add an equipment cost budget depending on the size of the facility of 250′000 CHF, 500′000 CHF, and 750′000 CHF for small, medium, and large sorting facilities, respectively.

For the operational cost, as no publically available data exists, we choose uniformly distributed random numbers that increase depending on the size of the facility. We choose $\mathcal{U}(100'000, 200'000)$, $\mathcal{U}(200'000, 400'000)$, and $\mathcal{U}(400'000, 600'000)$, for facilities of size small, medium, and large, respectively.
The DALY amounts seen in Table 3.2 show the health impact per person. Thus, we add a uniformly random population number for every vertex, V, on the generated graph, G, $\mathcal{U}(35'000, 80'000)$. In addition, we add $720'000 \cdot U(0,1)$ to the candidate nodes, as these should inhabit more people due to the larger land area. When calculating the population living near the facilities, which would be the affected area of the facility, we take the population of that node multiplied by the direct land usage of the facility of size l in squared kilometres. When calculating the population living near transport routes, we use the following equation, with $\alpha = \mathcal{U}(0.01, 0.05)$ and a graph $G = (V, E)$.

$$P_{ij} = \alpha(1-\alpha) \cdot (\|V^i - V^j\|_2 \cdot (V_P^i + V_P^j)) \tag{3.1}$$

The population on the link ij is adapted based on the population at vertice i and vertice j.

	Direct land use (m^2)			**Indirect land use (m^2)**		
Waste facility	Small size	Medium size	Large size	Small size	Medium size	Large size
Sorting facility	4′800	8′000	16′000	6′191	10′780	21′559
Incinerator facility	8′000	16′000	24′000	11′971	23′941	35′911
Landfill facility	80′000	160′000	192′000	127′770	255′525	335′295

Table 3.1. Direct and indirect land use impacts (Taken from [Olapiriyakul et al., 2019, p. 8])

Waste facility	Size	Capacity (tonne/day)	DALYs
Sorting facility	Small	50	0.07
	Medium	100	0.14
	Large	300	0.28
Incinerator facility	Small	50	5.95
	Medium	100	11.9
	Large	150	17.85
Landfill facility	Small	50	3.89
	Medium	100	7.78
	Large	150	11.66

Table 3.2. Facility impact on public health. (Adapted from [Olapiriyakul et al., 2019, p. 8])

Fleet types	Capacity (ton)	DALYs full load (per tkm)
Light truck	16	$5.62 \cdot 10^{-8}$
Heavy truck	32	$1.12 \cdot 10^{-7}$

Table 3.3. Transportation impact on public health (Adapted from [Olapiriyakul et al., 2019, p. 9])

	Construction cost		
Waste facility	Small size (CHF)	Medium size (CHF)	Large size (CHF)
Sorting facility	523′600	956′000	1′662′000
Incinerator facility	8′410′400	16′820′800	25′231′200
Landfill facility	4′560′000	9′120′000	10′944′000

Table 3.4. Facility construction costs.

3.2 Evolutionary algorithms

This section proposes an adaptation to the evolutionary algorithms' crossover and mutation operators to solve the MILP problem proposed by Olapiriyakul et al. [2019]. In order to select only the best candidate cities, we have to add zeroes into the solutions of the evolutionary algorithm in such a way that we narrow the search space to feasible solutions and lead the algorithm to the global optimum. We achieve this by implementing an adapted version to the crossover and mutation operators and adding a repair operator that can alter the solutions to explore the feasible region.

3.2.1 Binary decision variables y

For the binary decision variable crossover, we first apply a single-point crossover and then apply the procedure shown in figures 3.1 and 3.2. We sample random columns with replacements for each row, slowly bringing the search space to a feasible region. Afterwards, we set each row's selected column to zero to improve the solution space slowly.

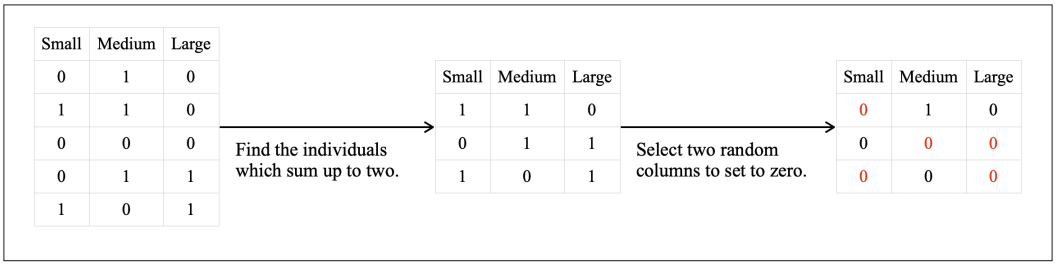


Figure 3.1. Method for the binary decision variable crossover, for individuals that sum to two.

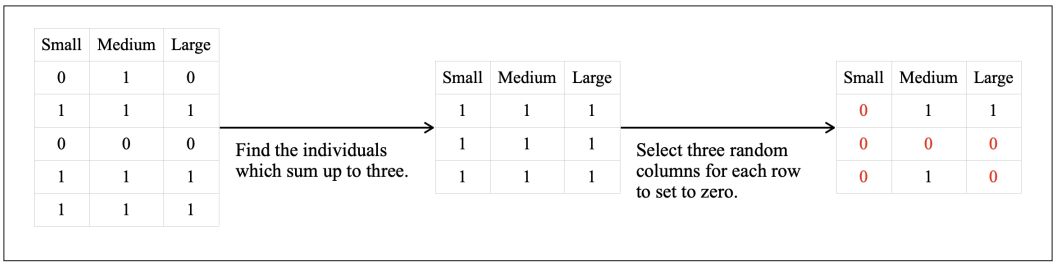


Figure 3.2. Method for the binary decision variable crossover, for individuals that sum to three.

Figure 3.3 shows the adaptation of the mutation operator by performing a typical binary mutation and then setting the non-mutated one values to zero in the rows that had a mutated one value. This mutation operator slowly moves the solution space into

the feasible region and can explore newly mutated individuals by staying in the feasible region.

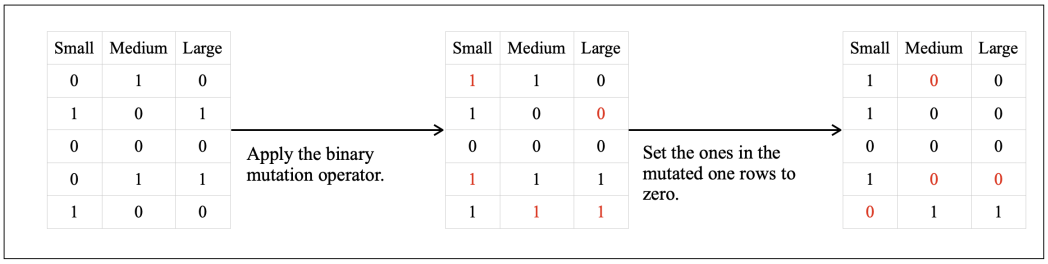


Figure 3.3. Method for the binary decision variable mutation.

Algorithm 10 gives the pseudocode for the binary y update procedure. We first apply the single-point crossover on the population, receiving a matrix of shape $|C| \times l_{size}$, which for example, would be of size 6×3, for an average city count of 2, as we would have 6 total candidate cities and 3 total facility sizes.
After that, we obtain the rows where the sum equals M, the sum of the rows to identify. We then sample random integers of size M for each row found and set the row and column combination to zero.

For the mutation, we sample points uniformly at random for all decision variables and create a copy of the population called P_{copy}. We then obtain the decision variables to flip and not to flip. After that, we mutate the decision variables in our flip category by applying a not operator for these in P_{copy}. Looking at the copied population, we will have some variables flipped to ones and some to zero. We select the flipped variables that equal one and set the row to zero in the original population. Then we overwrite the non-flipped variables in the copied population with the original population.

Algorithm 10: y update procedure for one generation t

Input : The probability of mutation p, probability of crossover c, population P, candidate sorting facilities J, candidate incinerator facilities K, and candidate landfill facilities K'.

Output: The binary crossover result X.

Crossover operator:

$l_{size} \leftarrow \sum L(J) \cap L(K) \cap L(K')$;

$C \leftarrow J \cup K \cup K'$;

$X_{|C| \times l_{size}} \leftarrow$ single-point-crossover(P, c);

for $(M = 2;\ M < 4;\ M = M + 1)$ **do**

$$\delta = \begin{cases} 1 & if\ \sum_{j=1}^{l_{size}} X_{ij} = M, \quad \forall i \in C; \\ 0 & otherwise \end{cases}$$

$\alpha \leftarrow$ Sample random integers with replacement of size $\delta_{\delta=1} \times M$;

$X_{\delta,\alpha} \leftarrow 0$;

$P \leftarrow X$;

Mutation operator:

Obtain probabilities uniformly for all binary variables:

$Z \leftarrow Z \in U[0,1]^{ij}, \quad \forall i, j \in P$;

$P^{copy} \leftarrow copy(P)$;

Get the variables to be mutated and not mutated:

$flip \leftarrow Z < p$;

$not\ flip \leftarrow Z \geq p$;

Apply a not operator on the variables to be flipped and add them to the copied population:

$P^{copy}_{flip} \leftarrow \neg P_{flip}$;

Set the rows of the flipped to one variables to zero in the original population:

$P^{i}_{P^{copy}_{flip,ij}=1} \leftarrow 0, \quad \forall i, j \in P$

Overwrite the non-flipped variables in the copied population:

$P^{copy}_{not\ flip} \leftarrow P_{not\ flip}$

return P^{copy}

3.2.2 Real decision variables f

Equation 3.2 shows the general matrix of the real number decision variables, f, where it conveys the amount of waste sent from collection centres i to sorting facilities j, where $n \in \mathbb{N}$ is the average amount of centres available.

$$F_{n\times n} = \begin{bmatrix} f_{i_1,j_1} & f_{i_1,j_2} & \cdots & f_{i_1,j_n} \\ f_{i_2,j_1} & f_{i_2,j_2} & \cdots & f_{i_2,j_n} \\ \vdots & \vdots & \ddots & \vdots \\ f_{i_n,j_1} & f_{i_n,j_2} & \cdots & f_{i_n,j_n} \end{bmatrix} \tag{3.2}$$

In order to satisfy the hard constraint 2.5, we have to normalise the matrix F by dividing each row by the amount of supply at collection centre i. This normalisation will allow us to make the sum of each row one and satisfy constraint 2.5. We achieve this by applying equation 3.3 to each row of the normalised matrix F.

$$F_i = \frac{F_i}{\sum_{j=1}^{n} f_{ij}}, \quad \forall i \in I \tag{3.3}$$

We apply this procedure in the real number single-point crossover operator and the mutation operator to quickly satisfy constraint 2.5 without spending time on infeasible solutions. For constraint 2.6, we apply the same procedure; however, we use a matrix that combines the incinerator and landfill real numbers, as seen in the matrix of equation 3.4. We normalise the rows by taking the sum of each column of the matrix in equation 3.2. Then we divide each row by the sum of that row to obtain a sum equal to one (equation 3.5).

$$F_{n\times 2n} = \begin{bmatrix} \overbrace{f_{j_1,k_1} & f_{j_1,k_2} & \cdots & f_{j_1,k_n}}^{K} & \overbrace{f_{j_1,k'_1} & f_{j_1,k'_2} & \cdots & f_{j_1,k'_n}}^{K'} \\ f_{j_2,k_1} & f_{j_2,k_2} & \cdots & f_{j_2,k_n} & f_{j_2,k'_1} & f_{j_2,k'_2} & \cdots & f_{j_2,k'_n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ f_{j_n,k_1} & f_{j_n,k_2} & \cdots & f_{j_n,k_n} & f_{j_n,k'_1} & f_{j_n,k'_2} & \cdots & f_{j_n,k'_n} \end{bmatrix} \tag{3.4}$$

$$F_j = \frac{F_j}{\sum_{k=1}^{2n} f_{jk}}, \quad \forall j \in J \tag{3.5}$$

In order to close facilities while staying in the feasible region of the constraints, we add a repair operator that repairs the solutions after the crossover and mutation operators. The proposed approach adds zeroes to the integer, x, and real, f, numbered decision variables. We achieve this by setting the columns to zero, where the sum of the rows of the binary decision variable, y, is zero. In addition, we also set the values

of f to zero where x is zero and vice versa. Afterwards, we use the same normalisation and division approach as in the real numbered, f, crossover and mutation to fix constraints 2.5 and 2.6. We then denormalise by multiplying the rows of f by the supply at collection centre i and fix the integer-numbered values, x, by applying equation 3.6 to fix constraint 2.10. To fix constraints 2.11 and 2.12, we apply the same equation for links jk and jk'.

$$x_{ij} = \left\lceil \frac{f_{ij}}{\overline{C}_{ij}} \right\rceil, \quad \forall i \in I, j \in J \tag{3.6}$$

This section's proposed approach helps us obtain a Pareto front of feasible solutions even on large-scale problems. The evolutionary algorithm is thus able to minimise the three objective functions instead of trying to find feasible solutions.

