FAKULTÄT FÜR !INFORMATIK

Interactive Optimization, Distance Computation and Data Estimation in Parallel Coordinates

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Computergraphik & Digitale Bildverarbeitung

eingereicht von

Matthias Froschauer
Matrikelnummer 9726227

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung:
Betreuer: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller
Mitwirkung: Dipl.-Ing. Harald Piringer

Wien, 25.02.2009

(Unterschrift Verfasser) (Unterschrift Betreuer)

Technische Universität Wien
A-1040 Wien · Karlsplatz 13 · Tel. +43/(0)1/58801-0 · http://www.tuwien.ac.at

Matthias Froschauer
Taborstraße 22/1/2a
1020 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, am 25.02.2009

Unterschrift

Abstract

The field of information visualization tries to find graphical representations of data to explore regions of interest in potentially large data sets. Additionally, the use of algorithms to obtain exact solutions, which cannot be provided by basic visualization techniques, is a common approach in data analysis. This work focuses on optimization, distance computation and data estimation algorithms in the context of information visualization.

Furthermore, information visualization is closely connected to interaction. To involve human abilities in the computation process, the goal is to embed these algorithms into an interactive environment. In an analysis dialog, the user observes the current solution, interprets the results and then formulates a strategy of how to proceed. This forms a tight loop of interaction, which uses human evaluation to improve the quality of the results.

Optimization is a crucial approach in decision making. This work presents an interactive optimization approach, exemplified by parallel coordinates, which are a common visualization technique when dealing with multi-dimensional problems. According to this goal-based approach, multi-dimensional distance computation is discussed as well as a data estimation approach with the objective of approximating simulations by the analysis of existing values.

All these approaches are integrated in an existing visual analysis framework and deal with multi-dimensional goals, which can be defined and modified interactively by the user. The goal of this work is to support decision makers to extract useful information from large data sets.

Kurzfassung

Das Anwendungsgebiet der Informationsvisualisierung versucht graphische Darstellungen von Daten zu finden, um relevante Teilbereiche in oftmals hochdimensionalen Datensätzen zu untersuchen. Zudem werden oft Algorithmen eingesetzt, die exakte Lösungen bereitstellen, welche durch elementare Visualisierungstechniken nicht vermittelt werden können. In der vorliegenden Arbeit werden die Themen Optimierung, Distanzberechnung und Datenabschätzung im Kontext der Informationsvisualisierung behandelt.

Darüber hinaus ist der Begriff Informationsvisualisierung eng mit Interaktion verbunden. Um menschliche Fähigkeiten in den Berechnungsprozess einzubeziehen, werden diese Algorithmen in eine interaktive Umgebung eingebunden. In einem Mensch-Maschine-Dialog beurteilt der Benutzer die derzeitige Lösung, interpretiert das Ergebnis und formuliert die weitere Vorgehensweise. Dies führt zu einem Interaktionskreislauf, der sich menschlicher Beurteilung bedient, um die Qualität der Lösung zu verbessern.

Optimierung ist ein äußerst wichtiger Ansatz in der Entscheidungsfindung. Die vorliegende Arbeit behandelt einen interaktiven Optimierungsansatz. Der Lösungsweg wird am Beispiel von parallelen Koordinaten erläutert, welche eine gängige Visualisierungstechnik in Verbindung mit mehrdimensionalen Problemen darstellen. In Anlehnung an diesen zielbasierten Ansatz wird mehrdimensionale Distanzberechnung diskutiert, sowie ein Ansatz zur Datenabschätzung mit dem Ziel, Simulationen durch die Analyse von vorhandenen Werten zu approximieren.

Die Algorithmen sind in ein bestehendes Visualisierungssystem integriert und wurden in Hinblick auf mehrdimensionale Ziele entwickelt, die interaktiv vom Benutzer definiert und modifiziert werden können. Das Ziel der Arbeit ist es, Entscheidungsträger bei der Suche nach zweckdienlichen Informationen in großen Datensätzen zu unterstützen.

Contents

Chapter 1

Introduction

The large amount of information available today bears enormous potential. Thus, it gets more and more important to find ways to determine and present sets of data which are relevant for a specific task. The comparatively young field of information visualization aims to give insight into the flood of data. However, the retrieved information is often visualization dependent, which means that only the actually displayed information can be explored. For example, figure 1.1 shows a scatter plot of a data set comparing cars. The image clearly indicates a relation between horsepower and fuel consumption (miles per gallon). To determine more relations, an advanced visualization technique can be used, like parallel coordinates. However, some tasks might need an exact solution that cannot be provided by basic visualization techniques. In this case, algorithms are necessary that prepare the desired results first.

The intention of this work is to examine ways to integrate algorithms into information visualization in order to solve complex optimization tasks, distance computation and data estimation.

