Multicriteria Optimization and Decision Making

Principles, Algorithms and Case Studies

Michael Emmerich and André Deutz
LIACS Master Course: Autumn/Winter 2014/2015

Contents

1 Introduction **5**

1.1 Viewing mulicriteria optimization as a task in system design and analysis . . . 7

1.2 Formal Problem Definitions . . . 9

1.2.1 Other Problem Classes in Optimization . . . 10

1.2.2 Multiobjective Optimization . . . 11

1.3 Problem Difficulty in Optimization . . . 12

1.3.1 Problem Difficulty in Continuous Optimization . . . 12

1.3.2 Problem Difficulty in Combinatorial Optimization . . . 14

1.4 Pareto dominance and incomparability . . . 17

1.5 Formal Definition of Pareto Dominance . . . 19

I Foundations **21**

2 Orders and dominance **22**

2.1 Preorders . . . 22

2.2 Preorders . . . 23

2.3 Partial orders . . . 25

2.4 Linear orders and anti-chains . . . 26

2.5 Hasse diagrams . . . 26

2.6 Comparing ordered sets . . . 28

2.7 Cone orders . . . 29

3 Pareto optima and efficient points **34**

3.1 Search Space vs. Objective Space . . . 34

3.2 Global Pareto Fronts and Efficient Sets . . . 36

3.3 Weak efficiency . . . 37

3.4 Characteristics of Pareto Sets . . . 38

3.5 Optimality conditions based on level sets . . . 38

3.6 Local Pareto Optimality . . . 40

3.7 Barrier Structures 43
3.8 Shapes of Pareto Fronts 46

4 Optimality conditions for differentiable problems 51
4.1 Linear approximations 51
4.2 Unconstrained Optimization 51
4.3 Equality Constraints 53
4.4 Inequality Constraints 57
4.5 Multiple Objectives 59

5 Scalarization Methods 62
5.1 Linear Aggregation 63
5.2 Nonlinear Aggregation 65
5.3 Multi-Attribute Utility Theory 65
5.4 Distance to a Reference Point Methods 70

6 Transforming Multicriteria into Constrained Single-Criterion Problems 74
6.1 Compromise Programming or ϵ-Constraint Methods 74
6.2 Concluding remarks on single point methods 76

II Algorithms for Pareto Optimization 79

7 Pareto Front Computing with Deterministic Methods 80
7.1 Computing the Maximal Set of Vectors 80

8 Evolutionary Multicriterion Optimization 84
8.1 Single Objective Evolutionary Algorithms 85
8.2 Evolutionary Multiobjective Optimization 89
8.2.1 Algorithms Balancing Dominance and Diversity 91
8.2.2 Indicator-based Algorithms 92
8.2.3 Multiobjective Evolution Strategy 95
8.3 Final Remarks 96

9 Conclusion 97

Preface

Finding optimal solutions in large and constrained search spaces has been since long the core topic of operations research and engineering optimization. Such problems are typically challenging from an algorithmic point of view. Multicriteria optimization can be seen as a modern variant of it, that also takes into account that in real world problems we also have to deal with multicriteria decision making and the aspect of searching for an optimum should be combined with aspects of multicriteria decision analysis (MCDA), which is the science of making good choices based on the systematic analysis of alternatives:

Real world decision and optimization problems usually involve conflicting criteria. Think of chosing a means to travel from one country to another. It should be fast, cheap or convenient, but you probably cannot have it all. Or you would like to design an industrial process, that should be safe, environmental friendly and cost efficient. Ideal solutions, where all objectives are at their optimal level, are rather the exception than the rule. Rather we need to find good compromises, and avoid lose-lose situations.

These lecture nodes deal with Multiobjective Optimization and Decision Analysis (MODA). We define this field, based on some other scientific disciplines:

DEF *Multicriteria Decision Aiding (MCDA)* (or: Multiattribute Decision Analysis) is a scientific field that studies evaluation of a finite number of alternatives based on multiple criteria. It provides methods to compare, evaluate, and rank solutions.

DEF *Multicriteria Optimization (MCO)* (or: Multicriteria Design, Multicriteria Mathematical Programming) is a scientific field that studies search for optimal solutions given multiple criteria and constraints. Here, usually, the search space is very large and not all solutions can be inspected.

DEF *Multicriteria Decision Making (MCDM)* deals with MCDA and MCO or combinations of these.

We use here the title: **Multicriteria Optimization and Decision Analysis = MODA** as a synonym of MCDM in order to focus more on the algorithmically challenging optimization aspect.

In this course we will deal with algorithmic methods for solving (constrained) multi-objective optimization and decision making problems. The rich mathematical structure of such problems as well as their high relevance in various application fields led recently to a significant increase of research activities. In particular algorithms that make use of fast, parallel computing technologies are envisaged for tackling hard combinatorial and/or nonlinear application problems. In the course we will discuss the theoretical foundations of multi-objective optimization problems and their solution methods, including order and decision theory, analytical, interactive and meta-heuristic solution methods as well as state-of-the-art tools for their performance-assessment. Also an overview on decision aid tools and formal ways to reason about conflicts will be provided. All theoretical concepts will be accompanied by illustrative hand calculations and graphical visualizations during the course. In the second part of the course, the discussed approaches will be exemplified by the presentation of case studies from the literature, including various application domains of decision making, e.g. economy, engineering, medicine or social science.

This reader is covering the topic of Multicriteria Optimization and Decision Making. Our aim is to give a broad introduction to the field, rather than to specialize on certain types of algorithms and applications. Exact algorithms for solving optimization algorithms are discussed as well as selected techniques from the field of metaheuristic optimization, which received growing popularity in recent years. The lecture notes provides a detailed introduction into the foundations and a starting point into the methods and applications for this exciting field of interdisciplinary science. Besides orienting the reader about state-of-the-art techniques and terminology, references are given that invite the reader to further reading and point to specialized topics.

Chapter 1

Introduction

For several reasons multicriteria optimization and decision making is an exciting field of computer science and operations research. Part of its fascination stems from the fact that in MCO and MCDM different scientific fields are addressed. Firstly, to develop the general foundations and methods of the field one has to deal with structural sciences, such as algorithmics, relational logic, operations research, and numerical analysis:

- How can we state a decision/optimization problem in a formal way?
- What are the essential differences between single objective and multiobjective optimization?
- How can we rank solutions? What different types of orderings are used in decision theory and how are they related to each other?
- Given a decision model or optimization problem, which formal conditions need to be satisfied for solutions to be optimal?
- How can we construct algorithms that obtain optimal solutions, or approximations to them, in an efficient way?
- What is the geometrical structure of solution sets for problems with more than one optimal solution?

Whenever it comes to decision making in the real world, these decisions will be made by people responsible for it. In order to understand how people come to decisions and how the psychology of individuals (cognition, individual decision making) and organizations (group decision making) needs to be studied. Questions like the following may arise:

- What are our goals? What makes it difficult to state goals? How do people define goals? Can the process of identifying goals be supported?
- Which different strategies are used by people to come to decisions? How can satisfaction be measured? What strategies are promising in obtaining satisfactory decisions?
- What are the cognitive aspects in decision making? How can decision support systems be build in a way that takes care of cognitive capabilities and limits of humans?
- How do groups of people come to decisions? What are conflicts and how can they be avoided? How to deal with minority interests in a democratic decision process? Can these aspects be integrated into formal decision models?

Moreover, decisions are always related to some real world problem. Given an application field, we may find very specific answers to the following questions:

- What is the set of alternatives?
- By which means can we retrieve the values for the criteria (experiments, surveys, function evaluations)? Are there any particular problems with these measurements (dangers, costs), and how to deal with them? What are the uncertainties in these measurements?
- What are the problem-specific objectives and constraints?
- What are typical decision processes in the field, and what implications do they have for the design of decision support systems?
- Are there existing problem-specific procedures for decision support and optimization, and what about the acceptance and performance of these procedures in practice?

In summary, this list of questions gives some kind of bird's eye view of the field. However, in these lecture notes we will mainly focus on the structural aspects of multi-objective optimization and decision making. On the other hand, we also devote one chapter to human-centric aspects of decision making and one chapter to the problem of selecting, adapting, and evaluating MOO tools for application problems.

1.1 Viewing mulicriteria optimization as a task in system design and analysis

The discussion above can be seen as a rough sketch of questions that define the scope of multicriteria optimization and decision making. However, it needs to be clarified more precisely what is going to be the focus of these lecture notes. For this reason we want to approach the problem class from the point of view of system design and analysis. Here, with system analysis, we denote the interdisciplinary research field, that deals with the modeling, simulation, and synthesis of complex systems.

Beside experimentation with a physical system, often a system model is used. Nowadays, system models are typically implemented as computer programs that solve (differential) equation systems, simulate interacting automata, or stochastic models. We will also refer to them as *simulation models*. An example for a simulation model based on differential equations would be the simulation of the fluid flow around an airfoil based on the Navier Stokes equations. An example for a stochastic system model, could be the simulation of a system of elevators, based on some agent based stochastic model.

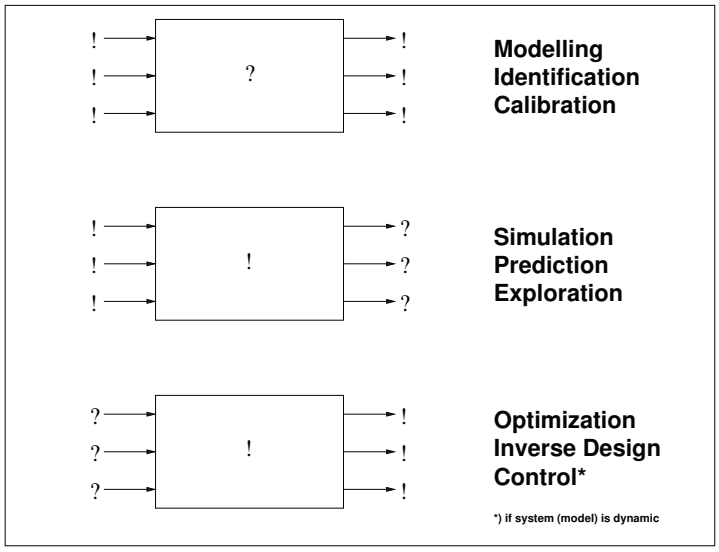

Figure 1.1: Different tasks in systems analysis.