Algorithm 11: f update procedure for one generation t

Input : The population P, candidate sorting facilities J, candidate incinerator facilities K, and candidate landfill facilities K'.

Output: The updated solutions X.

$X_t \leftarrow$ selection(P);
Crossover operator:
Single-point-crossover: $f \leftarrow$ single-point-crossover(X_t^f);
$S \leftarrow D$;
$\tau_1 \leftarrow I$, $\tau_2 \leftarrow J$;
$t \leftarrow 1$;
while $t \leq 2$ **do**
  for $i \in \tau_1$ **do**
    for $j \in \tau_2$ **do**
      Normalisation: $f_{ij}^n \leftarrow \dfrac{f_{ij}}{S_i}$;
      Sum to one: $f_{ij}^n \leftarrow \dfrac{f_i^n j}{\sum_{j \in \tau_2} f_{ij}}$;
      Denormalise: $f_{ij} \leftarrow f_{ij}^n \cdot S_i$;
  Get new supplies: $S \leftarrow \sum_{i \in I} f_{ij}, \quad \forall j \in J$
  $\tau_1 \leftarrow j$, $\tau_2 \leftarrow K \cup K'$;
  $t \leftarrow t + 1$;
Mutation operator:
Apply mutation: $f \leftarrow$ mutation(f);
Do the same as in steps (4) – (15);
Repair operator:
$S \leftarrow D$;
$\tau_1 \leftarrow I$, $\tau_2 \leftarrow J$;
$t \leftarrow 1$;
while $t \leq 2$ **do**
  Get sum of binaries: $\beta \leftarrow \sum_{j=1}^{L(\tau_2)} X_t^{y_{ij}}, \ \forall i \in \tau_1$;
  Set f to zero where the sum of binaries are zero: $f_{\beta=0} = 0$;
  Set f to zero where integers x are zero: $f_{X_t^x=0} = 0$;
  Do the same as in steps (7) – (12);
  Get new supplies: $S \leftarrow \sum_{i \in I} f_{ij}, \quad \forall j \in J$
  $\tau_1 \leftarrow j$, $\tau_2 \leftarrow K \cup K'$;
  $t \leftarrow t + 1$;
$X_{t+1}^f \leftarrow f$;
return X_{t+1}

3.3 Graph Generation

This section gives insight into the generation of the graphs used throughout this Thesis. We first start looking into the generation of x and y coordinates and choosing the candidate and collection centre locations. Furthermore, we show the design choices of the amount of supply at each collection centre and the calculation of the distance to each city. Finally, we show a generated graph and explain the different types of symbols used.

We first create random instances by generating random x and y coordinates based on a seed ranging from 0 to 2^{32}. These instances, as well as the seeds, are then saved. We generate these x and y coordinates by sampling a random number and scaling it by a factor K; we select $K = 100$ as we chose a graph of size 100×100. The amount of x and y coordinates sampled is the average number of cities, n_c, and multiplying it by four as we have four different types of facilities. Equation 3.7 shows the generation of x and y coordinates.

$$\{x, y\} = K \cdot \mathcal{U}(0, 1) \tag{3.7}$$

For the locations, we select the n_c randomly created x and y coordinates for each collection centre, sorting, incinerator, and landfill candidate, respectively. In order to keep a good balance between the number of chosen candidate facilities and the supply amount, the supplies for each collection centre are selected uniformly at random, between ten and forty tons. Finally, calculating the distances between every city connected with every other city uses the Euclidean distance between the respective x and y coordinates.

Figure 3.4 shows an example generated graph for an average city amount of three, including a total daily supply of 60.194 tons. In addition, the red links show that a connection from one city to another has a distance of 60 km or more, the yellow links show a connection that has a distance of less than 60 km but greater than 40 km, and the green links show a connecting distance of less than 40 km.

Furthermore, we colour-coded the facilities such that the green triangles shown in the figure are the sorting candidate facilities, the red rectangles are the incinerator facilities, and the blue circles are the landfill facilities. When scaling up the graphs to more extensive graphs, for example, in figure 3.5 with a total daily supply of 358.615 tons, we see a lot of different connections and a lot of different random candidate city locations. In addition, we also see some overlapping in the facilities.

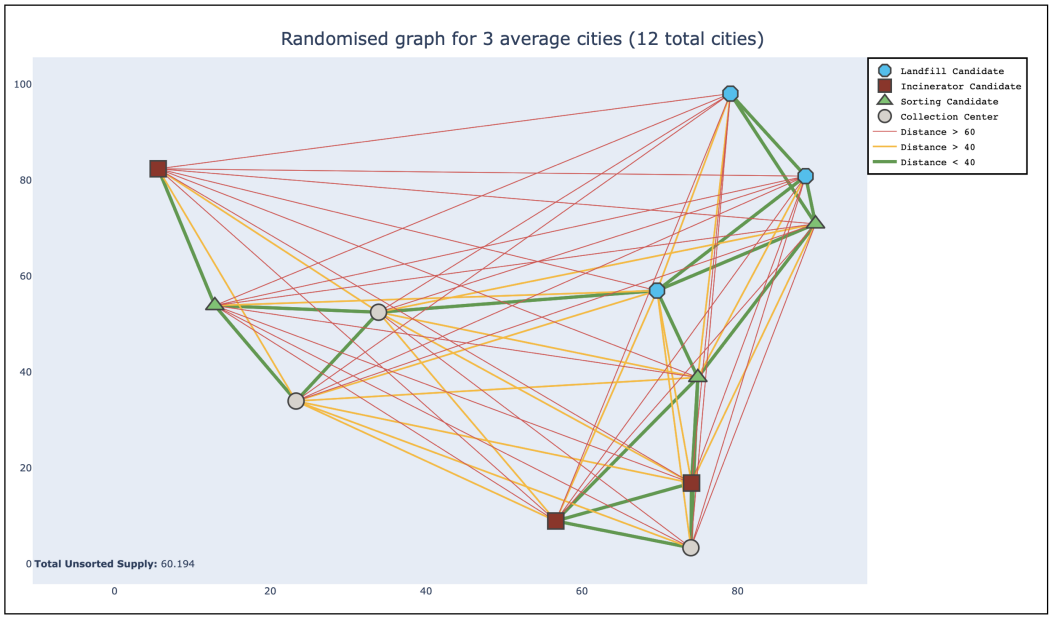


Figure 3.4. Graph generation with three number of average cities, 12 total cities.

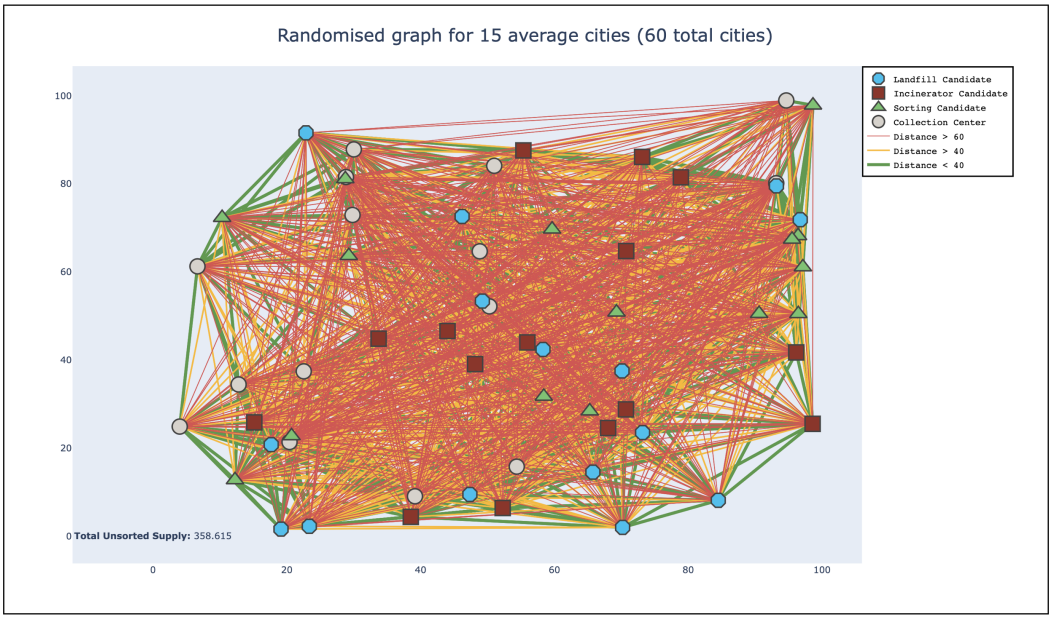


Figure 3.5. Graph generation with 15 number of average cities, 60 total cities.

3.4 Conclusion

In this chapter, we started by showing the data gathering procedure by looking into the case study of Pathum Thani by Olapiriyakul et al. [2019] and using the health impact, land usage, facility and vehicle capacity data. Moreover, we showed data gathered from real-life sources and research papers on the construction cost of incinerators and landfill facilities. Additionally, we displayed the cost of the sorting facilities, as there were no reputable sources for the construction cost of these facilities. Afterwards, we conveyed the method for the evolutionary algorithms, where we adapted the crossover and mutation operators to obtain solutions that are in the feasible regions of the problem and that have a reduced number of active facilities. Finally, we presented the graph generation procedure, which includes the generation of random graphs used to confirm the claims made throughout this Thesis.

Chapter 4

Implementation

This section covers the implementation approaches of the initially proposed MILP model by Olapiriyakul et al. [2019] with CPLEX 20.1.0 and the problems and solutions to the evolutionary algorithm implementation using the Pymoo multi-objective optimisation library by Blank and Deb [2020]. Finally, we give insight into the parameter choices of the various algorithms and show the Hypervolume performance indicator to measure the algorithms' performance.

4.1 Exact solution optimisation

As illustrated in the original paper proposed by Olapiriyakul et al. [2019], we use the CPLEX optimisation framework as an exact solution approach to single objective problems and an approximate solution approach for bi-objective and multi-objective problems. The CPLEX framework solves linear programming problems, mixed-integer linear programming problems, and quadratic objective problems. If the objective is a linear or convex quadratic function, CPLEX can solve them. The CPLEX callable library is written in C and allows for embedding into Python as it can call C functions for easier use by the programmer. We use this library as a baseline for the approach by Olapiriyakul et al. [2019] as it is state-of-the-art for optimising linear programs.

Figures 4.1 and 4.2 show the CPLEX solutions to the problem in figure 3.4. As shown by Olapiriyakul et al. [2019], we obtain the single objective solutions and then solve for the bi-objective and the three objective problem formulations. As this approach only gives a single best solution for each problem formulation, we will always receive seven solutions for every problem.