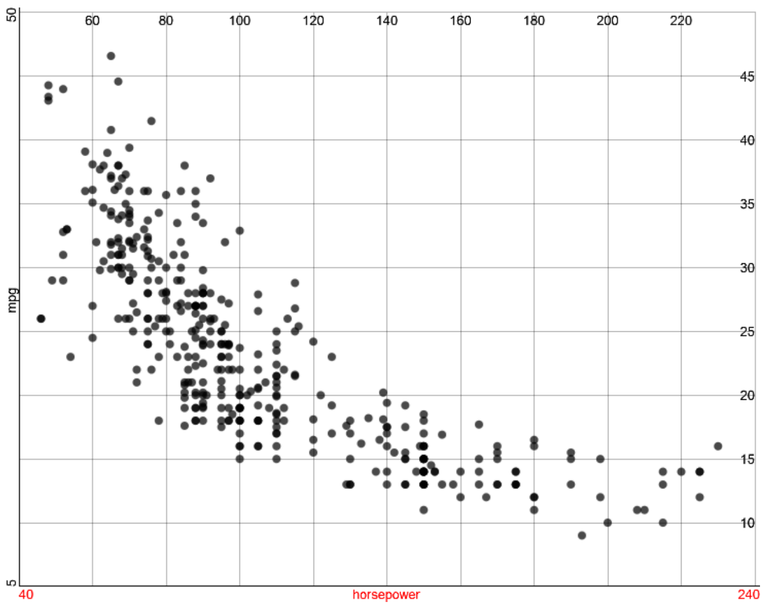


Figure 1.1: Scatter plot showing horsepower and fuel consumption (miles per gallon) of cars. Data courtesy of Donoho and Ramos [12].

1.1 Visualization

Visualization is the process of creating visual representations of data. The intention is to use knowledge about human perception for data analysis. The use of visual metaphors helps to amplify cognition. Visualization can be seen as a mapping from data to the human perception system. Data is mapped to visual structures, which meet the properties of perception. The challenge is to find a suitable visualization technique to give more insight into the data. At the same time, it has to be assured that no incorrect patterns are perceived, which would lead to incorrect decisions [34].

A reference model for mapping data to visual form was introduced by Card et al. [5]. This model is leading to the *visualization pipeline* (figure 1.2), which describes the process of converting raw data to visual representations. The first step is to acquire data by measurements, simulations or modeling. The goal of the *data transformation* step is to transform the raw data into appropriate data sets to work with. Often the data format is built upon data tables, where each data entity is represented by one row, containing a fixed number of dimensions (columns). This step also includes data enhancement like filtering, resampling and interpolation. For example, the data could require noise suppression or aggregation. Once the data is stored in a suitable form, a *visual mapping* is applied. The intention of this step is to generate renderable structures. The mapping approach depends on the data itself and the intention of the visualization. For example, volume data generated in medical applications is often mapped to geometry or point clouds, since an inherent spacial reference is given. On the other hand, abstract data can be represented by structures, whose appearance have no clear reference to the underlying data, like glyphs. The challenge in this step is to find suitable structures to represent the data and to support the user in the analysis of the data set. The final step is to generate an image by applying computer graphics methods. The *view transformation* handles visibility, illumination and other viewing parameters. Nowadays, this part is typically realized in a flexible way to support user interaction. By changing view parameters, the user is able to explore the data set. Apart from navigation actions like zooming, panning and rotations, other interaction methods may involve changing the color parameters and altering visibility.

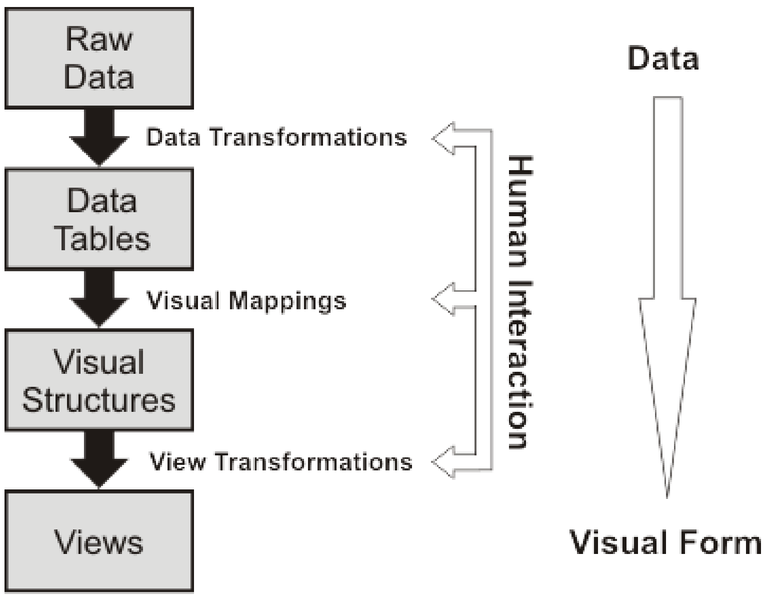


Figure 1.2: The visualization pipeline. It describes a reference model for mapping data to a visual form. Human interaction is used to tailor the visualization to specific needs. Image adapted from Card et al. [5].