In Figure 1.1 different tasks of systems analysis based on simulation models are displayed in a schematic way. *Modeling* means to identify the internal structure of the simulation model. This is done by looking at the relationship between known

inputs and outputs of the system. In many cases, the internal structure of the system is already known up to a certain granularity and only some parameters need to be identified. In this case we usually speak of *calibration* of the simulation model instead of modeling. In control theory, also the term *identification* is common.

Once a simulation-model of a system is given, we can simulate the system, i.e. predict the state of the output variables for different input vectors. Simulation can be used for predicting the output for not yet measured input vectors. Usually such model-based predictions are much cheaper than to do the experiment in the real world. Consider for example crash test simulations or the simulation of wind channels. In many cases, such as for future predictions, where time is the input variable, it is even impossible to do the experiments in the physical world. Often the purpose of simulation is also to learn more about the behavior of the systems. In this case systematic experimenting is often used to study effects of different input variables and combinations of them. The field of Design and Analysis of Computer Experiments (DACE) is devoted to such systematic explorations of a systems behavior.

Finally, we may want to optimize a system: In that case we basically specify what the output of the system should be. We also are given a simulation-model to do experiments with, or even the physical system itself. The relevant, open question is how to choose the input variables in order to achieve the desired output. In optimization we typically want to maximize (or minimize) the value of an output variable.

On the other hand, a very common situation in practice is the task of adjusting the value of an output variable in a way that it is as close as possible to a desired output value. In that case we speak about inverse design, or if the system is dynamically changing, it may be classified as a optimal control task. An example for an inverse design problem is given in airfoil design, where a specified pressure profile around an airfoil should be achieved for a given flight condition. An example for an optimal control task would be to keep a process temperature of a chemical reactor as close to a specified temperature as possible in a dynamically changing environment.

Note, that the inverse design problem can be reformulated as optimization problem, as it aims at minimizing the deviation between the current state of the output variables and the desired state.

In multi-objective optimization we look at the optimization of systems w.r.t. more than one output variables. Single-objective optimization can be considered as a special case of multi-objective optimization with only one output variable.

Moreover, classically, multi-objective optimization problems are most of the time reduced to single-objective optimization problems. We refer to these reduction techniques as *scalarization* techniques. A chapter in these lecture notes is

devoted to this topic. Modern techniques, however, often aim at obtaining a set of 'interesting' solutions by means of so-called Pareto optimization techniques. What is meant by this will be discussed in the remainder of this chapter.

1.2 Formal Problem Definitions in Mathematical Programming

Researchers in the field of *operations research* use an elegant and standardized notation for the classification and formalization of optimization and decision problems, the so-called *mathematical programming problems*, among which linear programs (LP) are certainly the most prominent representant. Using this notion a *generic definition of optimization problems* is as follows:

$$
\begin{aligned}
f(\mathbf{x}) &\rightarrow \text{min!} && \text{(* Objectives *)} && (1.1)\\
g_1(\mathbf{x}) &\leq 0 && \text{(* Inequality constraints *)} && (1.2)\\
&\vdots && && (1.3)\\
g_{n_g}(\mathbf{x}) &\leq 0 && && (1.4)\\
h_1(\mathbf{x}) &= 0 && \text{(* Equality Constraints *)} && (1.5)\\
&\vdots && && (1.6)\\
h_{n_h}(\mathbf{x}) &= 0 && && (1.7)\\
\mathbf{x} &\in \mathcal{X} = [\mathbf{x}^{\min}, \mathbf{x}^{\max}] \subset \mathbb{R}^{n_x} \times \mathbb{Z}^{n_z} && \text{(* Box constraints *)} && (1.8)\\
& && && (1.9)
\end{aligned}
$$

The objective function f is a function to be minimized (or maximized[1]). This is the goal of the optimization. The function f can be evaluated for every point $\mathbf{x}$ in the search space (or decision space). Here the *search space* is defined by a set of intervals, that restrict the range of variables, so called bounds or box constraints. Besides this, variables can be integer variables that is they are chosen from $\mathbb{Z}$ or subsets of it, or continuous variable (from $\mathbb{R}$). An important special case of integer variables are binary variables which are often used to model binary decisions in mathematical programming.

Whenever inequality and equality constraints are stated explicitly, the search space $\mathcal{X}$ can be partitioned in a *feasible search space* $\mathcal{X}_f \subseteq \mathcal{X}$ and an *infeasible subspace* $\mathcal{X} - \mathcal{X}_f$. In the feasible subspace all conditions stated in the mathematical programming problem are satisfied. The conditions in the mathematical

[1]Maximization can be rewritten as minimization by changing the sign of the objective function, that is, replacing $f(\mathbf{x}) \rightarrow \max$ with $-f(\mathbf{x}) \rightarrow \min$

program are used to avoid constraint violations in the system under design, e.g., the excess of a critical temperature or pressure in a chemical reactor (an example for an inequality constraint), or the keeping of conservation of mass formulated as an equation (an example for an equality constraint). The conditions are called constraints. Due to a convention in the field of operations research, constraints are typically written in a *standardized form* such that 0 appears on the right hand side side. Equations can easily be transformed into the standard form by means of algebraic equivalence transformations.

Based on this very general problem definition we can define several classes of optimization problems by looking at the characteristics of the functions f, $g_i, i = 1, \ldots, n_g$, and $h_i, i = 1, \ldots, n_h$. Some important classes are listed in Table 1.1.

Name	Abbreviation	Search Space	Functions
Linear Program	LP	$\mathbb{R}^{n_r}$	linear
Quadratic Program	QP	$\mathbb{R}^{n_r}$	quadratic
Integer Linear Progam	ILP	$\mathbb{Z}^{n_z}$	linear
Integer Progam	IP	$\mathbb{Z}^{n_z}$	arbitrary
Mixed Integer Linear Program	MILP	$\mathbb{Z}^{n_z} \times \mathbb{R}^{n_r}$	linear
Mixed Integer Nonlinear Program	MINLP	$\mathbb{Z}^{n_z} \times \mathbb{R}^{n_r}$	nonlinear

Table 1.1: Classification of mathematical programming problems.

1.2.1 Other Problem Classes in Optimization

There are also other types of mathematical programming problems. These are, for instance based on:

- The handling of uncertainties and noise of the input variables and of the parameters of the objective function: Such problems fall into the class of *robust optimization problems* and *stochastic programming*. If some of the constants in the objective functions are modeled as stochastic variables the corresponding problems are also called a *parametric optimization problem*.

- Non-standard search spaces: Non-standard search spaces are for instance the search spaces of trees (e.g. representing regular expressions), network configurations (e.g. representing flowsheet designs) or searching for 3-D structures (e.g. representing bridge constructions). Such problem are referred to as *topological*, *grammatical*, or *structure optimization*.

- A finer distinction between different mathematical programming problems based on the characteristics of the functions: Often subclasses of mathematical programming problems have certain mathematical properties that

can be exploited for faster solving them. For instance *convex quadratic programming problems* form a special class of relatively easy to solve quadratic programming problems. Moreover, *geometrical programming problems* are an important subclass of nonlinear programming tasks with polynomials that are allowed to h negative numbers or fractions as exponents.

In some cases, the demand that a solution consists of a vector of variables is too restrictive and instead we can define the search space as some set $\mathcal{X}$. In order to capture also these kind of problems a more general definition of a general optimization problem can be used:

$$f_1(\mathbf{x}) \to \min, \quad \mathbf{x} \in \mathcal{X} \tag{1.10}$$

$\mathbf{x} \in \mathcal{X}$ is called the *search point* or *solution candidate* and $\mathcal{X}$ is the search space or decision space. Finally, $f : \mathcal{X} \to \mathbb{R}$ denotes the objective function. Only in cases where $\mathcal{X}$ is a vector space, we may talk of a *decision vector*.

Another important special case is given, if $\mathcal{X} = \mathbb{R}^n$. Such problems are defined as *continuous unconstrained optimization problems* or, simply, unconstrained optimization problems.

For notational convenience in the following we will refer mostly to the generic definition of an optimization problem given in equation 1.10, whenever constraint treatment is not particularly addressed. In such cases we assume that $\mathcal{X}$ already contains only feasible solutions.

1.2.2 Multiobjective Optimization

All optimization problems and mathematical programming problem classes can be generalized to multiobjective optimization problem classes by stating multiple objective functions:

$$f_1(\mathbf{x}) \to \min, \ldots, f_m(\mathbf{x}) \to \min, \quad \mathbf{x} \in \mathcal{X} \tag{1.11}$$

At this point in time it is not clear, how to deal with situations with conflicting objectives, e.g. when it is not possible to minimize all objective functions simultaneously. Note that the problem definition does not yet prescribe how to compare different solutions. To discuss this we will introduce concepts from the theory of ordered sets, such as the Pareto dominance relation. A major part of these lecture notes will then be devoted to the treatise of multiobjective optimization.

Before proceeding in this direction, it is however important to note, that many difficulties of solving single objective optimization problems are inherited by the more general class of multiobjective optimization problems. We will therefore first summarize these.