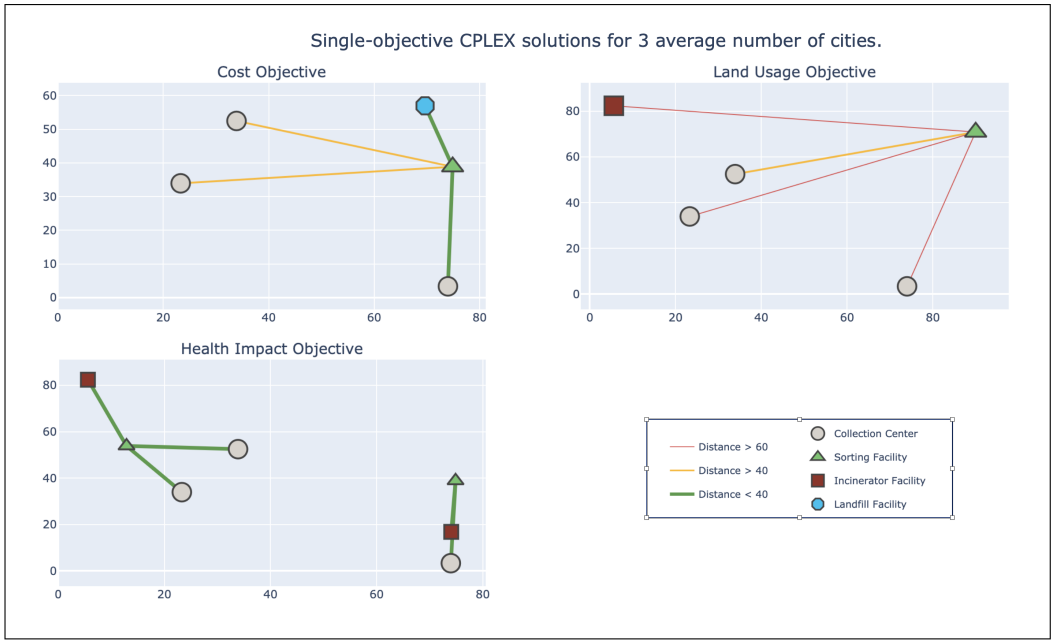


Figure 4.1. Single objective solution with CPLEX for three average cities.

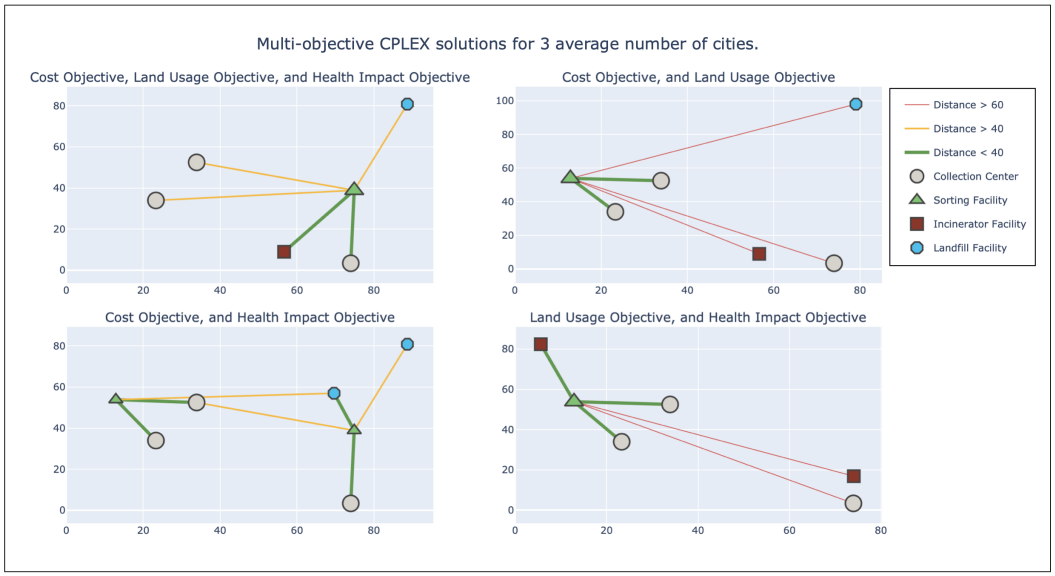


Figure 4.2. Multi-objective solution with CPLEX for three average cities.

4.2 Evolutionary Algorithms

We use the Pymoo framework (version 0.5.0), Blank and Deb [2020], for the evolutionary optimisation, which allows for a straightforward implementation of the constrained MILP problem by Olapiriyakul et al. [2019]. Furthermore, it allows for a fast implementation of multiple evolutionary algorithms. We start by implementing the problem and its constraints in a vectorised form, allowing for a faster execution time. Furthermore, we set the upper bound of the integer decision variables, x, and real numbered decision variables, f, to equations 4.1 and 4.2, respectively.

$$x_{ub} = \left\lceil \frac{\sum_{i \in I} D_i}{\overline{C}_{ij}} \right\rceil, \quad \forall j \in J \tag{4.1}$$

$$f_{ub} = \sum_{i \in I} D_i \tag{4.2}$$

The lower bound for all decision variables is zero. In the worst case, we allow for the transportation of the total supply on a single link from a sorting facility to a landfill or incinerator. The original vectorised implementation contained three significant problems:

1. It was finding a minimal amount of solutions that are in the feasible region.

2. It was slow at finding the feasible region of the solution space.

3. The feasible solutions found used all available sorting, incinerator, and landfill candidate facilities, which induces very high objective costs.

For the first problem, instead of prioritising selecting feasible solutions, we added a penalty of 10^8 multiplied by the amount of constraint violation to the objective functions and, as such, found more solutions, even if they were infeasible. The next goal was to move into the feasible region faster. To move into the feasible region faster, we added the real numbered crossover and mutation operators found in the methodology in section 3.2.2. For the base mutation, we use a polynomial mutation which uses the same probability distribution found in Deb et al. [2007], where u_i is a random number between 0 and 1.

$$\mathcal{P}(\beta_{qi}) = \begin{cases} (2u_i + (1-2u_i)(1-\beta_{(1,i)}^{\eta_c}))^{1/\eta_c} - 1, & \text{if } u_i \leq 0.5; \\ 1 - (2(1-u_i) + 2(u_i - 0.5)\beta_{(2,i)}^{\eta_c})^{1/\eta_c}, & \text{otherwise.} \end{cases} \tag{4.3}$$

The spread factor β_i is the ratio of the absolute difference in selected mutation values

to that of the upper bound and lower bound.

$$\beta_{(1,i)} = \left| \frac{x_i^t - x_{lb}}{x_{ub} - x_{lb}} \right| \tag{4.4}$$

$$\beta_{(2,i)} = \left| \frac{x_i^t - x_{ub}}{x_{ub} - x_{lb}} \right| \tag{4.5}$$

The distribution index η_c is any non-negative real number. A large value of η_c gives a higher probability of creating values closer to the non-mutated values. In contrast, a small value of η_c allows the creation of distant values to the non-mutated values.
After obtaining β_{qi} from equation 4.3 the calculation of the new mutated values is as follows:

$$x_i^{t+1} = x_i^t + (x_{ub} - x_{lb})\beta_{qi} \tag{4.6}$$

This probability distribution helps us create new values similar to the old ones. Both of these operators allowed us to meet the hard constraints of the problem that required to be precisely equal to each other. These allowed us to obtain feasible solutions; however, these still used all available sorting, incinerator, and landfill candidate facilities.

In order to cancel out redundant facilities, we apply the repair operator outlined at the end of section 3.2.2, which allows searching for facility cancellations while staying in the feasible region of the search space.
Furthermore, the introduction of zeroes produced a division by zero for the normalisation of the incinerator and landfill values, as there was no supply flowing into some sorting centres. To fix these issues, in the adapted crossover and mutation, we set the row of the sorting facility with zero supply flow to the incinerators K and landfills K' to zero. We also set the flow supply of the sorting facility with zero supply flow to one to create a division of one. Algorithm 12 delineates the updated normalisation procedure.

This approach gave satisfactory results for smaller problems; however, there was a violation of the binary constraints 2.13 – 2.15 on large-scale problems. These violations had multiple reasons,

1. There needed to be a higher number of facilities open in order to satisfy the amount of supply.

2. The standard crossover and mutation operators introduce multiple sizes into the solution space without removing the old sizes.

For these reasons, we decided to adapt the binary crossover and mutation to the one proposed in section 3.2.1.

In order to find the most performant algorithm without knowing the Pareto front, we

Algorithm 12: Updated normalisation procedure

$S \leftarrow \sum_{i \in I} f_{ij}, \quad \forall j \in J;$
for $j \in J$ **do**
  for $k \in K \cup K'$ **do**
    if $S_j = 0$ **then**
      $f_{jk} \leftarrow 0;$
  $S_j \leftarrow 1;$
Apply Normalisation:
for $j \in J$ **do**
  for $k \in K \cup K'$ **do**
    $f^n_{jk} \leftarrow\leftarrow \frac{f_{jk}}{S_j};$

use the hypervolume indicator first proposed by Zitzler and Künzli [2004]. The hypervolume indicator indicates the quality of a non-dominated set. It does this by using a reference point, usually set to one, after normalising the objectives using the nadir and ideal points, i.e. the worst and best points found, respectively. The goal of the hypervolume is to maximise the dominated area, as this means that the points are closer to the ideal point. Figure 4.3 shows an example of the hypervolume indicator, adapted from Fonseca et al. [2006]. It illustrates a dual objective case with three points where the dominated area is grey.

We use a tolerance on the objective space for the termination criterion to terminate once we reach a converged state. It calculates the change from the last ideal and nadir points to the current ideal and nadir points. The parameters include a sliding window that checks the last γ points at every iteration t, a maximum amount of generations, g, and a tolerance value, f_{tol}, that stops the algorithm if there has not been a significant change between the last and current ideal and nadir points.

Tables 4.1 and 4.2 summarise the parameters used for the evolutionary algorithm and the termination criterion for large and small-scale problems. We decrease the mutation probability for the evolutionary algorithm to reduce the exploration of the vast search space. Furthermore, we use the Riesz s-energy approach (Section 2.3.2) for the reference points implemented in Pymoo from Blank et al. [2021]. We use this approach as we want a solution for each population member on each reference point to maximise the number of solutions.

Figures 4.4 and 4.5 show the best solutions of the evolutionary algorithms for three and 15 cities, respectively, based on the hypervolume of the ideal points, which are the single objective solutions found by CPLEX, and the nadir points, which are the worst points found by all algorithms. For figure 4.4 we see quite the difference in solutions compared with the previous solution found by using the min-max approach with CPLEX in figure 4.2. Furthermore, in figure 4.5, we see the use of more sizes than in the previous figures. We use large, medium, and small facilities for incinerators and sorting facilities. We only use medium and small facilities across the algorithms for the landfill facilities.

Evolutionary algorithm parameters				
Problem size	*Population size*	*Reference points*	*Crossover probability*	*Mutation probability*
Small scale (≤ 10)	200	200	80%	1%
Large scale (> 10)	200	200	80%	0.5%

Table 4.1. Parameters used for the evolutionary algorithms, for large and small scale problems.

Termination criterion parameters				
Problem size	*Termination tolerance (f_{tol})*	*Sliding window amount (γ)*	*Sliding window iteration (t)*	*Maximum amount of generations (g)*
Small scale (≤ 10)	0.01	30	5	1000
Large scale (> 10)	0.01	30	5	1000

Table 4.2. Parameters used for the termination criterion, for small scale and large scale problems.

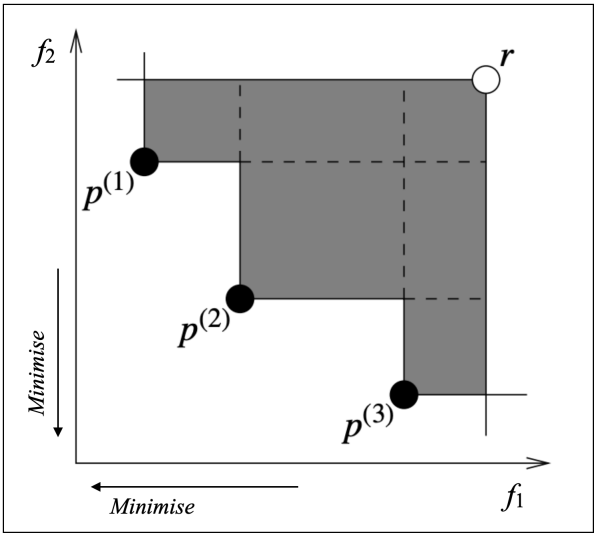


Figure 4.3. The hypervolume indicator shown for a dual objective problem.