1.1.1 Goals

The major goals of visualization are presentation, confirmatory analysis and exploratory analysis [26]. The Oxford English Dictionary (1989) defines visualization as follows:

> *"To form a mental vision, image, or picture of (something not visible or present to sight, or of an abstraction); to make visible to the mind or imagination."*

This describes a very abstract visualization term, but it becomes clear that visualization is more than visual information seeking. In this section, some goals and related application areas of visualization are given, as denoted by Keim [24]. Basically, the goals of visualization can be summarized by three types.

Exploration. Visualization can be used to create hypotheses. Typically, very little is a priori known about the data. In this case, visualization helps searching for structures, trends, etc. For example, in medical applications, visualization can be used to find anomalies in data sets generated be computer tomography. Interaction plays an important role in data exploration, especially when dealing with large data sets [5].

Analysis. If there are hypotheses about the data, visualization can be used for examination. The goal is to verify or to disprove these hypotheses. This also includes quality control of simulations and measurements. In the automotive industry, for example, flow visualization is used to improve the aerodynamics behavior by evaluating the car designer's work.

Presentation. If (almost) everything is known about the data, the third important visualization goal is to present the data. In this case, visualization serves for communication of the results. For example, the results of elections are often visualized as bar charts. The data is known and the intention is to present them in a way that provides an easy perception of the distribution of votes.

1.1.2 Scientific Visualization versus Information Visualization

Card et al. [5] give the following rough classification of visualization disciplines.

Volume visualization. This topic deals with representing 3D volumes and provides techniques to explore them. A typical application is the analysis of medical data.

Flow visualization. Flow visualization is the study of methods to display the dynamic behavior of flows. It is used in the analysis of liquid or gas flows, as well as in weather observation applications. The challenge here is to combine spatial and temporal variables in one visualization.

Information visualization. Information visualization focuses on representing abstract data. The goal is to deliver insight into (mostly large) data sets that often do not have any spatial or temporal reference.

Most other kinds of visualization like *illustrative visualization* and *software visualization* can be seen as subtypes of the general classification above. Volume visualization and flow visualization are often grouped under the term *scientific visualization*, since the applications in this area involve the visualization of physical data such as the human body, the earth, and so on. Unlike scientific visualization, information visualization deals with abstract, not necessarily physically-based data. The challenge with abstract data is to design visual representations. Often virtual objects are created that can be manipulated as if they were real objects. For example, figure 1.3 shows a visualization of hierarchical data. Inspired by nature, a file structure is visualized as a tree [29].

Rhyne [45] poses the question, whether this separation is justified anymore. There are visualization applications combining techniques from both types of visualization. The recent interest in visualizing bioinformatics data is mentioned as an example. Information visualization techniques are well suited for genomic data, as they use metaphors to deal with the inherently non-spatial characteristics of genomic data sets. On the other hand, there are biological and chemical aspects, which rather belong to the area of scientific visualization.

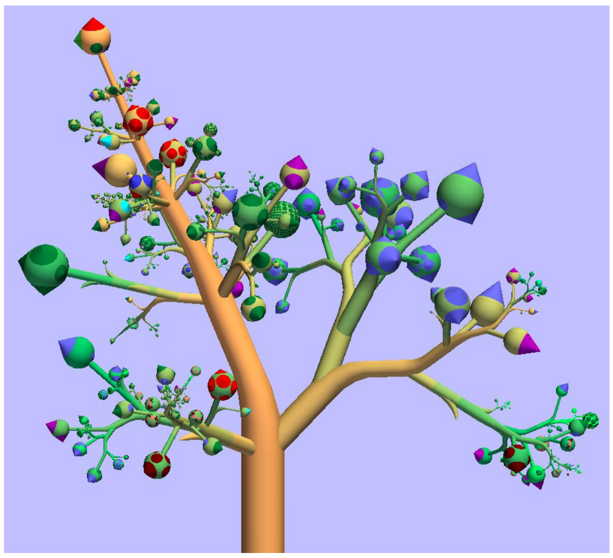


Figure 1.3: Botanic visualization of a Unix home-directory. The file structure is visualized as tree. Image from Kleiberg et al. [29].

Another issue is cartographic and geographic information. Weather modeling, environmental sciences, geology, and other earth sciences make use of cartographic metaphors as typically found in information visualization. At the same time, the inherent spacial reference and the clearly non-abstract data require scientific visualization and sometimes real-time rendering.

It turns out that many application areas benefit from both scientific and information visualization. Therefore, it is a legitimate question, if it is really necessary to distinguish between information and scientific visualization.