1.3 Problem Difficulty in Optimization

The way problem difficulty is defined in continuous unconstrained optimization differs widely from the concepts typically referred to in discrete optimization. This is why we look at these two classes separately. Thereafter we will show that discrete optimization problems can be formulated as constrained continuous optimization problems, or, referring to the classification scheme in Table 1.1, as nonlinear programming problems.

1.3.1 Problem Difficulty in Continuous Optimization

In continuous optimization the metaphor of a optimization landscape is often used in order to define problem difficulty. As opposed to just talking about a function, when using the term (search) *landscapes* one explicitly requires the search space to be equipped with a neighborhood structure, which could be a metric or a topology. This topology is typically the standard topology on $\mathbb{R}^n$ and as a metric typically the Euclidean metric is used.

As we will discuss in more rigor in Chapter 4, this gives rise to definitions such as *local optima*, which are points that cannot be improved by replacing them by neighboring points. For many optimum seeking algorithms it is difficult to escape from such points or find a good direction (in case of plateaus). If local optima are not also global optima the algorithm might return a suboptimal solutions.

Problems with multiple local optima are called *multimodal optimization problems*, whereas a problem that has only a single local optimum is called a *unimodal optimization problem*. Multimodal optimization problems are, in general, considered to be more difficult to solve than unimodal optimization problems. However, in some cases unimodal optimization problems can be very difficult, too. For instance, in case of large neighbourhoods it can be hard to find the neighbour that improves a point.

Exampes of continuous optimization problems are given in Figure . The problem TP2 is difficult due to discontinuities. The function TP3 has only one local optimum (unimodal) and no discontinuities; therefore it can be expected that local optimization can easily solve this problem. Highly multimodal problems are given in TP5 and TP6.

Another difficulty is imposed by constraints. In constrained optimization problems optima can be located at the boundary of the search space and they can give rise to disconnected feasible subspaces. Again, connectedness is a property that requires the definition of neighbourhoods. The definition of a continuous path can be based on this, which again is used to define connectivity. The reason why disconnected subspaces make problems hard to solve are, similar to the multimodal case, that in these problems barriers are introduced might prevent optimum seeking

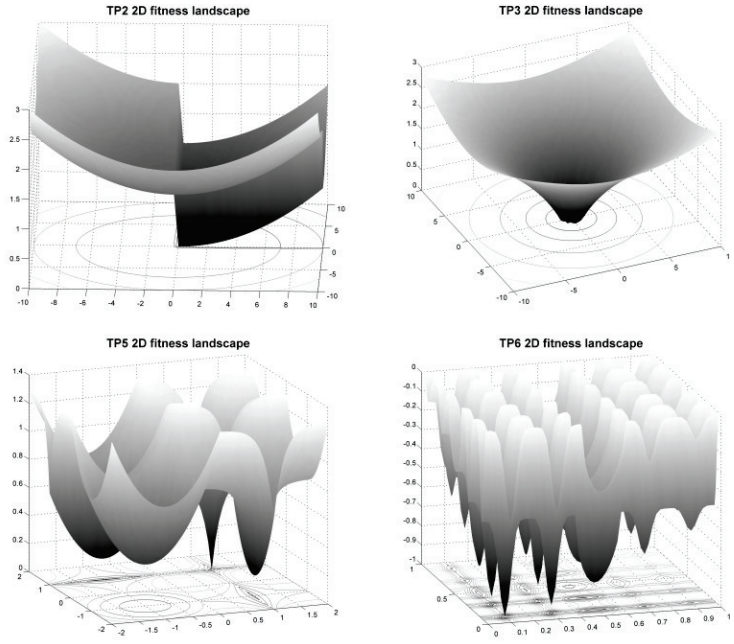

Figure 1.2: Examples of continuous optimization problems.

algorithms that use strategies of gradual improvement to find the global optimum.

Finally, discontinuities and ruggedness of the landscape make problems difficult to solve for many solvers. Discontinuities are abrupt changes of the function value in some neighborhood. In particular these cause difficulties for optimization methods that assume the objective function to be continuous, that is they assume that similar inputs cause similar outputs. A common definition of continuity is that of Lipschitz continuity:

Definition 1 *Let $d(x, y)$ denote the Euclidean distance between two points in the search space. Then function f is Lipschitz continuous, if and only if*

$$|f(x) - f(y)| < kd(x, y) \text{ for some } k > 0.$$

For instance the work by Ritter and Novak [66] has clarified that Lipschitz continuity alone is not sufficient to guarantee that a problem is easy to solve. However, continuity can be exploited for guaranteeing that a region has been sufficiently explored and therefore a small Lipschitz constant has a dampening effect on the worst case time complexity for continuous global optimization, which, even given Lipschitz continuity, grows exponentially with the number of variables involved [66]. In cases where we omit continuity assumptions the time complexity might

even grow super-exponentially. Here complexity is defined as the number of function evaluations it takes to get a solution that has a distance of ϵ to the global optimum and it is assumed that the variables are restricted in an closed interval range.

As indicated above, the number of optimization variables is another source of difficulty in continuous optimization problems. In particular, if f is a black box function it is known that even for Lipschitz continuous problems the number of required function evaluations for finding a good approximation to the global optimum grows exponentially with the number of decision variables. This result is also referred to as the *curse of dimensionality*.

Again, a word of caution is in order: The fact that a problem is low dimensional or even one dimensional in isolation does not say something about its complexity. *Kolmogorov's superposition theorem* shows that every continuous multivariate function can be represented by a one dimensional function, and it is therefore often possible to re-write optimization problems with multiple variables as one-dimensional optimization problems.

Besides continuity assumptions, also differentiability of the objective function and constraints, convexity and mild forms of non-linearity (as given in convex quadratic optimization), as well as limited interaction between variables can make a continuous problem easier to solve. The degree of interaction between variables is given by the number of variables in the term of the objectve function: Assume it is possible to (re)-write the optimization problem in the form $\sum_{i=1}^{n} f_i(x_{i_1}, \ldots, x_{i_{k(i)}}) \rightarrow \max$, then the value of $k(i)$ is the degree of interaction in the i-th component of the objective function. In case of continuous objective functions it can be shown that problems with a low degree of interaction can be solved more efficiently in terms of worst case time complexity[66]. One of the reasons why convex quadratic problems can be solved efficiently is that, given the Hessian matrix, the coordinate system can be transformed by simple rotation in such a way that these problems become decomposable, i.e. $k(i)$ is bounded by 1.

1.3.2 Problem Difficulty in Combinatorial Optimization

Many optimization problem in practice, such as scheduling problems, subset selection problems, and routing problems, belong to the class of *combinatorial optimization problems* and, as the name suggests, they look in some sense for the best combination of parts in a solution (e.g. selected elements of a set, travelled edges in a road network, switch positions in a boolean network). Combinatorial optimization problems are problems formulated on (large) finite search spaces. In the classification scheme in Table 1.1 they belong to the classses IP and ILP. Although combinatorial optimization problems are originally not always formulated on search spaces with integer decision variables, most combinatorial optimization

problems can be transformed to equivalent IP and ILP formulations with binary decision variables. For the sake of brevity, the following discussion will focus on binary unconstrained problems. Most constrained optimization problems can be transformed to equivalent unconstrained optimization problems by simply assigning a sufficiently large ('bad') objective function value to all infeasible solutions.

A common characteristic of many combinatorial optimization problems is that they have a concise (closed form) formulation of the objective function and the objective function (and the constraint functions) can be computed efficiently.

Having said this, a combinatorial optimization problem can be defined by means of a pseudo-boolean objective function, i.e. $f : \{0,1\}^n \rightarrow \mathbb{R}$ and stating the goal $f(\mathbf{x}) \rightarrow \min$. Theoretical computer science has developed a rich theory on the complexity of decision problems. A *decision problem* is the problem of answering a query on input of size n with the answer being either `yes` or `no`. In order to relate the difficulty of optimization problems to the difficulty of decision problems it is beneficial to formulate so-called decision versions of optimization problems.

Definition 2 *Given an combinatorial optimization problem of the form* $f(\mathbf{x}) \rightarrow \max$ *for* $\mathbf{x} \in \{0,1\}^n$ *its* decision version *is defined as the query:*

$$\exists \mathbf{x} \in \{0,1\}^n : f(\mathbf{x}) \leq k \tag{1.12}$$

for a given value of $k \in \mathbb{R}$*.*

NP hard combinatorial optimization problems

A decision problem is said to belong to the class P if there exists an algorithm on a Turing machine[2] that solves it with a time complexity that grows at most polynomially with the size n of the input. It belongs to the class NP if a candidate solution $\mathbf{x}$ of size n can be verified ('checked') with polynomial time complexity (does it satify the formula $f(\mathbf{x}) \leq k$ or not). Obviously, the class NP subsumes the class P, but the P perhaps not neccessarily subsumes NP. In fact, the question whether P subsumes NP is the often discussed open problem in theoretical computer science known as the '$P = NP$' problem. Under the assumption '$P \neq NP$', that is that P does include NP, it is meaningful to introduce the complexity class of NP complete problems:

Definition 3 *A decision problem* D *is* NP complete*, if and only if for all problems* D' *in* NP *there exists an algorithm with polynomial time complexity that*

[2]or any in any common programming language operating on infinite memory and not using parallel processing and not assuming constant time complexity for certain infinite precision floating point operations such as the floor function.

reformulates an instance of D as an instance of D' (One can also say: 'there exists a polynomial-time reduction of D to P').

If any NP complete problem could be solved with polynomial time complexity then all problems in NP have polynomial time complexity.

Many decision versions of optimization problems are NP complete. Closely related to the class of NP complete problems is the class of NP hard problems.

Definition 4 (NP hard) *A computational problem is NP hard, if and only if any problem in the class of NP complete problems can in polynomial time be reduced to this problem.*

That a problem is NP hard does not imply that it is in NP. Moreover, given any NP hard problem could be solved in polynomial time, then all problems in NP could be solved in polynomial time, but not vice versa.