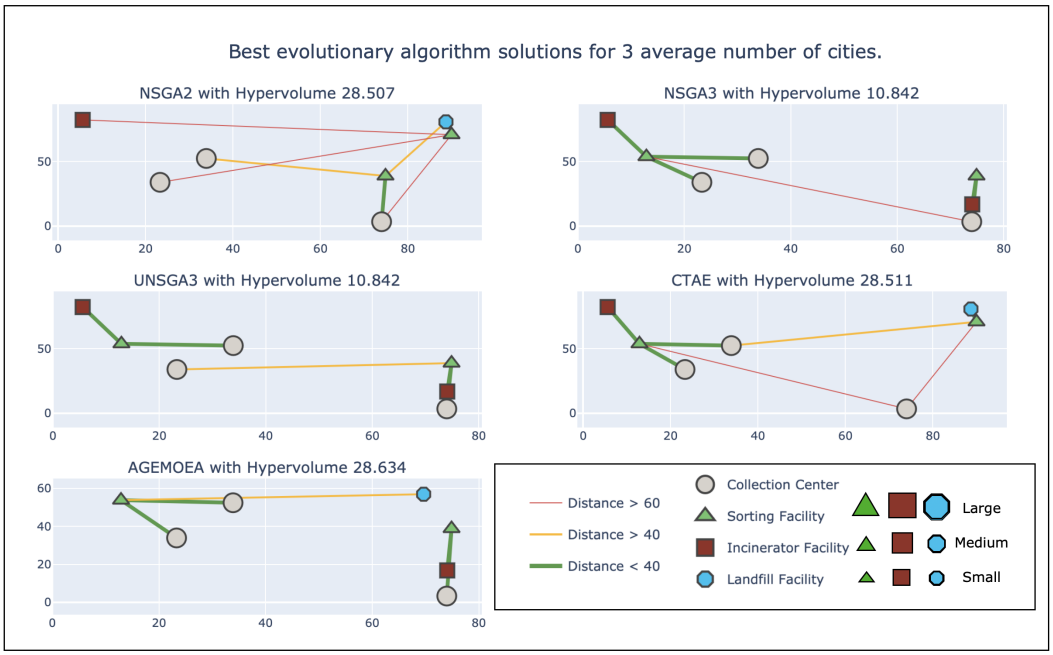


Figure 4.4. Evolutionary algorithm solutions for three cities.

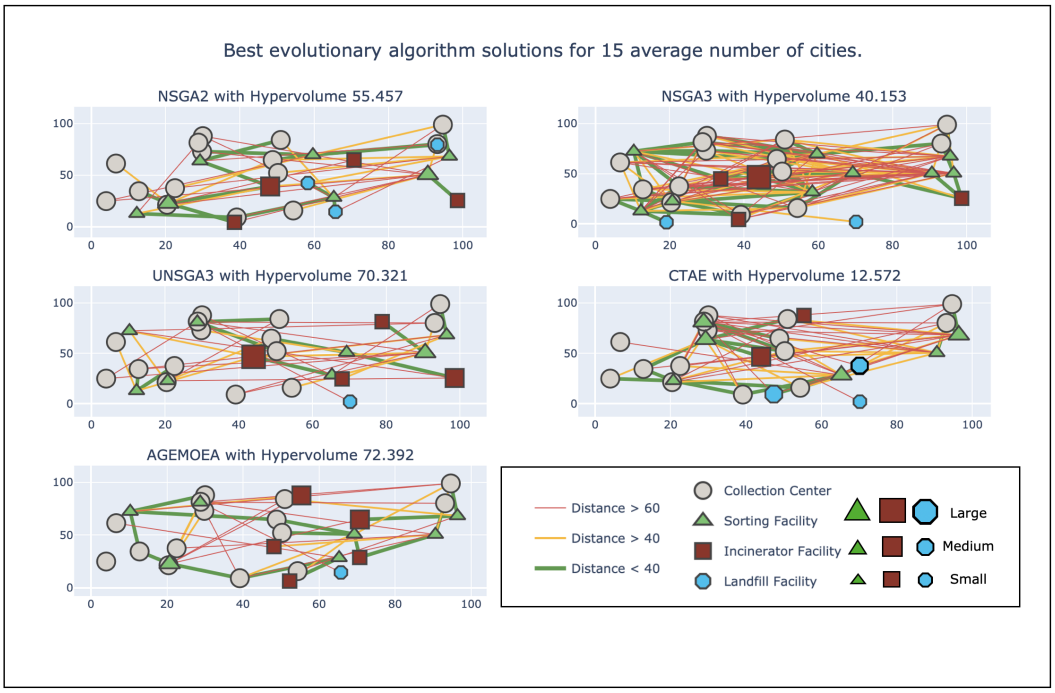


Figure 4.5. Evolutionary algorithm solutions for 15 cities.

4.3 Conclusion

In this chapter, we illustrated the implementation of the branch and cut algorithm using the CPLEX algorithm and provided an example of a solution to the randomised graph generated in section 3.3. Furthermore, we gave insight into the evolutionary algorithms' implementation, calculating the upper and lower bound for the integer and continuous variables, x and f, respectively. Moreover, we showed the updated normalisation to handle division by zero cases. Finally, we gave the evolutionary algorithms' parameter choices, providing the population amount, reference point amount, crossover probability, and mutation probability. We also gave insight into the hypervolume performance measure to find the difference between the solutions without providing a Pareto front.

Chapter 5

Empirical Study

This chapter presents the results obtained to validate the claims made throughout this Thesis. The claims we made in this Thesis included:

1. We are decreasing the time needed to solve large-scale multi-objective MILP problems in waste management.
2. We are obtaining reasonable solutions close to the CPLEX MILP model.
3. We are creating trade-offs over the three objectives by obtaining a Pareto set of diverse solutions.

We ran the experiments with the CPLEX and evolutionary algorithms on two to ten average numbers of cities, with ten randomised graphs per average city and three seeds per graph. We then went from 15 to 30 average number of cities and a final experiment on 40 average number of cities. For these, we only used the evolutionary algorithms and the same graph construction count with three seeds. In total, we ran 2370 experiments, with 1620 being the experiments from two cities to ten cities and 750 being the experiments from 15 cities to 30 cities and 40 cities.

We calculated the hypervolume percentage for each multi-objective solution using the ideal points from the CPLEX solutions for the small-sized problems and the ideal points from the evolutionary algorithm solutions for the large-sized problems. Moreover, we calculated the nadir points for both approaches as the worst points obtained in that run. Table 5.1 shows a summarisation of the construction of experiments.

Experiment construction				
Number of average cities	*Algorithms*	*Number of generated random graphs*	*Seeds*	Total number of experiments
2	CPLEX and Evolution	10	3	180
3	CPLEX and Evolution	10	3	180
4	CPLEX and Evolution	10	3	180
5	CPLEX and Evolution	10	3	180
6	CPLEX and Evolution	10	3	180
7	CPLEX and Evolution	10	3	180
8	CPLEX and Evolution	10	3	180
9	CPLEX and Evolution	10	3	180
10	CPLEX and Evolution	10	3	180
15	Evolution	10	3	150
20	Evolution	10	3	150
25	Evolution	10	3	150
30	Evolution	10	3	150
40	Evolution	10	3	150
				Total: 2370

Table 5.1. Construction of the experiments for the empirical study.

5.1 Time observations

Figure 5.1 shows the average time for the CPLEX optimiser (Branch and cut) and the evolutionary algorithms described in 2.3.1 – 2.3.5 (NSGA-II, NSGA-III, U-NSGA-III, C-TAEA, and AGE-MOEA). We took the average of all solved graphs with three seeds to obtain an average time for every number of cities. We scaled the results onto a log scale to see them more clearly. The figure shows that the branch and cut algorithm increases exponentially with the number of cities, and the evolutionary algorithms increase polynomially. Additionally, the branch and cut algorithm takes two orders of magnitude longer than all the evolutionary algorithms for ten cities. We also see that C-TAEA is as slow on average for smaller problems as the branch and cut algorithm. In addition, NSGA-III and U-NSGA-III are the fastest algorithms to reach a converged state on large problems. Moreover, NSGA-II and AGE-MOEA were the fastest algorithms on average for smaller problems and turned into the slowest evolutionary algorithms on more extensive problems.

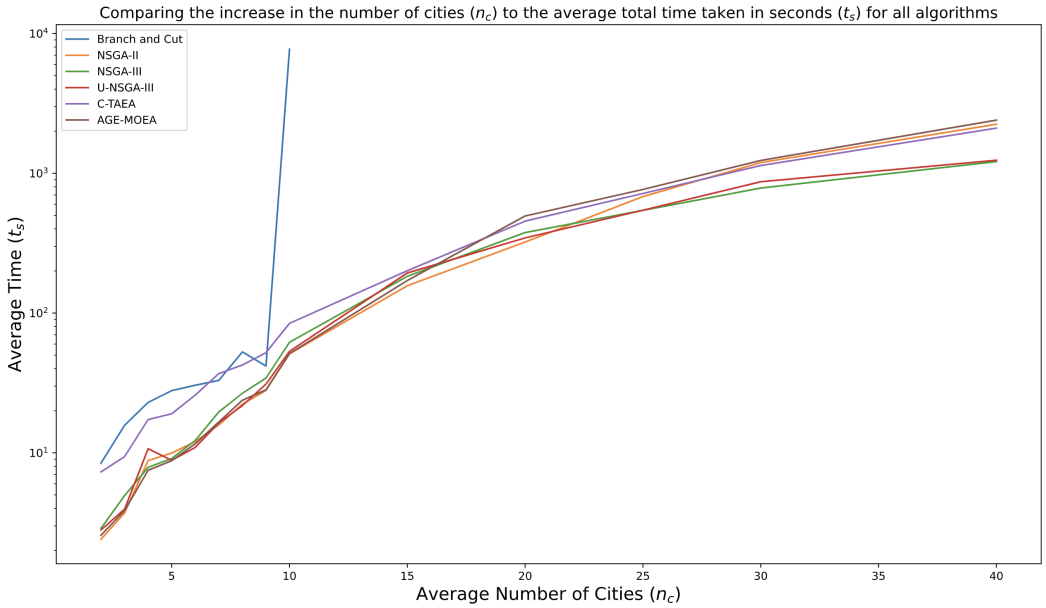


Figure 5.1. The average total time taken shown on a log scale for all algorithms.

5.2 Hypervolume results

We took the average hypervolume for each city number, graph, and algorithm seed. Figure 5.3 shows the result for the small-scale problems where we use all the algorithms, and figure 5.4 shows the results for the large-scale algorithms. The MILP algorithm has the same average hypervolume as the evolutionary algorithms for problems with fewer cities. Additionally, as the problem size increases, the hypervolume of all the evolutionary algorithms is much higher. The hypervolume for NSGA-III is twice as high as the MILP solution for the nine city problems, and the other evolutionary algorithms behave similarly. For the large-scale problems, NSGA-II and AGE-MOEA reign supreme for 40, 30 and 25 cities. U-NSGA-III is the third best, finding the best results on average for 15 cities. C-TAEA provides the worst results for 15, 20, 25, and 30 cities. However catches up on 40 cities, with it being a bit better on average than NSGA-III.

5.3 Number of solution results

As before, we obtain the average number of solutions for each city number, graph, and algorithm seed. Figure 5.2 gives information on the average number of solutions for small-scale problems, and figure 5.5 illustrates the average number for large-scale problems. Figure 5.2 conveys that the NSGA-II and AGE-MOEA algorithms obtain the most solutions, with the highest number of solutions obtained by NSGA-II with six cities. NSGA-III and U-NSGA-III obtain the lowest amount of solutions on average. In addition, the CPLEX MILP always has the same number of seven solutions because we found one solution for each optimisation, including single-objective, bi-objective, and tri-objective problem formulations. Figure 5.5 exhibits the same trend as in the small-scale problems, with fewer solutions as the problems get larger for all algorithms.