1.2 Data Enrichment

As mentioned in section 1.1, the visualization pipeline includes data enrichment in the data transformation step. Additional information may be derived from the source data that might be useful for a specific application. For example, computing gradients enhances the visualization of a 3D data set. Data enrichment in the present context is concerned with the creation of useful information about the underlying data set.

Data analysis is not only done by visualization. Applying algorithms on the data set is still a much more common approach, which is suitable to get specific information about certain characteristics. This section provides an overview of approaches, which are relevant for this work. All these approaches have in common that they rely on multi-dimensional goals.

The major contribution of this work is the integration of domain-specific automatic methods into information visualization. The approach given here defines a tight *loop of interaction*. Information visualization helps to explore the current solution. The user analyzes the solution and uses the new knowledge to modify the algorithm parameters, if the results are not satisfying. Once the interaction process is finished, the computation is restarted and the new solution is visualized. The interaction between the user and the combination of information visualization and the used algorithm forms the core of this work. Figure 1.4 illustrates this loop.

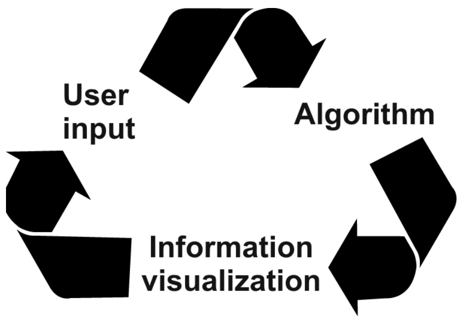


Figure 1.4: Loop of interaction. The tight coupling of user input, automated data processing and information visualization plays an important role in this work.

1.2.1 Optimization

Optimization is a crucial approach in decision making. The goal is to identify an optimal solution to a problem with respect to one or more objectives (i.e., maximizing or minimizing a function), where the parameter space is typically restricted by constraints. A classical optimization task is to achieve maximum benefit at minimum cost. Chapter 4 describes the computation and visualization of a specific optimization algorithm. This algorithm is embedded in an interactive environment.

1.2.2 Distance Computation

A useful data extension is to store an additional value to each data entry that describes the distance to a specified goal. The interactive approach presented in this work helps to detect entries, which are beneficial for a specific purpose as well as outliers. Several distance computation techniques are discussed in chapter 5, dealing with assigning numerical distance values to the data with respect to predefined multi-dimensional goals.

1.2.3 Data Estimation

This work describes a method to estimate the data characteristics at a predefined multi-dimensional goal. Based on an approximation approach, the intention is to provide estimations of not yet simulated parameterizations in order to get a "simulation preview". In the application context of analyzing multiple simulation runs, the purpose of this method is to spare costly simulations by approximation. In a loop of interaction, the user may specify and alter the multi-dimensional goal and the visualization system updates and displays the estimated result. Chapter 6 gives a detailed description of this task.

Chapter 2

State of the Art and Fundamentals

In the introduction the necessity of effective viewing techniques was discussed. This chapter presents approaches that were developed in the field of information visualization. In particular, the concept of parallel coordinates is discussed. The subsequent sections give an overview of multi-criteria and interactive optimization, as well as a brief discussion on memory-based reasoning. Furthermore, the Bulk Analyzer visualization tool is described.

2.1 Information Visualization

A study done by the University of Berkeley showed that in 2002 print, film, magnetic and optical storage media produced about 5 exabytes (five times 2^{60} bytes) of new data and this value increases every year [33]. The amount of data to deal with is huge in many areas. As an example, in the field of engineering large, multi-dimensional data sets are generated by measurements and simulations. Simple methods to display data such as lists or tables are of limited use for such amounts of complex data due to the lack of visual structures that can be perceived well by the human and the limited interaction potential [5]. The emerging and rapidly growing field of information visualization seems to offer solutions in order to get insight into the data, to show its structure, anomalies and behavior. The idea is to combine human cognition and perception with the computational power of computers. Card et al. [5] give the following definition:

"Information visualization: The use of computer-supported, interactive, visual representations of abstract data to amplify cognition."

2.1.1 Visualization Techniques

A classification of visualization techniques is not straightforward. Some techniques are combining several ideas and others are very specific to a certain application, like special topics in document visualization. Keim [25] classifies information visualization techniques by their basic visualization principle. This section briefly summarizes this classification and presents some examples.

Standard 2D/3D displays. Simple displays, such as bar charts and scatter plots, are often used since they are easy to understand and it is common knowledge to interpret them. The major disadvantage is that the number of shown dimensions is very limited.

Geometrically transformed displays. This kind of displays aims at mapping multi-dimensional data to a set of points or lines. The main idea is to transform the multi-dimensional data set into the two-dimensional space. This way, a high number of dimensions can be visualized. Typical examples here are scatter plot matrices or parallel coordinates, which are discussed in detail later in this chapter.