Many combinatorial optimization problems fall into the class of NP hard problems and their decision versions belong to the class of NP complete problems. Examples of NP hard optimization problems are the knapsack problem, the traveling salesperson problem, and integer linear programming (ILP).

At this point in time, despite considerable efforts of researchers, no polynomial time algorithms are known for NP complete problems and thus also not for NP hard problems. As a consequence, relying on currently known algorithms the computational effort to solve NP complete (NP hard) problems grows (at least) exponentially with the size n of the input.

The fact that a given problem instance belongs to a class of NP complete problems does not mean that this instance itself is hard to solve. Firstly, the exponential growth is a statement about the *worst case* time complexity and thus gives an upper bound for the time complexity that holds for all instances of the class. It might well be the case that for a given instance the worst case is not binding. Often certain structural features such as *bounded tree width* reveal that an instance belongs to an easier to solve subclass of a NP complete problem. Moreover, exponential growth might occur with a small growth rate and problem sizes relevant in practice might still be solvable in an acceptable time.

Continuous vs. discrete optimization

Given that some continuous versions of mathematical programming problems belong to easier to solve problem classes than their discrete counterparts one might ask the question whether integer problems are essentially more difficult to solve than continuous problems.

As a matter of fact, optimization problems on binary input spaces can be

reformulated as quadratic optimization problems by means of the following construction:

Given an integer programming problem with binary decision variables $b_i \in \{0,1\}$, $i = 1,\ldots,n$, we can reformulate this problem as a quadratic programming problem with continuous decision variables $x_i \in \mathbb{R}$ by introducing the constraints $(x_i)(1-x_i) = 1$ for $i = 1,\ldots,n$.

For obvious reasons continuous optimization problems cannot always be formulated as discrete optimization problems. However, it is sometimes argued that all problems solved on digital computers are discrete problems and infinite accuracy is almost never required in practise. If, however, infinite accuracy operations are assumed to have constant time in the construction of an algorithm this can lead to strange consequences. For instance, it is known that polynomial time algorithms can be constructed for NP complete problems, if one would accept that the evaluation of the floor function with infinite precision can be achieved in polynomial time. However, it is not possible to implement these algorithms on a von Neumann architecture with finite precision arithmetics.

Finally, in times of growing amounts of decision data, one should not forget that even guarantees of polynomial time complexity can be insufficient in practise. Accordingly, there is a growing interest for problem solvers that require only subquadratic running time. Similar to the construction of the class of NP complete problems, a definition of the class of $3SUM$ complete problems has been constructed by theoretical computer scientists. For this class up to date only quadratic running time algorithms are known. A prominent problem from the domain of mathematical programming that belongs to this group is the *linear satisfiability problem*, i.e. the problem of whether a set of r linear inequality constraints formulated on n continuous variables can be satisfied [33].

1.4 Pareto dominance

A fundamental problem in multicriteria optimization and decision making is to compare solutions w.r.t. different, possibly conflicting, goals. Before we lay out the theory of orders in a more rigorous manner, we will introduce some fundamental concepts by means of a simple example.

Consider the following decision problem: We have to select one car from the following set of cars: For the moment, let us assume, that our goal is to minimize the price and maximize speed and we do not care about other components.

In that case we can clearly say that the BMW outperforms the Lincoln stretch limousine, which is at the same time more expensive and slower then the BMW. In such a situation we can decide clearly for the BMW. We say that the first solution *(Pareto) dominates* the second solution. Note, that the concept of Pareto

Criterion	Price [kEuro]	Maximum Speed [km/h]	length [m]	color
VW Beetle	3	120	3.5	red
Ferrari	100	232	5	red
BMW	50	210	3.5	silver
Lincoln	60	130	8	white

domination is named after Vilfredo Pareto, an italian economist and engineer who lived from 1848-1923 and who introduced this concept for multi-objective comparisons.

Consider now the case, that you have to compare the BMW to the VW Beetle. In this case it is not clear how to make a decision, as the beetle outperforms the BMW in the cost objective, while the BMW outperforms the VW Beetle in the speed objective. We say that the two solutions are *incomparable*. Incomparability is a very common characteristic that occurs in so-called *partial ordered* sets.

We can also observe, that the BMW is incomparable to the Ferrari, and the Ferrari is incomparable to the VW Beetle. We say these three cars form a set of mutually incomparable solutions. Moreover, we may state that the Ferrari is incomparable to the Lincoln, and the VW Beetle is incomparable to the Lincoln. Accordingly, also the VW Beetle, the Lincoln and the Ferrari form a mutually incomparable set.

Another characteristic of a solution in a set can be that it is *non-dominated* or Pareto optimal. This means that there is no other solution in the set which dominates it. The set of all non-dominated solutions is called the *Pareto front*. It might exist of only one solution (in case of non-conflicting objectives) or it can even include no solution at all (this holds only for some infinite sets). Moreover, the Pareto set is always a mutually incomparable set. In the example this set is given by the VW Beetle, the Ferrari, and the BMW.

An important task in multi-objective optimization is to identify the Pareto front. Usually, if the number of objective is small and there are many alternatives, this reduces the set of alternatives already significantly. However, once the Pareto front has been obtained, a final decision has to be made. This decision is usually made by interactive procedures where the decision maker assesses trade-offs and sharpens constraints on the range of the objectives. In the subsequent chapters we will discuss these procedures in more detail.

Turning back to the example, we will now play a little with the definitions and thereby get a first impression about the rich structure of partially ordered sets in Pareto optimization: What happens if we add a further objective to the set of objectives in the car-example? For example let us assume, we also would like to have a very big car and the size of the car is measured by its length! It is easy to verify that the size of the non-dominated set increases, as now the Lincoln is

also incomparable to all other cars and thus belongs to the non-dominated set. Later we will prove that introducing new objectives will always increase the size of the Pareto front. On the other hand we may define a constraint that we do not want a silver car. In this case the Lincoln enters the Pareto front, since the only solution that dominates it leaves the set of feasible alternatives. In general, the introduction of constraints may increase or decrease Pareto optimal solutions or its size remains the same.

1.5 Formal Definition of Pareto Dominance

A formal and precise definition of Pareto dominance is given as follows. We define a partial order[3] on the *solution space* $\mathcal{Y} = f(\mathcal{X})$ by means of the Pareto dominance concept for vectors in $\mathbb{R}^m$:

For any $\mathbf{y}^{(1)} \in \mathbb{R}^m$ and $\mathbf{y}^{(2)} \in \mathbb{R}^m$: $\mathbf{y^{(1)}}$ dominates $\mathbf{y^{(2)}}$ (in symbols $\mathbf{y^{(1)}} \prec_{Pareto} \mathbf{y^{(2)}}$) if and only if: $\forall i = 1, \ldots m : y_i^{(1)} \leq y_i^{(2)}$ and $\exists i \in \{1, \ldots, m\} : y_i^{(1)} < y_i^{(2)}$.

Note, that in the bi-criteria case this definition reduces to: $\mathbf{y}^1 \prec_{Pareto} \mathbf{y}^2 :\Leftrightarrow y_1^{(1)} < y_1^{(2)} \wedge y_2^{(1)} \leq y_2^{(2)} \vee y_1^{(1)} \leq y_1^{(2)} \wedge y_2^{(1)} < y_2^{(2)}$.

In addition to the domination $\prec_{Pareto}$ we define further comparison operators: $\mathbf{y}^{(1)} \preceq_{Pareto} \mathbf{y}^{(2)} :\Leftrightarrow \mathbf{y}^{(1)} \prec_{Pareto} \mathbf{y}^{(2)} \vee \mathbf{y}^{(1)} = \mathbf{y}^{(2)}$.

Moreover, we say $\mathbf{y}^{(1)}$ is incomparable to $\mathbf{y}^{(2)}$ (in symbols: $\mathbf{y}^{(1)} || \mathbf{y}^{(2)}$),if and only if $\mathbf{y}^{(1)} \not\preceq_{Pareto} \mathbf{y}^{(2)} \wedge \mathbf{y}^{(1)} \not\preceq_{Pareto} \mathbf{y}^{(2)}$.

For technical reasons, we also define *strict* domination as: $\mathbf{y}^{(1)}$ strictly dominates $\mathbf{y}^{(2)}$, iff $\forall i = 1, \ldots, m : y_i^{(1)} < y_i^{(2)}$.

For any compact subset of R^m, say $\mathcal{Y}$, there exists a non-empty set of minimal elements w.r.t. the partial order $\preceq$ (cf. [Ehr05, page 29]). Minimal elements of this partial order are called non-dominated points. Formally, we can define a non-dominated set via: $\mathcal{Y}_N = \{\mathbf{y} \in \mathcal{Y} | \nexists \mathbf{y}' \in \mathbf{Y} : \mathbf{y}' \prec_{\mathbf{Pareto}} \mathbf{y}\}$. Following a convention by Ehrgott [Ehr05] we use the index N to distinguish between the original set and its non-dominated subset.

Having defined the non-dominated set and the concept of Pareto domination for general sets of vectors in $\mathbb{R}^m$, we can now relate it to the optimization task: The aim of Pareto optimization is to find the non-dominated set $\mathcal{Y}_N$ for $\mathcal{Y} = f(\mathcal{X})$ the image of $\mathcal{X}$ under f, the so-called *Pareto front* of the multi-objective optimization problem.

We define $\mathcal{X}_E$ as the inverse image of $\mathcal{Y}_N$, i. e. $\mathcal{X}_E = f^{-1}(\mathcal{Y}_N)$. This set will be called the *efficient set* of the optimization problem. Its members are called *efficient solutions*.

[3]Partial orders will be defined in detail in Chapter 2. For now, we can assume that it is an order where not all elements can be compared.

