Chapter 2

Preliminaries

This chapter provides a gentle introduction to the core subjects of this thesis. It covers the basics of optimisation, multi-objective optimisation, evolutionary computation, and building spatial design.

2.1 Optimisation

In optimisation the aim is to find an optimal, or at least an improved solution, to a given problem. A problem may be that you want to design a house, but how do you decide on the properties (number of rooms, floorspace, type of isolation material) of the house? Let us assume – for the moment – that your only concern is the purchase price, which you want to be as low as possible. That is, your objective is the minimisation of the price of the house. Naturally, how much a house costs depends on the properties of the house. A larger number of rooms, for instance, may result in a higher price. Such properties may be captured in variables. Given an objective, and the variables that influence its value, an objective function can be described. In other words, a function describing how the objective value changes, given different input variables. Altogether the optimisation problem can now be described mathematically as follows. An objective function f, with a vector of input variables $\mathbf{x}$, is to be minimised. With $\mathbf{x}$ being an element of the search space $\mathcal{S}$. Here the search space is the set of all possible variable combinations. In short:

$$f(\mathbf{x}) \to \min, \quad \mathbf{x} \in \mathcal{S}. \tag{2.1}$$

2.1.1 Constraints

Although unconstrained objective functions exist, in practice most optimisation problems consider constraints. Two classes of constraints are distinguished here, namely equality and inequality constraints. For instance, for the design of a house, a specific number of rooms c_1 may be required, leading to an equality constraint of the form:

$$g(\mathbf{x}) = c_1. \tag{2.2}$$

Further, there may be a minimum requirement to the number of square metres c_2 in floorspace, resulting in an inequality constraint of the form:

$$h(\mathbf{x}) \geq c_2. \tag{2.3}$$

Finally, it is noted that a constraint might be applicable to a subset of the input variables. For instance variables $x_1, \ldots, x_5$ may be subject to constraint function g_1, while variables $x_3, \ldots, x_n$ are subject to constraint function g_2.

2.1.2 Mixed-Integer Problems

Various types of variables may be considered in optimisation problems. Objective values will generally have some ordering to them, and no major problems arise whether they are real or integer. However, decision variables may consist of different types. In this work we consider three classes of decision variables: real (also, continuous), integer (also, discrete), and nominal discrete (also, categorical). Examples for each are as follows, the height of a window could be a real variable, the number of rooms on the ground floor is an integer, and the type of material in a wall can be nominal discrete. Note that nominal discrete variables are usually encoded as integers, but by their nominal nature are not ordered. As a result, handling them differently from regular integer variables may be advantageous in an optimisation algorithm. Although both real and integer variables are ordered, they still have differing properties and therefore benefit from being treated separately as well.

2.2 Multi-Objective Optimisation

Equivalent to single-objective optimisation, multi-objective optimisation considers the minimisation (or, without loss of generality, maximisation) of objective functions. Instead of optimising one function at a time, multiple functions are optimised in concert.

That is, rather than considering $f(\mathbf{x}) \to \min$, we consider $f_1(\mathbf{x}) \to \min, \dots, f_m(\mathbf{x}) \to \min$. Here, m indicates the number of objectives.

Beyond the basic concepts introduced in the following, the interested reader is referred to [44] for a general overview and introduction to the field of multi-objective optimisation, and state-of-the-art multi-objective evolutionary algorithms (MOEAs).

2.2.1 Pareto Optimality

In the single-objective case defining optimality is trivial: The smallest (or largest) possible value for a given objective function is the optimum. In the multi-objective case we could simply take the minimal value of each objective function, and say this is the set of optima. However, this poses a problem. Consider a bi-objective problem, with conflicting objectives. That is, optimal variables for one objective are not optimal for the other and vice versa. An optimal solution in one objective is in this case likely to be a low quality solution in the other objective. Clearly, it would be of interest to find solutions that are in between, good – but not necessarily optimal – in both individual objectives.