Icon-based displays. These displays try to map the attribute values of a multi-dimensional data item to the features of an icon. The number of displayable dimensions is not limited with this approach. However, they are not used very often with high-dimensional data sets, since a quick information exploration is problematic. The famous Chernoff faces or stick figure icons can be taken as examples.

Dense pixel displays. Pixel based displays, such as the circle segments technique [4], target at mapping each value to a colored pixel. Pixels are grouped according to the dimension the item belongs to and are arranged on the screen appropriate to different purposes (e.g., the most relevant data items may be presented in the center of the display). In general, one pixel is used per data value, so the number of displayable values is rather high.

Stacked displays. Stacked displays are based on a hierarchical subdivision of the data. For example, tree maps fill the screen space recursively, where the area reserved for each item is proportional to a selected attribute (e.g., the size of a file).

In the following, some examples are presented here that should demonstrate the wide range of techniques covered by the field of information visualization. Figure 2.1 shows a basic visualization of 2D and 3D data using *scatter plots*. Data values are simply mapped to a Cartesian coordinate system where each axis represents an attribute. As a well known visualization technique, scatter plots are used in many fields of application. Extensions to scatter plots have been introduced in order to

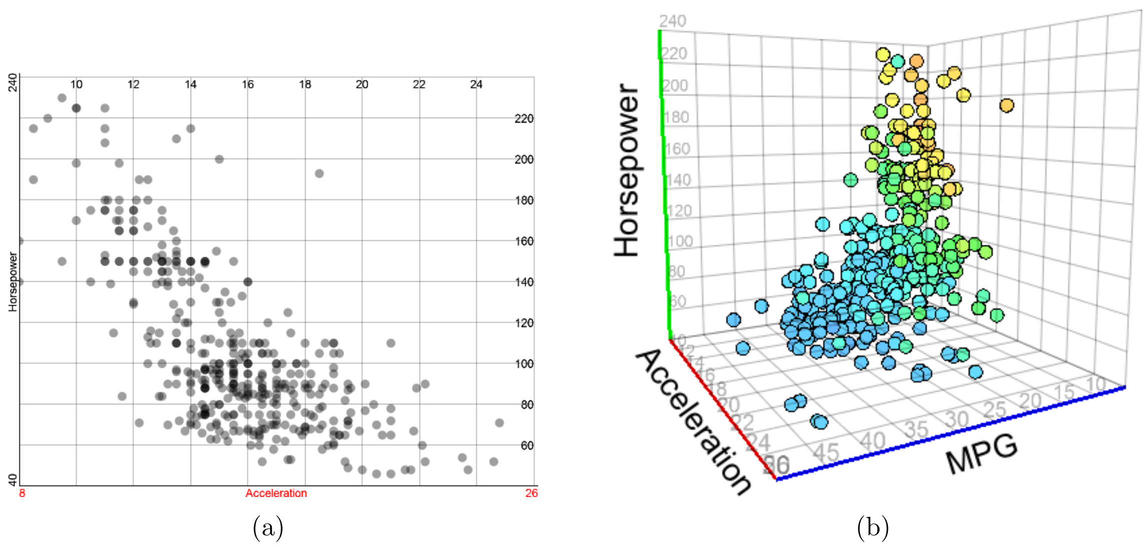
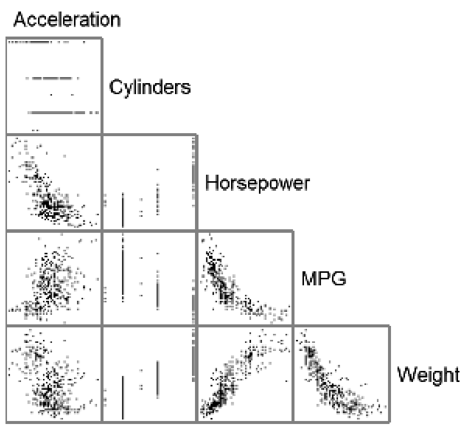
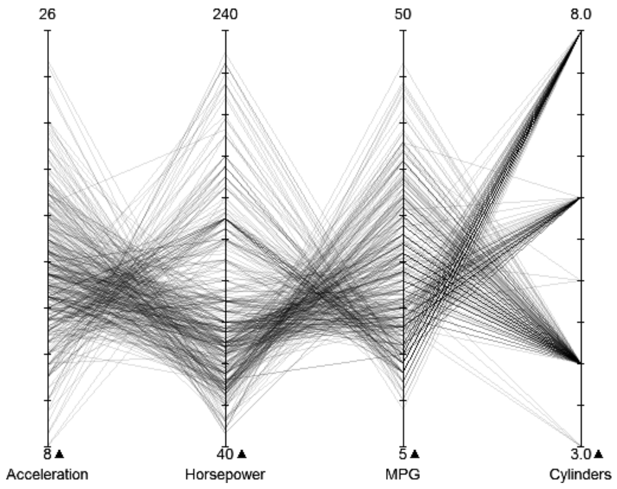