For notational convenience, we will also introduce an order (which we call prePareto) on the decision space via $\mathbf{x}^{(1)} \prec_{preParetо} \mathbf{x}^{(2)} \Leftrightarrow f(\mathbf{x}^{(1)}) \prec_{Pareto} f(\mathbf{x}^{(2)})$. Accordingly, we define $\mathbf{x}^{(1)} \preceq_{preParetо} \mathbf{x}^{(2)} \Leftrightarrow f(\mathbf{x}^{(1)}) \preceq_{Pareto} f(\mathbf{x}^{2})$. Note, the minimal elements of this order are the efficient solutions, and the set of all minimal elements is equal to $\mathcal{X}_E$.

Exercises

1. How does the introduction of a new solution influence the size of the Pareto set? What happens if solutions are deleted? Prove your results!

2. Why are objective functions and constraint functions essentially different? Give examples of typical constraints and typical objectives in real world problems!

3. Find examples for decision problems with multiple, conflicting objectives! How is the search space defined? What are the constraints, what are the objectives? How do these problems classify, w.r.t. the classification scheme of mathematical programming? What are the human-centric aspects of these problems?

Part I

Foundations

Chapter 2

Orders and dominance

The theory of ordered sets is an essential analytical tool in multi-objective optimization and decision analysis. Different types of orders can be defined by means of axioms on binary relations, and, if we restrict ourselves to vector spaces, also geometrically.

Next we will first show how different types of orders are defined as binary relations that satisfy a certain axioms[1]. Moreover, we will highlight the essential differences between common families of ordered sets, such as preorders, partial orders, linear orders, and cone orders.

The structure of this chapter is as follows: After reviewing the basic concept of binary relations, we define some axiomatic properties of pre-ordered sets, a very general type of ordered sets. Then we define partial orders and linear orders as special type of pre-orders. The difference between linear orders and partial orders sheds a new light on the concept of incomparability and the difference between multicriteria and single criterion optimization. Later, we discuss techniques how to visualize finite ordered sets in a compact way, by so called Hasse diagrams. The remainder of this chapter deals with an alternative way of defining orders on vector spaces: Here we define orders by means of cones. This definition leads also to an intuitive way of how to visualize orders based on the concept of Pareto domination.

2.1 Preorders

Orders can be introduced and compared in an elegant manner as binary relations that obey certain axioms. Let us first review the definition of a binary relation

[1]Using here the term 'axiom' to refer to an elementary statement that is used to define a class of objects (as promoted by, for instance, Rudolf Carnap[16]) rather than viewing them as self-evident laws that do not require proof (Euclid's classical view).

and some common axioms that can be introduced to specify special subclasses of binary relations and that are relevant in the context of ordered sets.

Definition 5 *A* binary relation $\mathcal{R}$ *on some set* $\mathcal{S}$ *is defined as a set of pairs of elements of* $\mathcal{S}$*, that is, a subset of* $\mathcal{S} \times \mathcal{S} = \{(\mathbf{x}^1, \mathbf{x}^2) | \, \mathbf{x}^1 \in \mathcal{S} \text{ and } \mathbf{x}^2 \in \mathcal{S}\}$*. We write* $\mathbf{x}^1 \mathcal{R} \mathbf{x}^2 \Leftrightarrow (\mathbf{x}^1, \mathbf{x}^2) \in \mathcal{R}$*.*

Definition 6 *Properties of binary relations*

$\mathcal{R}$ *is* reflexive $\Leftrightarrow \forall \mathbf{x} \in \mathcal{S} : \mathbf{x} \mathcal{R} \mathbf{x}$

$\mathcal{R}$ *is* irreflexive $\Leftrightarrow \forall \mathbf{x} \in \mathcal{S} : \neg \mathbf{x} \mathcal{R} \mathbf{x}$

$\mathcal{R}$ *is* symmetric $\Leftrightarrow \forall \mathbf{x}^1, \mathbf{x}^2 \in \mathcal{S} : \mathbf{x}^1 \mathcal{R} \mathbf{x}^2 \Leftrightarrow \mathbf{x}^2 \mathcal{R} \mathbf{x}^1$

$\mathcal{R}$ *is* antisymmetric $\Leftrightarrow \forall \mathbf{x}^1, \mathbf{x}^2 \in \mathcal{S} : \mathbf{x}^1 \mathcal{R} \mathbf{x}^2 \wedge \mathbf{x}^2 \mathcal{R} \mathbf{x}^1 \Rightarrow \mathbf{x}^1 = \mathbf{x}^2$

$\mathcal{R}$ *is* asymmetric $\Leftrightarrow \forall \mathbf{x}^1, \mathbf{x}^2 \in \mathcal{S} : \mathbf{x}^1 \mathcal{R} \mathbf{x}^2 \Rightarrow \neg (\mathbf{x}^2 \mathcal{R} \mathbf{x}^1)$

$\mathcal{R}$ *is* transitive $\Leftrightarrow \forall \mathbf{x}^1, \mathbf{x}^2, \mathbf{x}^3 \in \mathcal{S} : \mathbf{x}^1 \mathcal{R} \mathbf{x}^2 \wedge \mathbf{x}^2 \mathcal{R} \mathbf{x}^3 \Rightarrow \mathbf{x}^1 \mathcal{R} \mathbf{x}^3$

Example It is worthwhile to practise these definitions by finding examples for structures that satisfy the aforementioned axioms. An example for a reflexive relation is the equality relation on $\mathbb{R}$, but also the relation $\leq$ on $\mathbb{R}$. A classical example for a irreflexive binary relation would be marriage between two persons. This relation is also symmetric. Symmetry is also typically a characteristic of neighborhood relations – if A is neighbor to B then B is also neighbor to A.

Antisymmetry is exhibited by $\leq$, the standard order on $\mathbb{R}$, as $x \leq y$ and $y \leq x$ entails $x = y$. Relations can be at the same time symmetric and antisymmetric: An example is the equality relation. Antisymmetry will also occur in the axiomatic definition of a partial order, discussed later. Asymmetry, not to be confused with antisymmetry, is somehow the counterpart of symmetry. It is also a typical characteristic of strictly ordered sets – for instance $<$ on $\mathbb{R}$.

An example of a binary relation (which is not an order) that obeys the transitivity axiom is the path-accessibility relation in directed graphs. If node B can be reached from node A via a path, and node C can reached from node B via a path, then also node C can be reached from node A via a path.

2.2 Preorders

Next we will introduce preorders and some properties on them. Preorders are a very general type of orders. Partial orders and linear orders are preorders that obey additional axioms. Beside other reasons these types of orders are important, because the Pareto order used in optimization defines a partial order on the objective space and a pre-order on the search space.

Definition 7 *Preorder*
A preorder *(quasi-order) is a binary relation that is both transitive and reflexive. We write* $\mathbf{x}^1 \preceq_{pre} \mathbf{x}^2$ *as shorthand for* $\mathbf{x}^1 \mathcal{R} \mathbf{x}^2$*. We call* $(\mathcal{S}, \preceq_{pre})$ *a* preordered set*.*

In the sequel we use the terms preorder and order interchangeably. Closely related to this definition are the following derived notions:

Definition 8 *Strict preference*
$\mathbf{x}^1 \prec_{pre} \mathbf{x}^2 :\Leftrightarrow \mathbf{x}^1 \preceq_{pre} \mathbf{x}^2 \wedge \neg(\mathbf{x}^2 \preceq_{pre} \mathbf{x}^1)$

Definition 9 *Indifference*
$\mathbf{x}^1 \sim_{pre} \mathbf{x}^2 :\Leftrightarrow \mathbf{x}^1 \preceq_{pre} \mathbf{x}^2 \wedge \mathbf{x}^2 \preceq_{pre} \mathbf{x}^1$

Definition 10 *Incomparability*
A pair of solutions $\mathbf{x}^1, \mathbf{x}^2 \in \mathcal{S}$ *is said to be incomparable, iff neither* $\mathbf{x}^1 \preceq_{pre} \mathbf{x}^2$ *nor* $\mathbf{x}^2 \preceq_{pre} \mathbf{x}^1$*. We write* $\mathbf{x}^1 || \mathbf{x}^2$*.*

Strict preference is irreflexive and transitive, and, as a consequence asymmetric. Indifference is reflexive, transitive, and symmetric. The properties of the incomparability relation we leave for exercise.

Having discussed binary relations in the context of pre-orders, let us now turn to characteristics of pre-ordered sets. One important characteristic of pre-orders in the context of optimization is that they are elementary structures on which minimal and maximal elements can be defined. Minimal elements of a pre-ordered set are elements that are not preceded by any other element.

Definition 11 *Minimal and maximal elements of an pre-ordered set* $\mathcal{S}$
$\mathbf{x}^1 \in \mathcal{S}$ *is minimal, if and only if not exists* $\mathbf{x}^2 \in \mathcal{S}$ *such that* $\mathbf{x}^2 \prec_{pre} \mathbf{x}^1$
$\mathbf{x}^1 \in \mathcal{S}$ *is maximal, if and only if not exists* $\mathbf{x}^2 \in \mathcal{S}$ *such that* $\mathbf{x}^1 \prec_{pre} \mathbf{x}^2$

Proposition 12 *For every finite set (excluding here the empty set* $\emptyset$*) there exists at least one minimal and at least one maximal element.*

For infinite sets, pre-orders with infinite many minimal (maximal) elements can be defined and also sets with no minimal (maximal) elements at all, such as the natural numbers with the order $<$ defined on them, for which there exists no maximal element. Turning the argument around, one could elegantly define an infinite set as a non-empty set on which there exists a pre-order that has no maximal element.