The Pareto (also: Edgeworth-Pareto) order makes it possible to measure optimality of such in between solutions. This order is a partial order on the solutions. That is, for some solutions it is clear which solution is preferable, but in other cases there is no strict preference for one solution over the other. Any two solutions can be compared using the notions of dominance, incomparability, and indifference. Here $\mathbf{y}$ denotes a vector of m objective values. Also note that minimisation of the objectives is considered in the following.

Definition 2.1 (Dominance)**.** Solution $\mathbf{y}^{(1)}$ is said to dominate solution $\mathbf{y}^{(2)}$ if and only if $\forall_{i\in\{1,\dots,m\}} : y_i^{(1)} \leq y_i^{(2)}$ and $\exists_{i\in\{1,\dots,m\}} : y_i^{(1)} < y_i^{(2)}$, in symbols $\mathbf{y}^{(1)} \prec \mathbf{y}^{(2)}$.

Definition 2.2 (Incomparability)**.** Solution $\mathbf{y}^{(1)}$ is said to be incomparable to solution $\mathbf{y}^{(2)}$ if and only if $\exists_{i\in\{1,\dots,m\}} : y_i^{(1)} < y_i^{(2)}$ and $\exists_{i\in\{1,\dots,m\}} : y_i^{(1)} > y_i^{(2)}$, in symbols $\mathbf{y}^{(1)} \| \mathbf{y}^{(2)}$.

Definition 2.3 (Indifference)**.** Solution $\mathbf{y}^{(1)}$ is said to be indifferent to solution $\mathbf{y}^{(2)}$ if and only if $\forall_{i\in\{1,\dots,m\}} : y_i^{(1)} = y_i^{(2)}$, in symbols $\mathbf{y}^{(1)} \sim \mathbf{y}^{(2)}$.

Observe that indifference is equivalent to equality in the objective space. If the mapping $f : \mathcal{S} \to \mathbb{R}^m$ is considered, it is not equivalent to equality in the search space $\mathcal{S}$:

$$\mathbf{x}^{(1)} \sim \mathbf{x}^{(2)} \;\not\Rightarrow\; \mathbf{x}^{(1)} = \mathbf{x}^{(2)} \tag{2.4}$$

Note that these are all pairwise comparisons. In other words, they compare two solutions to each other, but do not say anything about how an individual solution compares to the set of all other solutions. For instance, identifying whether a solution $\mathbf{y}^{(1)}$ is Pareto optimal with respect to all other solutions $\mathbf{y}^{(i\neq 1)}$ requires (in the worst case) a pairwise comparison to each of them.

2.2.2 The Hypervolume Indicator

Although the Pareto order makes it possible to compare two solutions with each other, it provides no direct way to compare larger sets of (mutually incomparable) solutions. Indicators aggregate information about a set of solutions, or introduce additional preference information (on how to distribute the set across the front), but in doing so are necessarily imperfect measures. Two sets of solutions may, for instance, have an equivalent indicator value, even if they are not equivalent themselves. The hypervolume indicator [107] measures the region ($m = 2$, area; $m = 3$, volume; $m \geq 4$, hypervolume) covered by a set of points $\mathbf{Y}$, relative to a reference point $\boldsymbol{\rho}$. Given a single point this is easily done by measuring the region between this point and the reference point. With more points, things naturally get more complicated, since the different regions between the two points and the reference point may overlap. The hypervolume indicator measures the union of these two regions, rather than their sum. Further, it is defined as a general measure, for any number of dimensions. For a more detailed description of the hypervolume indicator, see [107].

Definition 2.4 (Hypervolume Indicator)**.**

$$H(\mathbf{Y}) = \Lambda_m(\cup_{\mathbf{y}\in\mathbf{Y}}[\mathbf{y}, \boldsymbol{\rho}]) \tag{2.5}$$

here Λ_m denotes the Lebesgue measure on $\mathbb{R}^m$, with m being the number of objective functions.