Figure 2.1: Visualization of low-dimensional data: (a) 2D scatter plot and (b) 3D scatter plot comparing cars. Data courtesy of Donoho and Ramos [12].

increase the number of displayable dimensions. The *scatter plot matrix* in figure 2.2 (a) represents each dimension plotted against each other in a matrix. Only one half of the matrix has to be displayed, since it is symmetric. The number of displayed dimensions is only limited by the physical screen space. Another idea to show a set of n-dimensional points is realized in *parallel coordinates*. As depicted in figure 2.2 (b), the axes are not set orthogonal, but parallel and every data point is represented as a line strip. The illustrations in this work focus on parallel coordinates, therefore this approach is discussed in detail in the next section.



(a)



(b)

Figure 2.2: Visualization in higher dimensions: (a) Scatter plot matrix in 5 dimensions and (b) parallel coordinates comparing cars. Data courtesy of Donoho and Ramos [12].

Icon-based displays follow a different approach. Data characteristics are represented by certain features and visual properties of icons (also referred to as glyphs). Figure 2.3 demonstrates this in two examples. The *Chernoff faces* in figure 2.3 (a) depict every n-dimensional data item as a face icon. Each dimension is represented by a face attribute (like head shape, nose length, and so on). Similar to this, *star glyphs* in figure 2.3 (b) consist of line segments radiating from a central point. The length of each line segment indicates the value of the corresponding data dimension. Although exact data values are hard to perceive with icons and glyphs, they are suitable to convey certain aspects efficiently and often make use of metaphors of the respective application domain.

Pixel-oriented visualization follows the pixel-per-value approach. As an example, the *circle segments technique* is shown in figure 2.4 (a). An important issue of pixel-oriented techniques is how the pixels are arranged on the screen. The idea of this example is to map a data set with m dimensions onto a circle with m segments. Each segment represents a different attribute. To improve the comparability between dimensions, the pixels are sorted (independently for each segment) and colored based upon their data value. It is intended for high dimensional data and each pixel stands for a single value of one dimension.

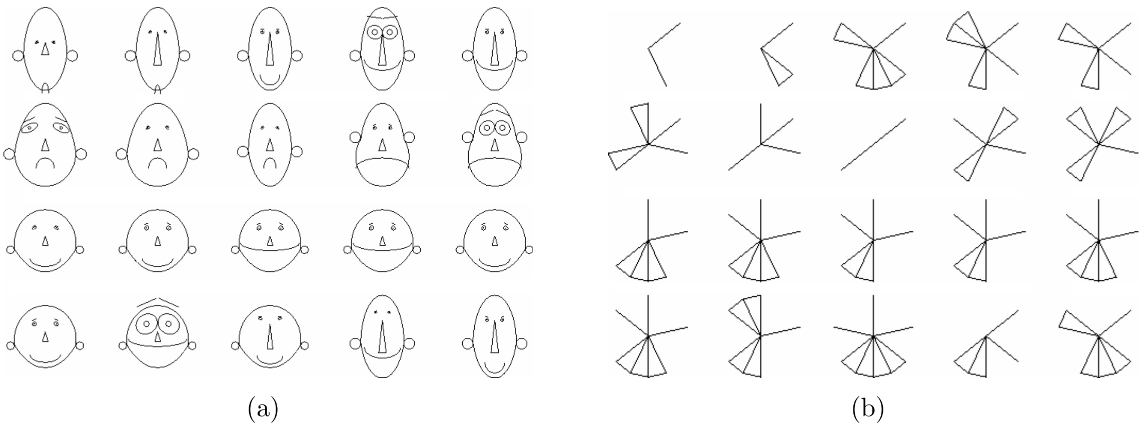


Figure 2.3: Icon-based visualization techniques: (a) Chernoff faces; (b) Star glyphs. Images from Lee et al. [31].

Another well-known visualization technique are *tree maps*. Tree maps belong to the class of stacked displays and are designed for visualizing hierarchical structures. Using size and color, leaf nodes and subtrees are encoded in a space-filling way. Figure 2.4 (b) shows a tree map representing a file system.