In absence of any additional information the number of pairwise comparisons required to find all minimal (or maximal) elements of a finite pre-ordered set of size $|\mathcal{X}| = n$ is $\binom{n}{2} = \frac{(n-1)n}{2}$. This follows from the effort required to find the minima in the special case where all elements are mutually incomparable.

2.3 Partial orders

Pareto domination imposes a partial order on a set of criterion vectors. The definition of a partial order is more strict than that of a pre-order:

Definition 13 *Partial order*
A **partial order** *is a preorder that is also* antisymmetric. *We call* $(\mathcal{S}, \preceq_{partial})$ *a partially ordered set* or **poset**.

As partial orders are a specialization of preorders, we can define *strict preference* and *indifference* as before. Note, that for partial orders two elements that are indifferent to each other are always equal: $\mathbf{x}^1 \sim \mathbf{x}^2 \Rightarrow \mathbf{x}^1 = \mathbf{x}^2$

To better understand the difference between pre-ordered sets and posets let us illustrate it by means of two examples:

Example
A pre-ordered set that is not a partially ordered set is the set of complex numbers $\mathbb{C}$ with the following precedence relation:

$$\forall(z_1, z_2) \in \mathbb{C}^2 : z_1 \preceq z_2 :\Leftrightarrow |z_1| \leq |z_2|.$$

It is easy to verify reflexivity and transitivity of this relation. Hence, $\preceq$ defines a pre-order on $\mathbb{C}$. However, we can easily find an example that proves that antisymmetry does not hold. Consider two distinct complex numbers $z = -1$ and $z' = 1$ on the unit sphere (i.e. with $|z| = |z'| = 1$. In this case $z \preceq z'$ and $z' \preceq z$ but $z \neq z'$ ∎

Example
An example for a partially ordered set is the subset relation $\subseteq$ on the power set[2] $\wp(S)$ of some finite set S. Reflexivity is given as $A \subseteq A$ for all $A \in \wp(S)$. Transitivity is fulfilled, because $A \subseteq B$ and $B \subseteq C$ implies $A \subseteq C$, for all triples (A, B, C) in $\wp(S) \times \wp(S) \times \wp(S)$. Finally, antisymmetry is fulfilled, since $A \subseteq B$ and $B \subseteq A$ implies $A = B$ for all pairs $(A, B) \in \wp(S) \times \wp(S)$ ∎

Remark In general the Pareto order on the search space is a preorder which is not always a partial order in contrast to the Pareto order defined on the objective space (that is, the Pareto order is always a partial order).

[2]the power set of a set is the set of all subsets including the empty set

2.4 Linear orders and anti-chains

Perhaps the most well-known specializations of a partially ordered sets are linear orders. Examples for linear orders are the $\leq$ relations on the set of real numbers or integers. These types of orders play an important role in single criterion optimization, while in the more general case of multiobjective optimization we deal typically with partial orders that are not linear orders.

Definition 14 (Linear order) *A* **linear** *(or:***total) order** *is a partial order that satisfies also the* comparability *or* totality *axiom:* $\forall \mathbf{x}^1, \mathbf{x}^2 :\in \mathcal{X} : \mathbf{x}^1 \preceq \mathbf{x}^2 \vee \mathbf{x}^2 \preceq \mathbf{x}^1$

Totality is only axiom that distinguishes partial orders from linear orders. This also explains the name 'partial' order. The 'partiality' essentially refers to the fact that not all elements in a set can be compared, and thus, as opposed to linear orders, there are incomparable pairs.

A linearly ordered set is also called a (also called *chain*). The counterpart of the chain is the anti-chain:

Definition 15 (Anti-chain) *A poset* $(\mathcal{S}, \preceq_{partial})$ *is said to be an* **antichain***, iff:* $\forall \mathbf{x}^1, \mathbf{x}^2 \in \mathcal{S} : \mathbf{x}^1 || \mathbf{x}^2$

When looking at sets on which a Pareto dominance relation $\preceq$ is defined, we encounter subsets that can be classified as anti-chains and subsets that can be classified as linear orders, or non of these two. Examples of anti-chains are *Pareto fronts*.

Subsets of ordered sets that form anti-chain play an important role in characterizing the time complexity when searching for minimal elements, as the following recent result shows [21]:

Theorem 16 (Finding minima of bounded width posets) *Given a poset* $(\mathcal{X}, \preceq_{partial})$*, then its* width w *is defined the maximal size of a mutually non-dominated subset. Finding the minimal elements of a poset of size* n *and width of size* w *has a time complexity in* $\Theta(wn)$ *and an algorithm has been specified that has this time complexity.*

In [21] a proof for this theorem is provided and efficient algorithms.

2.5 Hasse diagrams

One of the most attractive features of pre-ordered sets, and thus also for partially ordered sets is that they can be graphically represented. This is commonly done by so-called Hasse diagrams, named after the mathematician Helmut Hasse (1898 -

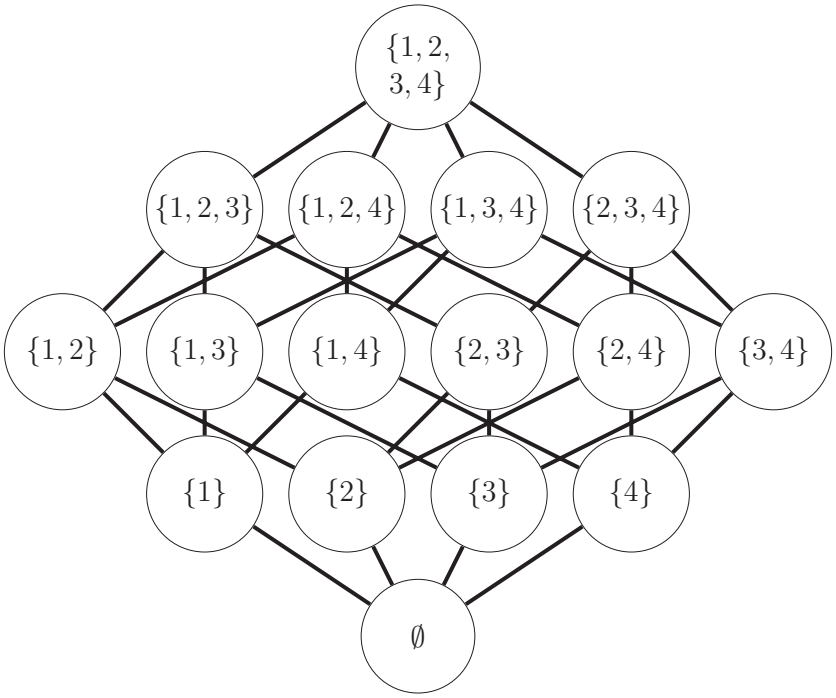

Figure 2.1: The Hasse Diagram for the set of all non-empty subsets partially ordered by means of $\subseteq$.

1979). The advantage of these diagrams, as compared to the graph representation of binary relations is essentially that edges that can be deduced by transitivity are omitted.

For the purpose of description we need to introduce the **covers** relation:

Definition 17 (Covers relation) *Given two elements* $\mathbf{x}^1$ *and* $\mathbf{x}^1$ *from a poset* $(\mathcal{X}, \prec_{partial})$. *Then* $\mathbf{x}^2$ covers $\mathbf{x}^1$, *in symbols* $\mathbf{x}^1 \lhd \mathbf{x}^2 :\Leftrightarrow \mathbf{x}^1 \prec_{partial} \mathbf{x}^2$ *and* $\mathbf{x}^1 \preceq_{partial} \mathbf{x}^3 \prec_{partial} \mathbf{x}^2$ *implies* $\mathbf{x}^1 = \mathbf{x}^3$.

One may also define the covers relation in more informal terms as: $\mathbf{x^2}$ covers $\mathbf{x^1}$ if and only if no element lies strictly between $\mathbf{x^1}$ and $\mathbf{x^2}$.

As an example, consider the covers relation on the linearly ordered set $(\mathbb{N}, \leq)$. Here $\mathbf{x}^1 \lhd \mathbf{x}^2$, iff $\mathbf{x}^2 = \mathbf{x}^1 + 1$. Note, that for $(\mathbb{R}, \leq)$ the covers relation is the empty set.

Another example where the covers relation has a simple interpretation is the subset relation $\subseteq$. In this example a set A is covered by a set B, if and only if B contains one additional element. In Fig. 2.1 the subset relation is summarized

in a Hasse diagram. In this diagram the cover relation defines the arcs. A good description of the algorithm to draw a Hasse diagram has been provided by Davey and Priestly ([22], page 11):

Algorithm 1 Drawing the Hasse Diagram

1: To each point $\mathbf{x} \in S$ assign a point $p(\mathbf{x})$, depicted by a small circle with centre $p(\mathbf{x})$
2: For each covering pair $\mathbf{x}^1$ and $\mathbf{x}^2$ draw a line segment $\ell(\mathbf{x}^1, \mathbf{x}^2)$.
3: Choose the center of circles in a way such that:
4: whenever $\mathbf{x}^1 \lhd \mathbf{x}^2$, then $p(\mathbf{x}^1)$ is positioned below $p(\mathbf{x}^2)$.
5: if $\mathbf{x}^3 \neq \mathbf{x}^1$ and $\mathbf{x}^3 \neq \mathbf{x}^2$, then the circle of $\mathbf{x}^3$ does not intersect the line segment $\ell(\mathbf{x}^1, \mathbf{x}^2)$

There are many ways of how to draw a Hasse diagram for a given order. Davey and Priestly [22] note that diagram-drawing is 'as much an science as an art'. Good diagrams should provide an intuition for symmetries and regularities, and avoid crossing edges.