2.3 Evolutionary Computation

Optimisation – in the single as well as the multi-objective case – concerns the minimisation (or maximisation) of some objective function(s). Well-behaved functions can be optimised efficiently by exact methods, such as gradient-based search. However, many functions are non-differentiable or have complex (e.g. multimodal) landscapes, and are not well suited to traditional optimisation techniques. Heuristic methods, and nature inspired methods in particular, have shown great success in optimising for such complex objective functions.

One class of nature inspired methods draws ideas from Darwinian evolution and genetics, and is termed evolutionary computation. Different branches of evolutionary computation were originally developed by independent communities. This resulted in evolution strategies [81, 88], evolutionary programming [48], and genetic algorithms [53]. Although differences exist between these branches, the core concepts are largely the same. As such, a generalised model of evolutionary computation is considered in the following.

Basically, a parent population of size μ is considered and evaluated on the target objective function. From this population, individuals are selected through mating selection. The selected parent individuals then produce λ offspring individuals through recombination. Offspring individuals are then mutated. In this manner variation is introduced into the offspring population, enabling the discovery of new solutions. These offspring individuals are evaluated, and finally survival selection determines a new parent population (again of size μ) to be used in the next generation, based on the used performance metric.

While the above represents a general outline for an evolutionary algorithm, much variation is possible. Imagine, for instance, an algorithm that only uses mutation. Furthermore, for each of the basic steps multiple variations have been introduced over time. In the case of survival selection the so called plus-strategy, and comma-strategy [89, 90] were considered. A plus-strategy, written $(\mu + \lambda)$, selects the new parent population from both the old parents and the offspring. On the other hand, the comma-strategy, written (μ, λ), selects only from the offspring (which necessitates $\lambda \geq \mu$).

2.4 Building Spatial Design

Applying optimisation techniques in building spatial design requires a basic understanding of the topic, which is introduced in the following.

Building spatial design concerns the design of the internal, and external shape of a building, subdivided into so-called spaces[1]. A space is similar to a room, but also

[1]In some fields spaces are referred to as zones

encapsulates concepts such as corridors and atria. In Figure 2.1 an example of a simple spatial design consisting of three spaces is shown.

The building spatial design is an important part of the full building design since it influences and restricts many other design decisions. In order to minimise the need for

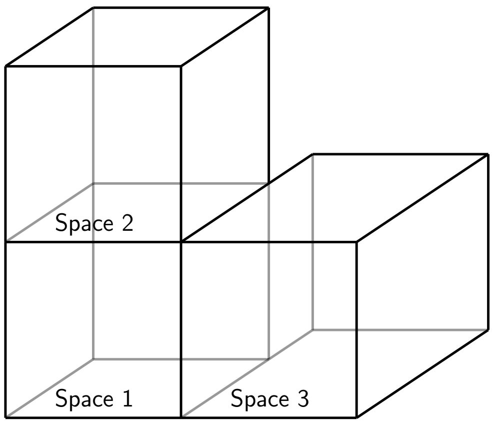

Figure 2.1: Example of a simple spatial design with three spaces.

repeated adjustments in other design components, the spatial design is more or less fixed early on. However, due to its large impact on the rest of the design process, it is of great value to find the best spatial design possible. Optimising spatial designs requires a performance metric. Here the structural performance, and the energy performance are considered.

Structural performance will be measured by the total amount of strain energy of the structural elements of the building. Strain energy is an indicator for how well loads are distributed via the different structural elements. The needed structural elements are, however, not directly available in the spatial design. As such, structural components with given properties are added to the spatial design in order to be able to measure the strain energy. Specific properties of the added components, and the procedures to arrange them, differ between experiments, and are therefore described with the experiments.

Measuring energy performance is carried out in two ways, (a) outside surface area, and (b) heating and cooling simulations. Outside surface area is cheap to compute, and serves as a proxy for energy transfer between the building and the environment in some experiments. Heating and cooling simulations make it possible to measure the required energy to maintain a comfortable temperature. These simulations give a more accurate picture since they are based on properties of the building elements, and they take also into account internal energy transfer between spaces. As with strain energy, the measurement of heating and cooling energy also requires additional properties not available in the spatial design. Likewise, these are described along with the specific experiments. More details are available in [25].