2.1.2 Parallel Coordinates

The challenge in visualizing multi-dimensional data is to map the desired information onto a two-dimensional display screen. Due to human physiology, the brain is trained to think in not more than three dimensions. To map a space of arbitrary dimensions onto the screen, techniques have to be applied that deal with this problem. Many visualization methods have been proposed and some of them were discussed in the previous section. One of the most important techniques for visualizing multi-dimensional data are *parallel coordinates*, introduced by Alfred Inselberg [22] in the 1980s. Since plotting more than three orthogonal axis is impossible, the dimensions are drawn as parallel lines, typically vertical and equally spaced. This way, there is no predefined restriction on the number of displayable dimensions. To show a set of points in an n-dimensional space, data values are depicted by polygonal lines, often called *polylines*. Each line intersects all axes and represents one data value in the n-dimensional space. The point of intersection with an axis is given by the mapped position of the value in this dimension. For $n = 2$ the result of this is a point $\leftrightarrow$ line

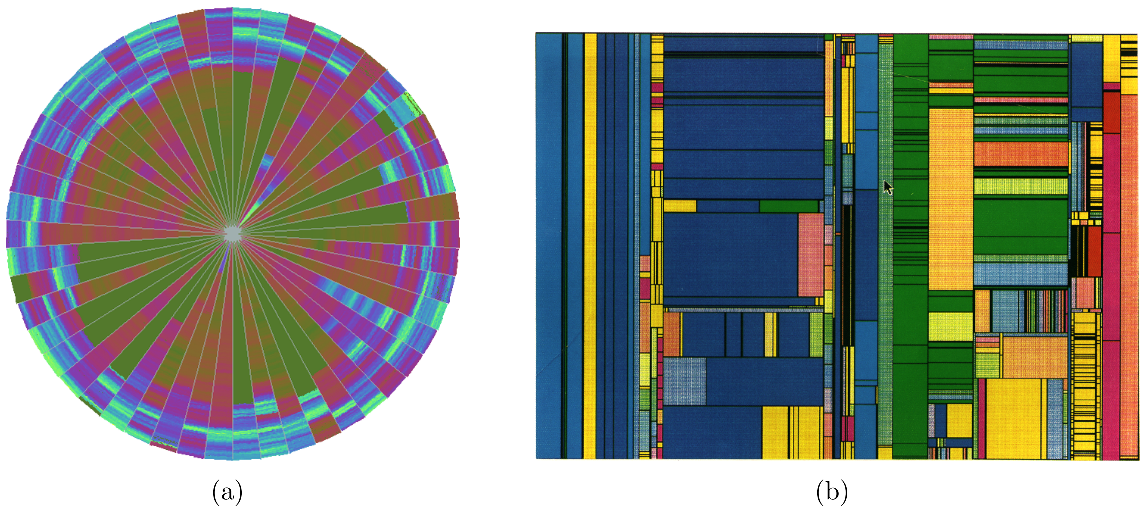


Figure 2.4: Dense pixel displays and stacked displays: (a) Circle segments technique representing about 265,000 data items in 50 dimensions; (b) Tree map representing a disk directory system containing 850 files, colored by file type. Images from Ankerst et al. [4] and Shneiderman [49].

duality (figure 2.5). Each point in the two-dimensional space represents a line in parallel coordinates. Vice versa, all lines in parallel coordinates that have a common intersection point, induce points in the plane, which are positioned on a line. This line corresponds to the intersection point in parallel coordinates. The order of axes influences the display a lot. To reduce clutter in this technique, it is sometimes helpful to rearrange the dimensions to minimize the outliers between neighboring dimensions [40]. A simple approach is to sort the dimensions according to the data characteristics (e.g., median, standard deviation, ...). Advanced methods can be used to determine the similarity of dimensions in order to find the optimal axis arrangement [3]. A useful axes order may also show clusters and correlations which are not visible in a different arrangement.

The parallel coordinates technique targets at continuous data variables. To use categorical data within parallel coordinates, the categories have to be mapped to numbers [46]. Because of the discrete nature of categorical data, polylines are bundled to few points in categorical dimensions and the space on each axis is not used efficiently. For purely categorical data, *parallel sets* are more practical, which adopt the layout of parallel coordinates with the idea of displaying frequencies as represen-