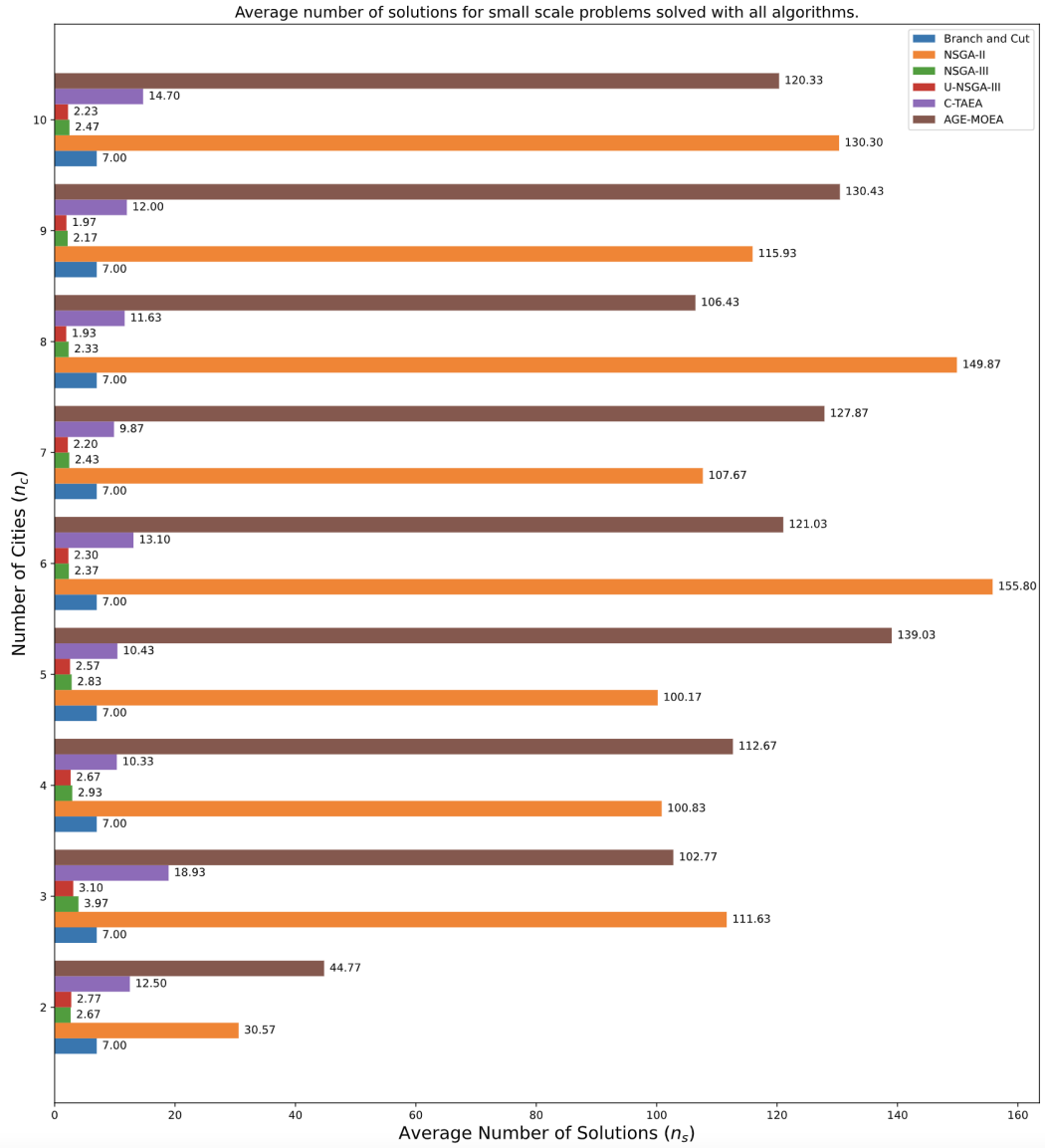


Figure 5.2. The average number of solutions (n_s) for all number of cities (n_c) from two – ten.

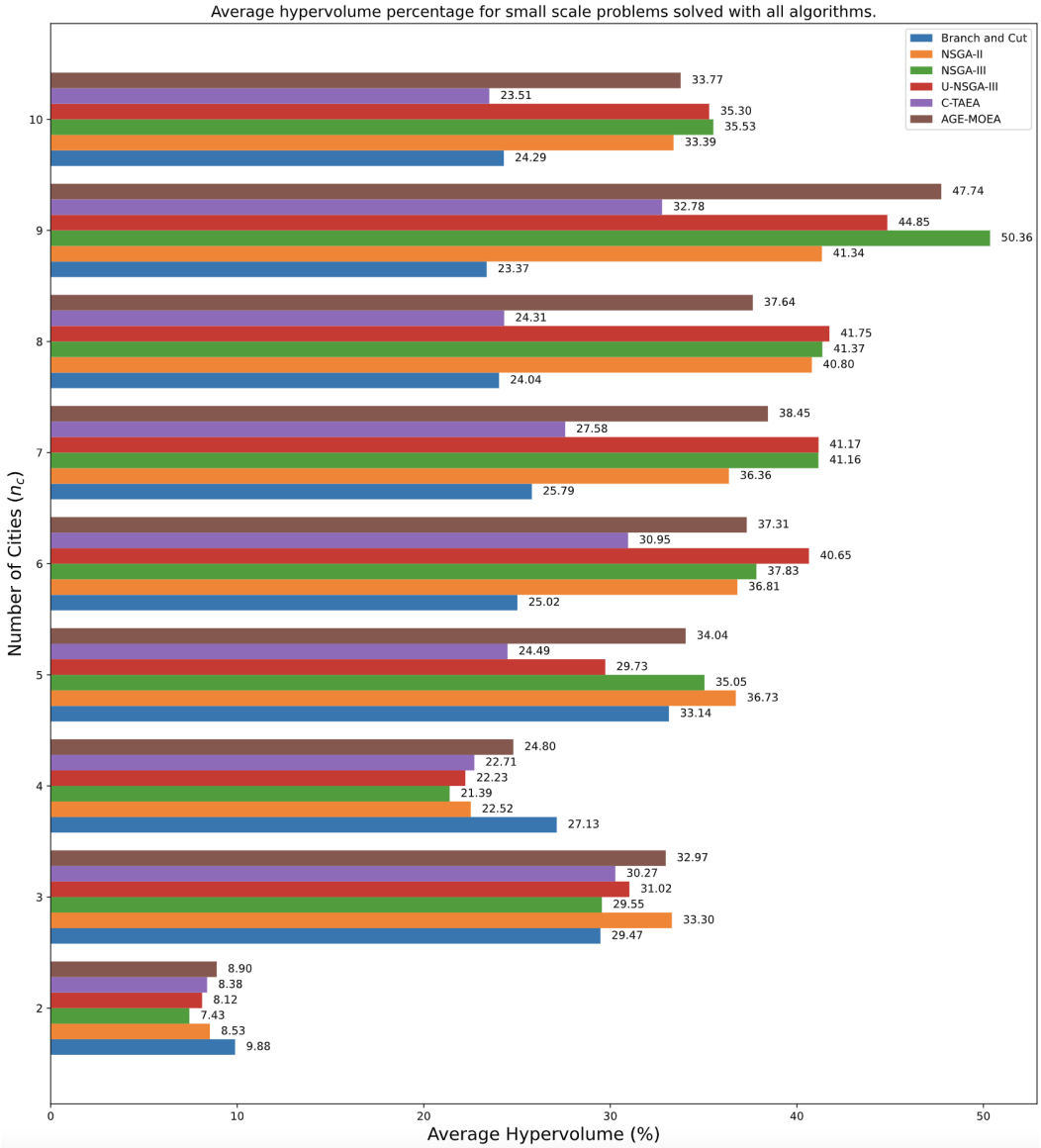


Figure 5.3. The average hypervolume shown in percentages for small problems solved with all algorithms for cities 2 to 10.

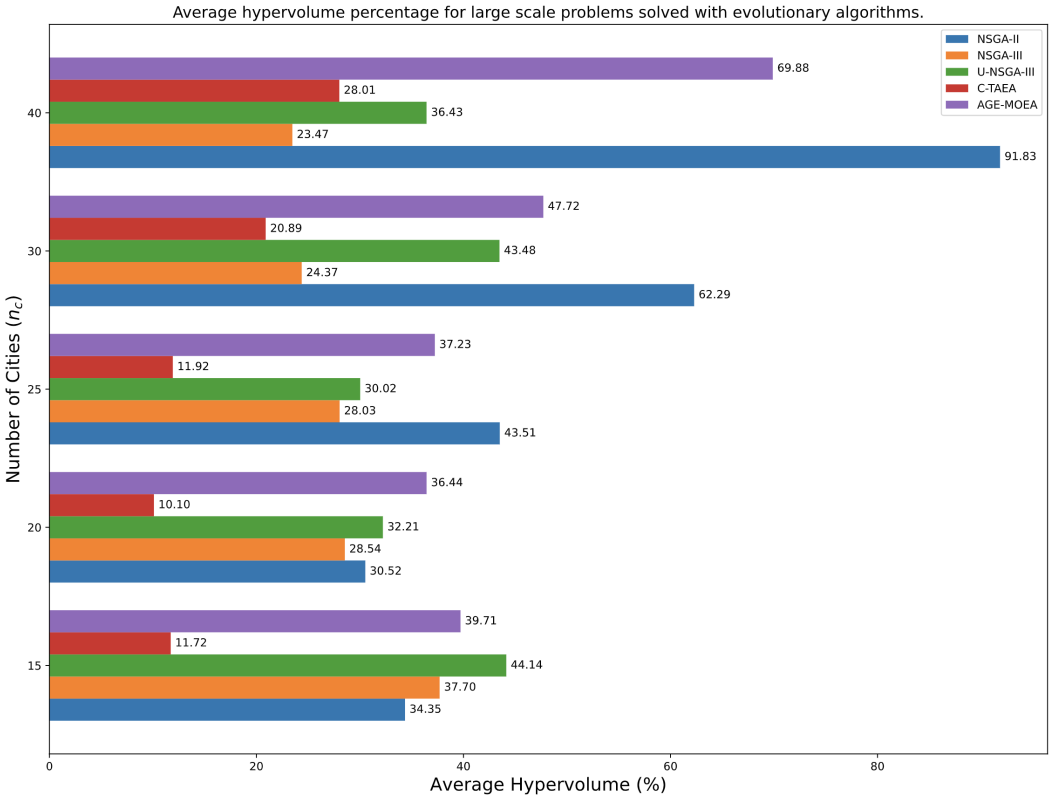


Figure 5.4. The average hypervolume shown in percentages for large problems solved with evolutionary algorithms for cities 15 – 30 and 40.