2.6 Comparing ordered sets

(Pre)ordered sets can be compared directly and on a structural level. Consider the four orderings depicted in the Hasse diagrams of Fig. 2.2. It should be immediately clear, that the first two orders $(\preceq_1, \preceq_2)$ on X have the same structure, but they arrange elements in a different way, while orders $\preceq_1$ and $\preceq_3$ also differ in their structure. Moreover, it is evident that all comparisons defined in $\prec_1$ are also defined in $\prec_3$, but not vice versa (e.g. c and b are incomparable in $\preceq_1$). The ordered set on $\preceq_3$ is an *extension* of the ordered set $\preceq_1$. Another extension of $\preceq_1$ is given with $\preceq_4$.

Let us now define these concepts formally:

Definition 18 (Order equality) *An ordered set $(X, \preceq)$ is said to be* equal *to an ordered set $(X, \preceq')$, iff $\forall x, y \in X : x \preceq y \Leftrightarrow x \preceq' y$.*

Definition 19 (Order isomorphism) *An ordered set $(X', \prec')$ is said to be an* isomorphic *to an ordered set $(X, \preceq)$, iff there exists a mapping $\phi : X \to X'$ such that $\forall x, x' \in X : x \preceq x' \Leftrightarrow \phi(x) \preceq' \phi(x')$. In case of two isomorphic orders, a mapping ϕ is said to be an* order embedding map *or* order isomorphism.

Definition 20 (Order extension) *An ordered set $(X, \prec')$ is said to be an* extension *of an ordered set $(X, \prec)$, iff $\forall x, x' \in X : x \prec x' \Rightarrow x \prec' x'$. In the latter*

case, $\prec'$ is said to be compatible with $\prec$. A linear extension *is an extension that is totally ordered.*

Linear extensions play a vital role in the theory of multi-objective optimization. For Pareto orders on continuous vector spaces linear extensions can be easily obtained by means of any weighted sum scalarization with positive weights. In general, topological sorting can serve as a means to obtain linear extensions. Both topics will be dealt with in more detail later in this work. For now, it should be clear that there can be many extensions of the same order, as in the example of Fig. 2.2, where $(X, \preceq_3)$ and $(X, \preceq_4)$ are both (linear) extensions of $(X, \preceq_1)$.

Apart from extensions, one may also ask if the structure of an ordered set is contained as a substructure of another ordered set.

Definition 21 *Given two ordered sets $(X, \preceq)$ and $(X', \preceq')$. A map $\phi : X \to X'$ is called* order preserving*, iff $\forall x, x' \in X : x \preceq x' \Rightarrow \phi(x) \preceq \phi(x')$.*

Whenever $(X, \preceq)$ is an extension of $(X, \preceq')$ the identity map serves as an order preserving map. An order embedding map is always order preserving, but not vice versa.

There is a rich theory on the topic of partial orders and it is still rapidly growing. Despite the simple axioms that define the structure of the poset, there is a remarkably deep theory even on finite, partially ordered sets. The number of ordered sets that can be defined on a finite set with n members, denoted with s_n, evolves as

$$\{s_n\}_1^\infty = \{1, 3, 19, 219, 4231, 130023, 6129859, 431723379, \dots\} \tag{2.1}$$

and the number of equivalence classes, i.e. classes that contain only isomorphic structures, denoted with S_n, evolves as:

$$\{S_n\}_1^\infty = \{1, 2, 5, 16, 63, 318, 2045, 16999, ...\} \tag{2.2}$$

. See Finch [34] for both of these results. This indicates how rapidly the structural variety of orders grows with increasing n. Up to now, no closed form expressions for the growth of the number of partial orders are known [34].

2.7 Cone orders

There is a large class of partial orders on $\mathbb{R}^m$ that can be defined geometrically by means of cones. In particular the so-called *cone orders* belong to this class. Cone orders satisfy two additional axioms. These are:

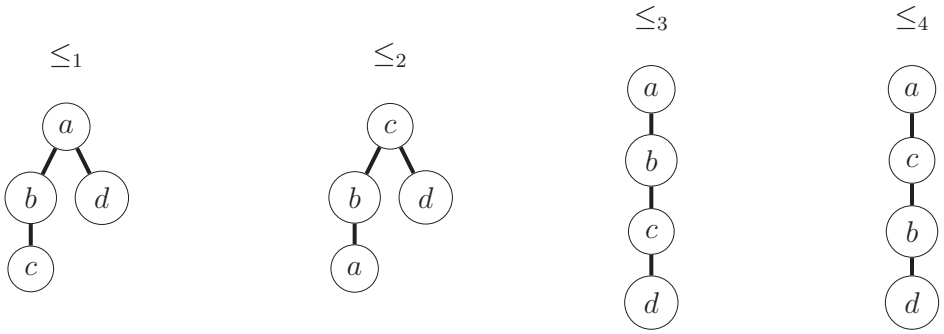

Figure 2.2: Different orders over the set $X = \{a, b, c, d\}$

Definition 22 (Translation invariance) *Let $\mathcal{R} \in \mathbb{R}^m \times \mathbb{R}^m$ denote a binary relation on $\mathbb{R}^m$. Then $\mathbb{R}$ is translation invariant, if and only if for all $\mathbf{t} \in \mathbf{R}^m$, $\mathbf{x}^1 \in \mathbb{R}^m$ and $\mathbf{x}^2 \in \mathbb{R}^m$: $\mathbf{x}^1 \mathcal{R} \mathbf{x}^2$, if and only if $(\mathbf{x}^1 + \mathbf{t})\mathcal{R}(\mathbf{x}^2 + \mathbf{t})$.*

Definition 23 (Multiplication invariance) *Let $\mathcal{R} \in \mathbb{R}^m \times \mathbb{R}^m$ denote a binary relation on $\mathbb{R}^m$. Then $\mathbb{R}$ is multiplication invariant, if and only if for all $\alpha \in \mathbf{R}$, $\mathbf{x}^1 \in \mathbb{R}^m$ and $\mathbf{x}^2 \in \mathbb{R}^m$: $\mathbf{x}^1 \mathcal{R} \mathbf{x}^2$, if and only if $(\alpha \mathbf{x}^1)\mathcal{R}(\alpha \mathbf{x}^2)$.*

We may also define these axioms on some other (vector) space on which translation and scalar multiplication is defined, but restrict ourselves to $\mathbb{R}^m$ as our interest is mainly to compare vectors of objective function values.

It has been found by V. Noghin [65] that the only partial orders on $\mathbb{R}^m$ that satisfy these two additional axioms are the cone orders on $\mathbb{R}^m$ defined by polyhedral cones. The Pareto dominance order is a special case of a strict cone order. Here the definition of strictness is inherited from the pre-order.

Cone orders can be defined geometrically and doing so provides a good intuition about their properties and minimal sets.

Definition 24 (Cone) *A subset $\mathcal{C} \subseteq \mathbb{R}^m$ is called a cone, iff $\alpha \mathbf{d} \in \mathcal{C}$ for all $\mathbf{d} \in \mathcal{C}$ and for all $\alpha \in \mathbb{R}, \alpha > 0$.*

In order to deal with cones it is useful to introduce notations for set-based calculus by Minkowski:

Definition 25 (Minkowski sum) *The Minkowski sum of two subsets S^1 and S^2 of $\mathbb{R}^m$ is defined as $S^1 + S^2 := \{s^1 + s^2 | s^1 \in S^1, s^2 \in S^2\}$. If S^1 is a singleton $\{x\}$, we may write $s + S^2$ instead of $\{s\} + S^2$.*

Definition 26 (Minkowski product) *The Minkowski product of a scalar* $\alpha \in \mathbb{R}^n$ *and a set* $S \subset \mathbb{R}^n$ *is defined as* $\alpha S := \{\alpha s | s \in S\}$.

Among the many properties that may be defined for a cone, we highlight the following two:

Definition 27 (Properties of cones) *A cone* $\mathcal{C} \in \mathbb{R}^m$ *is called:*

- *nontrivial or proper, iff* $\mathcal{C} \neq \emptyset$.
- *convex, iff* $\alpha \mathbf{d}^1 + (1-\alpha)\mathbf{d}^2 \in \mathcal{C}$ *for all* $\mathbf{d}^1$ *and* $\mathbf{d}^2 \in \mathcal{C}$ *for all* $0 < \alpha < 1$
- *pointed, iff for* $\mathbf{d} \in \mathcal{C}, \mathbf{d} \neq 0, -\mathbf{d} \notin \mathcal{C}$, *i e.* $\mathcal{C} \cap -\mathcal{C} \subseteq \{0\}$

Example As an example of a cone consider the possible futures of a particle in a 2-D world that can move with a maximal speed of c in all directions: This cone is defined as $\mathcal{C}^+ = \{\mathcal{D}(t) | t \in \mathbb{R}^+\}$, where $\mathcal{D}(t) = \{\mathbf{x} \in \mathbb{R}^3 | (x_1)^2 + (x_2)^2 \leq (ct)^2, x_3 = t\}$. Here time is measured by negative and positive values of t, where $t = 0$ represents the current time. We may ask now, whether given the current position $\mathbf{x}_0$ of a particle, a locus $\mathbf{x} \in \mathbb{R}^3$ is a possible future of the particle. The answer is in the affirmative, iff $\mathbf{x}_0$ if $\mathbf{x} \in \mathbf{x}_0 + \mathcal{C}^+$.

We will now can define Pareto dominance and the weak (strict) componentwise order by means of dominance cones. For this we have to define special convex cones in $\mathbb{R}$:

Definition 28 (Orthants) *We define*

- *the* positive orthant $\mathbb{R}^n_{\geq} := \{\mathbf{x} \in \mathbb{R}^n | x_1 \geq 0, \ldots, x_n \geq 0\}$.
- *the* null-dominated orthant $\mathbb{R}^n_{\prec_{pareto}} := \{\mathbf{x} \in \mathbb{R}^n | 0 \prec_{pareto} \mathbf{x}\}$.
- *the* strictly positive orthant $\mathbb{R}^n_{>} := \{\mathbf{x} \in \mathbb{R}^n | x_1 > 0, \ldots, x_n > 0\}$.