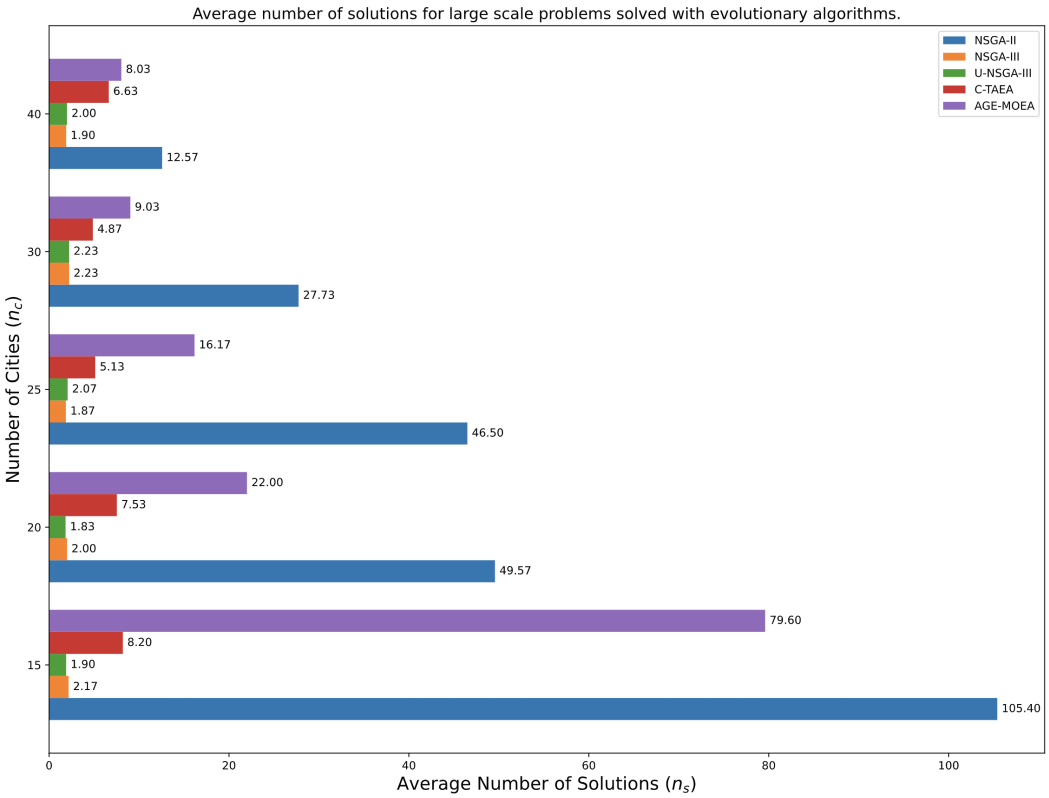


Figure 5.5. The average number of solutions (n_s) for all number of cities (n_c) from 15 – 30 and 40.

5.4 Solution results

Figures 5.6 – 5.8 convey the distribution of the total cost, land usage, and health impact obtained for each average city count, graph, and algorithm seed. Figure 5.6 shows this distribution for small-scale problems, from two to five average cities. Figure 5.7 illustrates the distributions for medium-sized graphs of six to ten cities. Finally, figure **??** displays the distributions for large-scale graphs of sizes 15 to 30 average cities, which use all evolutionary algorithms. For 40 cities, we display two versions in figure **??**: the original version that uses all the evolutionary algorithms solutions and a modified version that removes the C-TAEA algorithm.

For all the boxplots, the straight line shows the median value of the objective solutions, and the dotted line displays the mean value. Furthermore, three axes exist for each city plot, one the total cost (f_c) given in Swiss francs (CHF). The average land use stress (f_u), calculated by using equation 2.3, and the total public health impact (f_h) that uses DALYs.

In figure 5.6a, we do not see many variations in the solutions obtained for all algorithms. Furthermore, the means of the cost objective are very similar, and the medians are on the same level. The CPLEX branch and cut has the lowest interquartile range (IQR) for this objective, the highest minimum value, and the lowest maximum value. For the land use objective, the spread of the solutions is very similar, and the interquartile ranges do not differ much. All solutions found the same minimum value in the public health impact objective. However, the branch and cut algorithm again finds the lowest maximum value. For all objectives, we do not see any outliers.

In figure 5.6b, we already see a vast difference compared with 5.6a. The first very striking event is that the branch and cut algorithm on the cost and land use objective shows a meagre IQR, which means that the spread of the solutions is also shallow. In addition, the C-TAEA algorithm also has a low IQR for the land usage objective. U-NSGA-III finds the highest valued outliers and also the highest amount of upper quartile outliers. For the public health impact, all algorithms show a minimal IQR. The branch and cut algorithm has the lowest mean for all solutions for the total cost and the highest for land use and health impact. Figure 5.6d is very similar to figure 5.6b, with a very limited IQR for the cost and land use objectives.

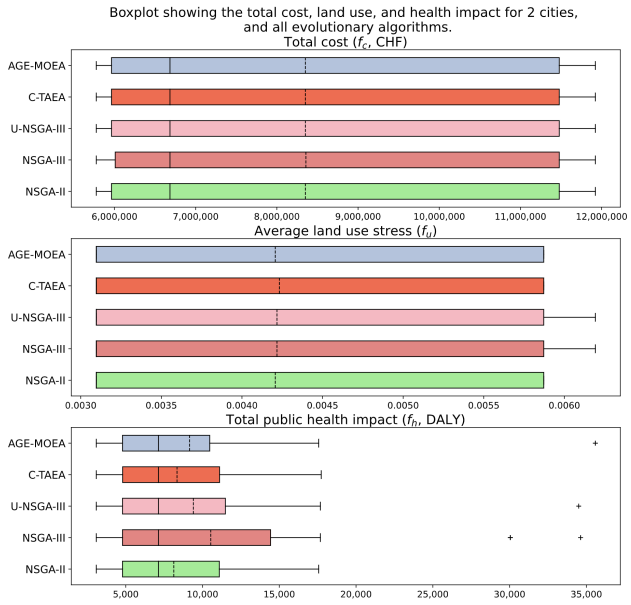
For figures 5.6c, the branch and cut algorithm obtains an extensive IQR for the cost objective, which is very similar to the U-NSGA-III algorithm. However, with the benefit of finding lower-cost solutions compared to U-NSGA-III. In the land use objective, the branch and cut has a limited range with no upper 25% and lower 25% of values existing. On the other hand, the evolutionary algorithms are very similar in achieving high

interquartile ranges.

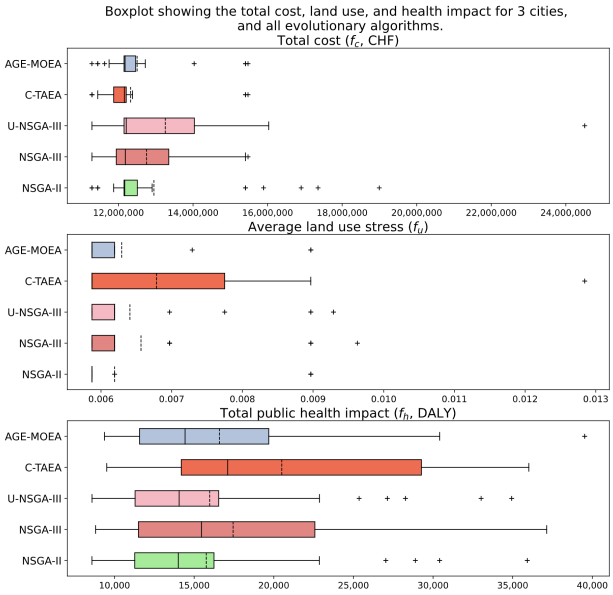
When looking into the medium-sized solutions in figure 5.7, we see that the interquartile ranges start to differ a lot with the C-TAEA algorithm achieving an immense IQR for the land use objective in figures 5.7a, 5.7b, and 5.7d. The evolutionary algorithms obtain a very high spread of solutions. However, they also obtain many more outliers than the branch and cut algorithm. For example, in figure 5.7d, in the land use objective, the AGE-MOEA, U-NSGA-III and NSGA-III algorithms find many outliers with AGE-MOEA finding a total of six outliers with a low spread of solutions. For figure 5.7e, all evolutionary algorithms find a higher spread of solutions or a lower value than the branch and cut algorithm.

For large problems, the spread of solutions becomes much less, with the highest IQR found in figure 5.8a by the C-TAEA algorithm. NSGA-II also finds a high solution spread for the average land usage objective. For figure 5.8b, a much lower IQR for all evolutionary algorithms exists, with NSGA-II finding the lowest average cost across all the different evolutionary algorithms. In addition, NSGA-III finds three very high outliers across all three objectives. In figure 5.8c, the spread of the solutions starts to increase again, with the IQR being very similar across all three objectives for all algorithms. Interestingly, there are many trade-offs across all of the evolutionary algorithms, with many being better in two objectives but one objective being worse than all the other algorithms. Figure 5.8d displays a small spread of solutions, as also seen in figure 5.8b. NSGA-II shows the smallest spread, finding the minimum solutions over all the objectives and having the lowest average solution value for three objectives. Furthermore, NSGA-II does not have many high outliers, with the highest amount of outliers found by U-NSGA-III and C-TAEA, respectively.

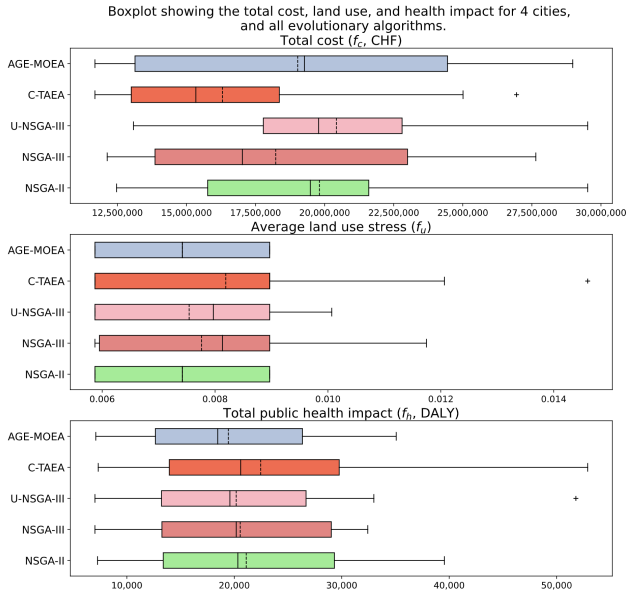
For the 40 city problem, in figure 5.8e, the NSGA-III, C-TAEA, and U-NSGA-III algorithms, to stay in the feasible region, start to select a lot of landfill facilities, which induces a very high land usage. NSGA-II still finds the minimum across all three objectives. Furthermore, the IQR of the solutions found by NSGA-II are all low enough to be minimal solutions for the other algorithms. U-NSGA-III, NSGA-III, and C-TAEA have the highest spread of solutions according to figure 5.8e.



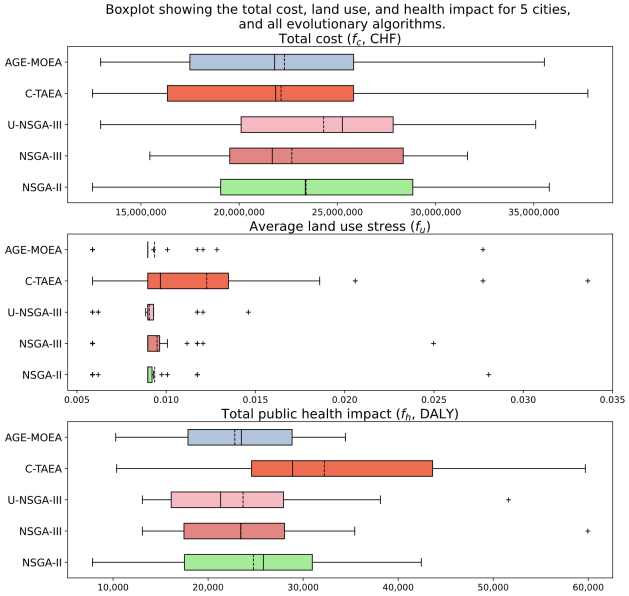
(a) Boxplot for two cities.



(b) Boxplot for three cities.

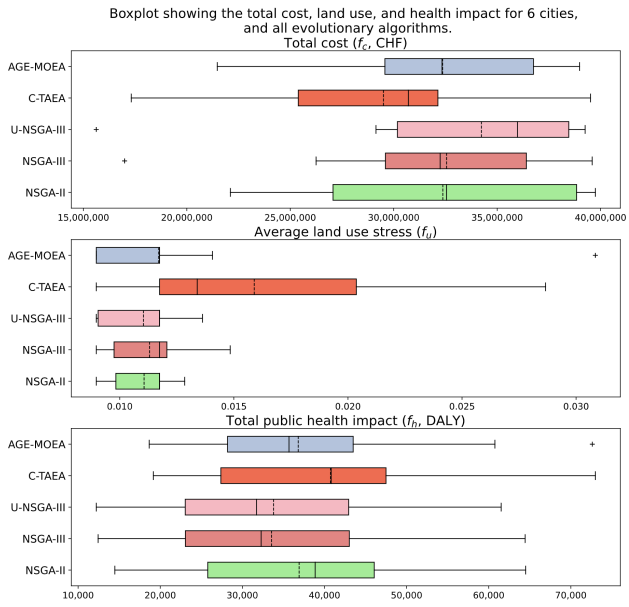


(c) Boxplot for four cities.

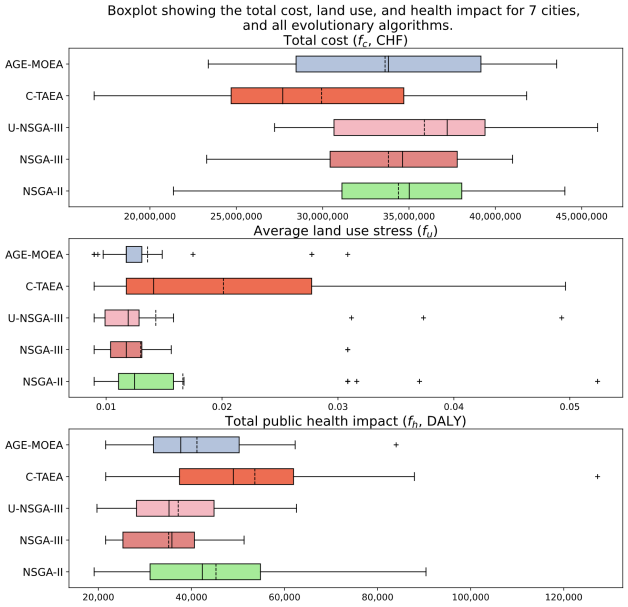


(d) Boxplot for five cities.

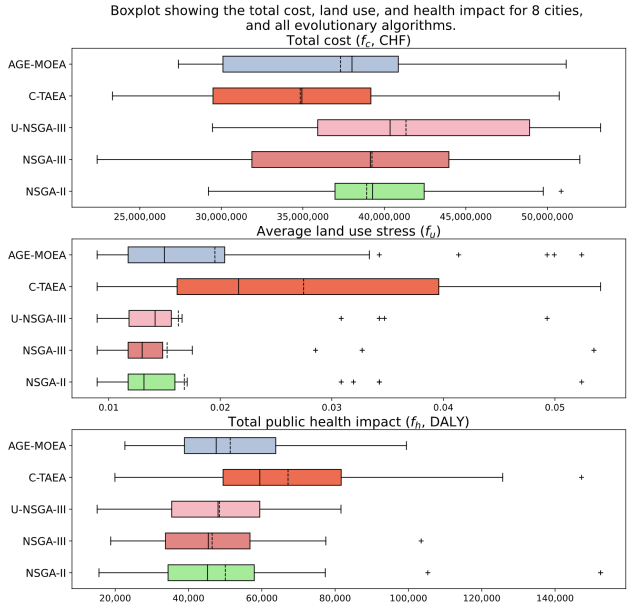
Figure 5.6. Boxplots for small-scale problems showing the total cost, land usage stress and health impact for all algorithms.



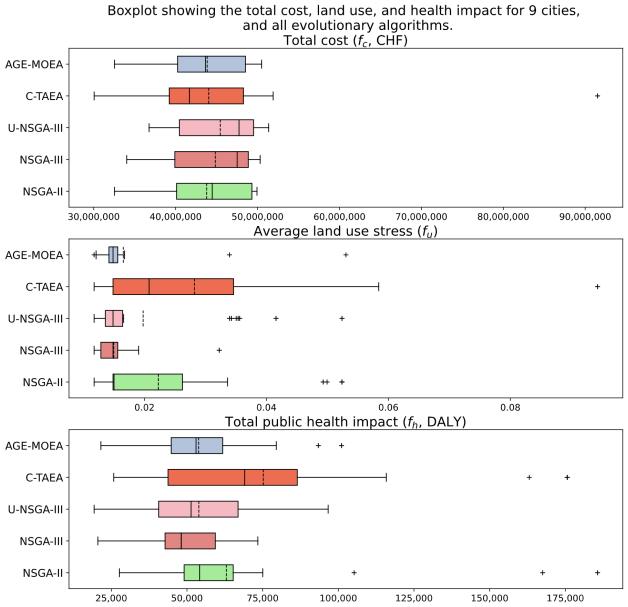
(a) Boxplot for six cities.



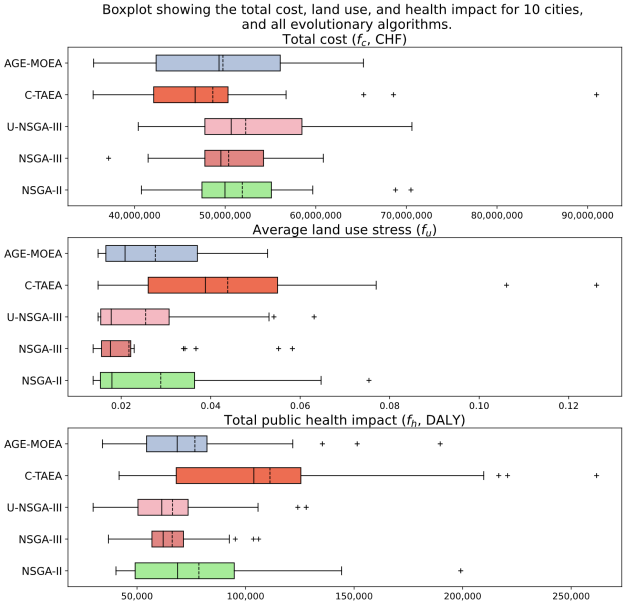
(b) Boxplot for seven cities.



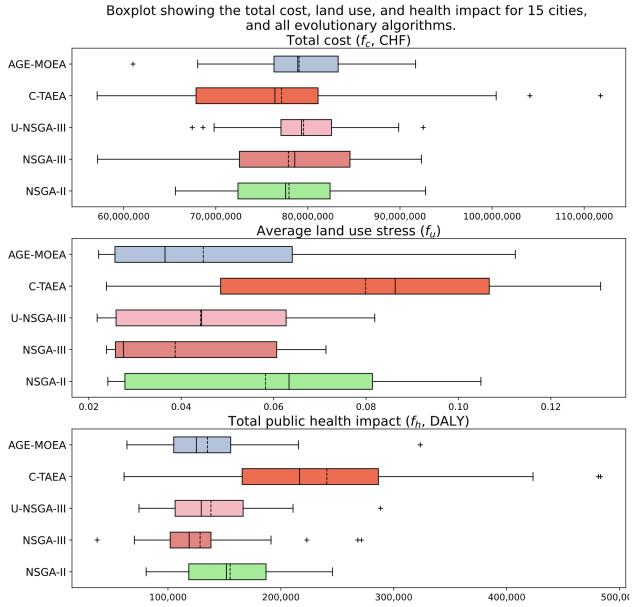
(c) Boxplot for eight cities.



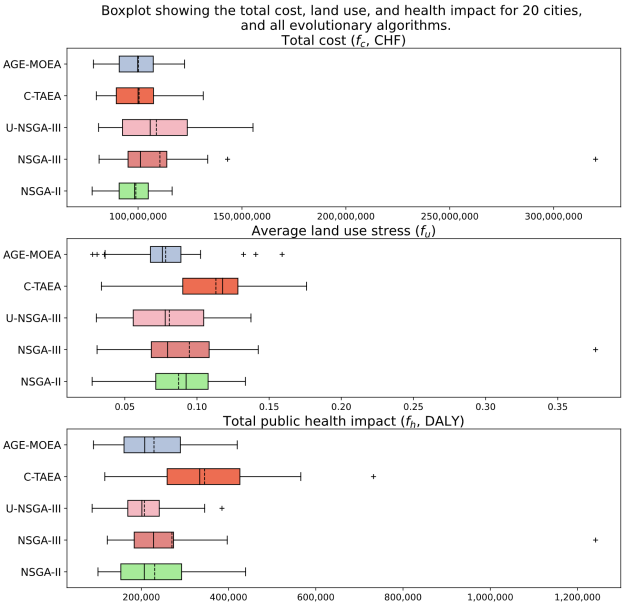
(d) Boxplot for nine cities.



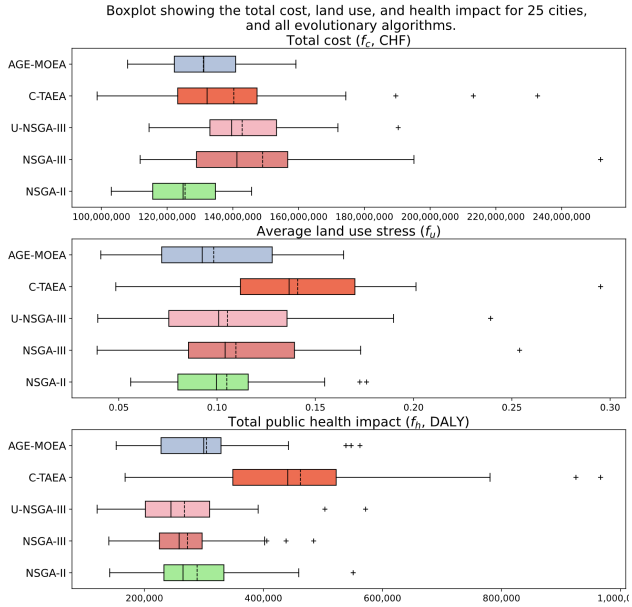
(e) Boxplot for ten cities.



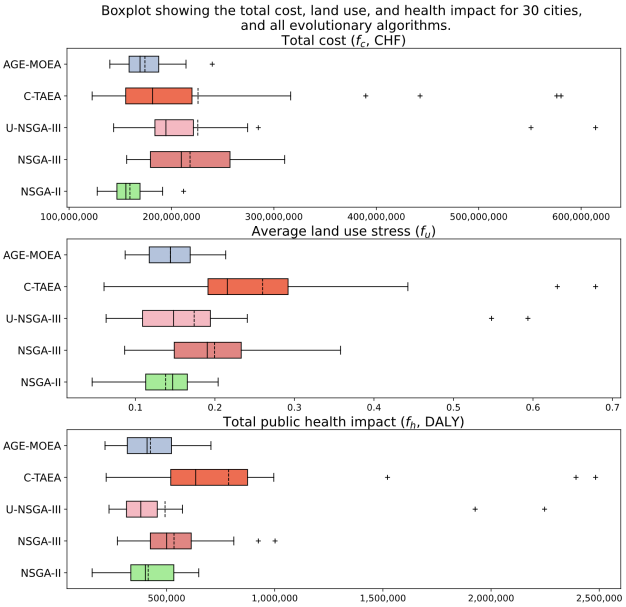
(a) Boxplot for 15 cities.



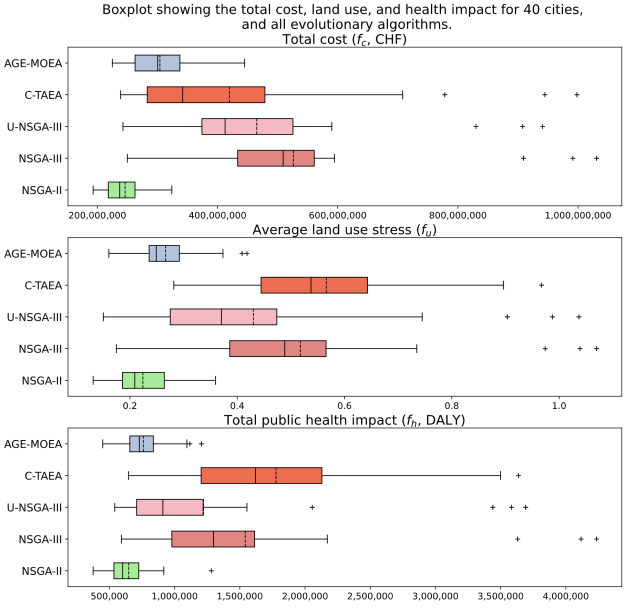
(b) Boxplot for 20 cities.