Now, let us introduce the alternative definitions for Pareto dominance:

Definition 29 (Pareto dominance) *Given two vectors* $\mathbf{x} \in \mathbb{R}^\mathbf{n}$ *and* $\mathbf{x}' \in \mathbb{R}^\mathbf{n}$*:*

- $\mathbf{x} < \mathbf{x}'$ *(in symbols:* $\mathbf{x}$ *dominates* $\mathbf{x}'$*) in the strict componentwise order* $\Leftrightarrow$ $\mathbf{x}' \in \mathbf{x} + \mathbb{R}^n_{>}$
- $\mathbf{x} \prec \mathbf{x}'$ *(in symbols:* $\mathbf{x}$ *dominates* $\mathbf{x}'$*)* $\Leftrightarrow \mathbf{x}' \in \mathbf{x} + \mathbb{R}^n_{\prec_{pareto}}$
- $\mathbf{x} \geq \mathbf{x}'$ *(in symbols:* $\mathbf{x}$ *dominates* $\mathbf{x}'$*) in the weak componentwise order* $\Leftrightarrow$ $\mathbf{x}' \in \mathbf{x} - \mathbb{R}^n_{\geq}$

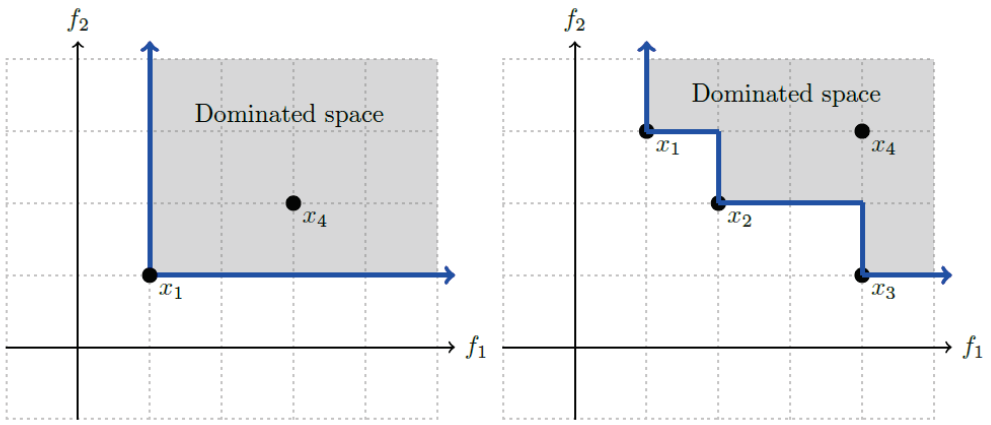

Figure 2.3: Pareto domination in $\mathbb{R}^2$ defined by means of cones. In the left hand side of the figure the points inside the dominated region are dominated by $\mathbf{x}$. In the figure on the right side the set of points dominated by the set $A = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$ is depicted.

It is often easier to assess graphically whether a point dominates another point by looking at cones (cf. Fig. 2.3 (l)). This holds also for a region that is dominated by a *set of points*, such that at least one point from the set dominates it (cf. Fig. 2.3 (r)).

Definition 30 (Dominance by a set of points) *A point* $\mathbf{x}$ *is said to be dominated by a set of points* A *(notation:* $A \prec \mathbf{x}$*, iff* $\mathbf{x} \in A + \mathbb{R}^n_{\prec}$*, i. e. iff there exists a point* $\mathbf{x}' \in \mathbf{A}$*, such that* $\mathbf{x}' \prec_{Pareto} \mathbf{x}$*.*

In the theory of multiobjective and constrained optimization, so-called polyhedral cones play a crucial role.

Definition 31 *A cone* $\mathcal{C}$ *is a polyhedral cone with a finite basis, if and only if there is a set of vectors* $D = \{\mathbf{d}_1, \ldots, \mathbf{d}_k\} \subset \mathbb{R}^m$ *and* $\mathcal{C} = \{\lambda_1\mathbf{d}_1 + \cdots + \lambda_k\mathbf{d}_k | \lambda \in \mathbb{R}^+_0, i = 1, \ldots, k\}$.

By chosing the coordinate vectors $\mathbf{e}_i$ to construct a polyhedral cones that resemble of the weak componentwise order.

Example In figure 2.4 an example of an cone is depicted with finite basis $D = \{\mathbf{d}_1, \mathbf{d}_2\}$ and $\mathbf{d}_1 = (2, 1)^\top, \mathbf{d}_2 = (1, 2)^\top$. It is defined as

$$C := \{\lambda_1\mathbf{d}_1 + \lambda_2\mathbf{d}_2 |\, \lambda_1 \in [0, \infty], \lambda_2 \in [0, \infty]\}\,.$$

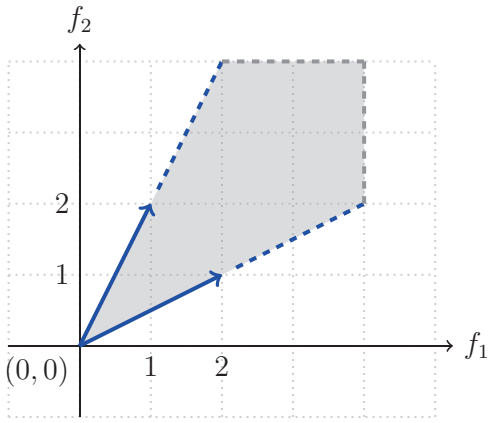

Figure 2.4: Dominance cone for cone order in Example 2.7.

This cone is pointed, because $C \cap -C = \emptyset$. Moreover, C is a convex cone. This is because two points in C, say $\mathbf{p}_1$ and $\mathbf{p}_2$ can be expressed by $\mathbf{p}_1 = \lambda_{11}\mathbf{d}_1 + \lambda_{21}\mathbf{d}_2$ and $\mathbf{p}_2 = \lambda_{12}\mathbf{d}_1 + \lambda_{22}\mathbf{d}_2$, $\lambda_{ij} \in [0, \infty)$, $i = 1, 2; j = 1, 2$. Now, for a given $\lambda \in [0, 1]$ $\lambda\mathbf{p}_1 + (1-\lambda)\mathbf{p}_2$ equals $\lambda\lambda_{11}\mathbf{d}_1 + (1-\lambda)\lambda_{12}\mathbf{d}_1 + \lambda\lambda_{21}\mathbf{d}_2 + (1-\lambda)\lambda_{22}\mathbf{d}_2 =: c_1\mathbf{d}_1 + c_2\mathbf{d}_2$, where it holds that $c_1 \in [0, \infty)$ and $c_2 \in [0, \infty)$. According to the definition of C the cone therefore the point $\lambda\mathbf{p}_1 + (1 - \lambda)\mathbf{p}_2$ is part of the cone C.

Further topics related to cone orders are addressed in [28].

Exercises

1. In definition 6 some common properties are defined that binary relations can have and some examples are given below. Find further examples from real-life for binary relations! Which axioms from definition 6 do they obey!

2. Characterize incomparability (definition 10) axiomatically! What are the essential differences to indifference?

3. Describe the Pareto order on the set of 3-D hypercube edges $\{(0, 1, 0)^T$, $(0, 0, 1)^T, (1, 0, 0)^T, (0, 0, 0)^T, (0, 1, 1)^T, (1, 0, 1), (1, 1, 0)^T, (1, 1, 1)^T\}$ by means of the graph of a binary relation and by means of the Hasse diagram!

4. Prove, that $(\mathbb{N} - \{1\}, \preceq)$ with $a \preceq b \Leftrightarrow a \mod b \equiv 0$ is a partially ordered set. What are the minimal (maximal) elements of this set?

5. Prove that the time cone $\mathcal{C}^+$ is convex! Compare the Pareto order to the order defined by time cones!

Chapter 3

Pareto optima and efficient points

In this chapter we will come back to optimization problems, as defined in the first chapter. We will introduce different notions of Pareto optimality and discuss necessary and sufficient conditions for (Pareto) optimality and efficiency in the constrained and unconstrained case. In many cases, optimality conditions directly point to solution methods for optimization problems. As in Pareto optimization there is rather a set of optimal solutions then a single optimal solution, we will also look at possible structures of optimal sets.

3.1 Search Space vs. Objective Space

In Pareto optimization we are considering two spaces - the *decision space* or *search space* $\mathbb{S}$ and the *objective space* $\mathbb{Y}$. The vector valued objective function $\mathbf{f} : \mathbb{S} \to \mathbb{Y}$ provides the mapping from the decision space to the objective space. The set of feasible solutions $\mathcal{X}$ can be considered as a subset of the decision space, i. e. $\mathcal{X} \subseteq \mathbb{S}$. Given a set $\mathcal{X}$ of feasible solutions, we can define $\mathcal{Y}$ as the image of $\mathcal{X}$ under $\mathbf{f}$.

The sets $\mathbb{S}$ and $\mathbb{Y}$ are usually not arbitrary sets. If we want to define optimization tasks, it is mandatory that an order structure is defined on $\mathbb{Y}$. The space $\mathbb{S}$ is usually equipped with a neighborhood structure. This neighborhood structure is not needed for defining global optima, but it is exploited, however, by optimization algorithms that gradually approach optima and in the formulation of local optimality conditions. Note, that the choice of neighborhood system may influence the difficulty of an optimization problem significantly. Moreover, we note that the definition of neighborhood gives rise to many characterizations of functions, such as local optimality and barriers. Especially in discrete spaces the neighborhood structure needs to be mentioned then, while in continuous optimization locality mostly refers to the Euclidean metric.

The definition of landscape is useful to distinguish the general concept of a