(c) Boxplot for 25 cities.



(d) Boxplot for 30 cities.



(e) Boxplot for 40 cities.

5.5 Discussion

Starting from the time claim, we conveyed in section 5.1 that the branch and cut algorithm takes much more time as the problem size increases. Due to the nature of the algorithm, it increases exponentially with the number of cities, while the evolutionary algorithms increase polynomially with the number of cities. Thus, we validate the claim of time.

For the subsequent claim, we claimed that our method obtains reasonable solutions close to the branch and cut algorithm. The hypervolume results in section 5.2 convey that most algorithms obtain close solutions based on the hypervolume. In some cities, the evolutionary algorithms even outperform the branch and cut. This outperformance could be due to the spread of solutions. When the evolutionary algorithms outperform the branch and cut algorithm, they find the lower minimum and maximum solutions than the branch and cut algorithm, which finds a smaller amount of spread-out solutions. However, the number of solutions does not necessarily correlate with the quality of the solutions. For example, NSGA-III and U-NSGA-III still find a good spread of solutions and a high hypervolume, even finding the highest hypervolume for nine cities in figure 5.3 with only two solutions found in figure 5.2 and still having a good spread of solutions for the cost and health impact in figure 5.7d.

The final claim included the creation of trade-offs for the three objectives. The boxplots in figures 5.6, 5.7, and 5.8, all show a trade-off for each algorithm with them being better in two of the objectives however finding worse solutions for one objective. The branch and cut approach also shows this feature in some figures such as figure 5.6a or 5.6c, however usually finding a very low trade-off amount across the objectives such as in figures 5.6b or 5.6d. As the problem size gets larger, NSGA-II still finds trade-offs between solutions. C-TAEA finds the most trade-offs for large-scale solutions with an immense spread of solutions in 5.8a, finding low costs and land usage while finding a moderately low health impact. Thus, we have validated that the evolutionary algorithm finds better trade-offs between the three objectives than the branch and cut algorithm.

5.6 Conclusion

For this chapter, we illustrated the observations of the empirical evidence in order to find differences between the proposed evolutionary algorithm method and the branch and cut approach. Furthermore, we also displayed the evolutionary algorithm results finding feasible solutions for large-scale problems. We convey that the claims established throughout this Thesis were validated, including the time taken, quality, and diversity of the solutions.

Chapter 6

Conclusion

In conclusion, we presented an extension to the MILP model proposed by Olapiriyakul et al. [2019] using evolutionary multi-objective algorithms to allow for large-scale graphs and a Pareto front of diverse solutions. In addition, we have validated the claims regarding the time taken, the quality of solutions, and the diversity of the solutions by using statistical inferences. Furthermore, we adapted existing crossover and mutation operators for the evolutionary algorithms to solve the MILP problem with feasible reasonable solutions in polynomial time.

The limitations of this approach are that as the problem size increases to huge problems, the evolutionary algorithm takes longer to find feasible solutions and sometimes converges to infeasible solutions, not finding them at all. However, our method advances research in sustainable waste management for large-scale problems, as it consistently provides better solutions than previous approaches.

Future work for this problem could include a deeper look into the sorting facilities as well as into sorting the waste instead of fully forwarding it to the incinerator or landfill facilities. Additionally, one should consider new algorithms that focus on finding better feasible solutions for large-scale problems faster.

Bibliography

ACR Azienda cantonale dei rifiuti. Rapporto Annuale 2010, 2010. URL https://www.aziendarifiuti.ch/Rapporto-annuale-2010-30fb4e00.

Julian Blank and Kalyanmoy Deb. Pymoo: Multi-Objective Optimization in Python. *IEEE Access*, 8:89497–89509, 2020. ISSN 2169-3536. doi: 10.1109/ACCESS.2020.2990567. URL https://ieeexplore.ieee.org/document/9078759/.

Julian Blank, Kalyanmoy Deb, Yashesh Dhebar, Sunith Bandaru, and Haitham Seada. Generating Well-Spaced Points on a Unit Simplex for Evolutionary Many-Objective Optimization. *IEEE Transactions on Evolutionary Computation*, 25(1):48–60, February 2021. ISSN 1089-778X, 1089-778X, 1941-0026. doi: 10.1109/TEVC.2020.2992387. URL https://ieeexplore.ieee.org/document/9086772/.

Marcus Brandenburg, Kannan Govindan, Joseph Sarkis, and Stefan Seuring. Quantitative models for sustainable supply chain management: Developments and directions. *European Journal of Operational Research*, 233(2):299–312, March 2014. ISSN 0377-2217. doi: 10.1016/j.ejor.2013.09.032. URL https://www.sciencedirect.com/science/article/pii/S037722171300787X.

Thomas Bäck. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford University Press, New York, 1996. ISBN 978-0-19-509971-3.

Carlos A. Coello Coello, Gary B. Lamont, and David A. Van Veldhuisen. Basic Concepts. In *Evolutionary algorithms for solving multi-objective problems*, Genetic and evolutionary computation series, pages 4 – 94. Springer, New York, 2nd ed edition, 2007. ISBN 978-0-387-36797-2.

Indraneel Das and J. E. Dennis. Normal-Boundary Intersection: A New Method for Generating the Pareto Surface in Nonlinear Multicriteria Optimization Problems. *SIAM Journal on Optimization*, 8(3):631–657, August 1998. ISSN 1052-6234, 1095-7189. doi: 10.1137/S1052623496307510. URL http://epubs.siam.org/doi/10.1137/S1052623496307510.

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002. ISSN 1089778X. doi: 10.1109/4235.996017. URL http://ieeexplore.ieee.org/document/996017/.

Kalyanmoy Deb and Himanshu Jain. An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601, August 2014. ISSN 1941-0026. doi: 10.1109/TEVC.2013.2281535. Conference Name: IEEE Transactions on Evolutionary Computation.

Kalyanmoy Deb, Karthik Sindhya, and Tatsuya Okabe. Self-adaptive simulated binary crossover for real-parameter optimization. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation - GECCO '07*, page 1187, London, England, 2007. ACM Press. ISBN 978-1-59593-697-4. doi: 10.1145/1276958.1277190. URL http://portal.acm.org/citation.cfm?doid=1276958.1277190.

Daker Elrabaya and Valentina Marchenko. Identifying the full cost to landfill municipal solid waste by incorporating emissions impact and land development lost opportunity: Case study, Sharjah-UAE. *International Journal of Engineering Sciences*, 10:33, June 2021. doi: 10.35629/6734-1006023341.

Amin Farahbakhsh and Mohammad Ali Forghani. Sustainable location and route planning with GIS for waste sorting centers, case study: Kerman, Iran. *Waste Management & Research: The Journal for a Sustainable Circular Economy*, 37(3):287–300, March 2019. ISSN 0734-242X, 1096-3669. doi: 10.1177/0734242X18815950. URL http://journals.sagepub.com/doi/10.1177/0734242X18815950.

C.M. Fonseca, L. Paquete, and M. Lopez-Ibanez. An Improved Dimension-Sweep Algorithm for the Hypervolume Indicator. In *2006 IEEE International Conference on Evolutionary Computation*, pages 1157–1163, Vancouver, BC, Canada, 2006. IEEE. ISBN 978-0-7803-9487-2. doi: 10.1109/CEC.2006.1688440. URL http://ieeexplore.ieee.org/document/1688440/.

Ralph Gomory. AN ALGORITHM FOR THE MIXED INTEGER PROBLEM. Technical report, RAND CORP SANTA MONICA CA, June 1960. URL https://apps.dtic.mil/sti/citations/AD0616505. Section: Technical Reports.

D.P. Hardin and E.B. Saff. Minimal Riesz energy point configurations for rectifiable d -dimensional manifolds. *Advances in Mathematics*, 193(1):174–204, May 2005. ISSN 00018708. doi: 10.1016/j.aim.2004.05.006. URL https://linkinghub.elsevier.com/retrieve/pii/S0001870804001537.

A. H. Land and A. G. Doig. An Automatic Method of Solving Discrete Programming Problems. *Econometrica*, 28(3):497, July 1960. ISSN 00129682. doi: 10.2307/1910129. URL https://www.jstor.org/stable/1910129?origin=crossref.

Ke Li, Renzhi Chen, Guangtao Fu, and Xin Yao. Two-Archive Evolutionary Algorithm for Constrained Multiobjective Optimization. *IEEE Transactions on Evolutionary Computation*, 23(2):303–315, April 2019. ISSN 1089-778X, 1089-778X, 1941-0026. doi: 10.1109/TEVC.2018.2855411. URL https://ieeexplore.ieee.org/document/8413136/.

C. J. Murray. Quantifying the burden of disease: the technical basis for disability-adjusted life years. *Bulletin of the World Health Organization*, 72(3):429–445, 1994. ISSN 0042-9686.

National-Geographic-Society. Microplastics, May 2022. URL https://education.nationalgeographic.org/resource/microplastics.

OECD. Waste: Municipal waste (Edition 2019). Technical report, OECD, 2019. URL https://www.oecd-ilibrary.org/environment/data/oecd-environment-statistics/waste-municipal-waste-edition-2019_039a0179-en. Type: dataset.

Sun Olapiriyakul, Warut Pannakkong, Warith Kachapanya, and Stefano Starita. Multi-objective Optimization Model for Sustainable Waste Management Network Design. *Journal of Advanced Transportation*, 2019:1–15, May 2019. doi: 10.1155/2019/3612809.

Annibale Panichella. An adaptive evolutionary algorithm based on non-euclidean geometry for many-objective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 595–603, Prague Czech Republic, July 2019. ACM. ISBN 978-1-4503-6111-8. doi: 10.1145/3321707.3321839. URL https://dl.acm.org/doi/10.1145/3321707.3321839.

Haitham Seada and Kalyanmoy Deb. A Unified Evolutionary Optimization Procedure for Single, Multiple, and Many Objectives. *IEEE Transactions on Evolutionary Computation*, 20(3):358–369, June 2016. ISSN 1089-778X, 1089-778X, 1941-0026. doi: 10.1109/TEVC.2015.2459718. URL http://ieeexplore.ieee.org/document/7164289/.

Statista. Global MSW recycling rates by country | Statista, 07 2022. URL https://www.statista.com/statistics/1052439/rate-of-msw-recycling-worldwide-by-key-country/.

A. C. Thompson. *Minkowski Geometry*. Cambridge University Press, 1 edition, June 1996. ISBN 978-0-521-40472-3 978-1-107-32584-5. doi: 10.

1017/CBO9781107325845. URL https://www.cambridge.org/core/product/identifier/9781107325845/type/book.

Ian Tiseo. Global waste generation - statistics & facts | Statista, June 2022. URL https://www.statista.com/topics/4983/waste-generation-worldwide/#dossierKeyfigures.

Valerie Volcovici. U.S. plastic recycling rate drops to close to 5% - report. *Reuters*, May 2022. URL https://www.reuters.com/world/us/us-plastic-recycling-rate-drops-close-5-report-2022-05-04/.

Eckart Zitzler and Simon Künzli. Indicator-Based Selection in Multiobjective Search. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Dough Tygar, Moshe Y. Vardi, Gerhard Weikum, Xin Yao, Edmund K. Burke, José A. Lozano, Jim Smith, Juan Julián Merelo-Guervós, John A. Bullinaria, Jonathan E. Rowe, Peter Tiňo, Ata Kabán, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VIII*, volume 3242, pages 832–842. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. ISBN 978-3-540-23092-2 978-3-540-30217-9. doi: 10.1007/978-3-540-30217-9_84. URL http://link.springer.com/10.1007/978-3-540-30217-9_84. Series Title: Lecture Notes in Computer Science.