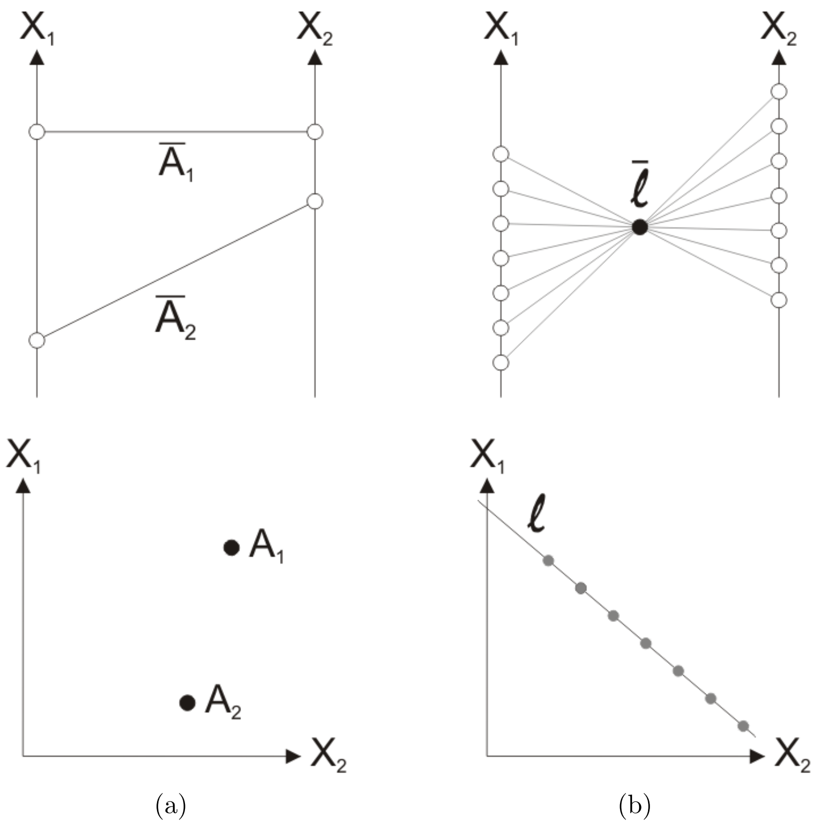


Figure 2.5: In the plane parallel coordinates induce duality: (a) Lines in parallel coordinates represent points in the scatter plot; (b) A point in parallel coordinates represents a (virtual) line in the scatter plot.

tatives for the categories [30]. One problem with this visualization technique is that lines usually need more space to be drawn than points. If there is a huge amount of data values, the lines will fill all the available space and it will be impossible to extract information of the visualization. Advanced drawing techniques or clustering algorithms are used to overcome this issue [21][15]. Also, interaction is a crucial component here to be able to explore the data conveniently, as described in the following section.

2.1.3 Interaction

Interaction plays an important role in information visualization. Adopting the visualization dynamically greatly facilitates the exploration of multi-dimensional data sets. A summary of interaction techniques is given in the following [25].

Dynamic projections. The dynamical change of projections allows to modify view transformations. An important example is camera movement, which allows the change of position of the observer. Viewing from another angle can avoid object occlusion, which is a major problem in dense information spaces [5].

Interactive filtering. The selection of desired subsets is very difficult for very large data sets. Therefore a number of interaction techniques have been developed to improve interactive filtering in data exploration. For example, *magic lenses* show a modified view of the selected region, while the rest of the visualization remains unaffected. The idea is to support filtering of the data directly in the visualization, where several lenses with different filters may be used [5].

Interactive zooming. Large data sets are often presented in a highly compressed form to provide an overview of the data. Zooming is used to display details, which does not only mean to display the data objects larger but it also means that the data representation automatically changes to present more details on higher zoom levels [25]. For example, the *table lens* approach implements interactive zooming, which is used to visualize large tabular data [43]. Each numerical value is represented by a small bar, which has a height of one pixel and the length is determined by the attribute value. In order to explore a region of interest the user can zoom in and the affected rows (or columns) are displayed in more detail, usually in textual form.

Interactive distortion. Distortion is a useful way to show specific portions of the data set with a higher level of detail while others are shown with a lower level of detail. This approach uses the *focus and context* concept, since the user needs both the overview (context) and detail information (focus) [5]. Popular techniques are hyperbolic and spherical distortion [25]. Examples of distortion techniques include *bifocal displays* [52] and *graphical fisheye views* [16].

Interactive linking and brushing. It is often necessary to combine several visualization techniques to overcome the shortcomings of single approaches, since all of them have strengths and weaknesses. Linking and brushing ensures that selected points are highlighted in all visualizations. Interactive changes made in one visualization are automatically reflected in the other visualizations [25].

Focusing on parallel coordinates in this work, interaction techniques specific for this kind of visualization are discussed in the following, which are also mentioned by Hauser et al. [21]. A more general discussion on direct manipulation of parallel coordinates is given by Siirtola [50].

Reordering of axes. One of the most important interaction features when working with parallel coordinates is the possibility to reorder axes. For example, to clearly indicate relations between the values of two dimensions, it is often necessary to put the appropriate axes side by side.

Flipping of axes. To improve the recognizability of correlations, it is useful to change the ordering of values on certain axes.

Changing the mapping. Normally, the value range of each axis is adjusted to the underlying data. Sometimes it it helpful to manually specify a range. It could also be advantageous to enforce that a specific value (e.g., zero) is mapped to the same height along all coordinate axes.

Clustering. One way to reduce visual clutter is clustering. The main cause of the clutter comes from too many polylines. To overcome this problem, polylines are grouped according to application-specific parameters (e.g., by similar values of a specific dimension). Zhou et al. [61] describe a clustering algorithm based on the geometric relationship between polylines.

2.2 Introduction to Optimization

Our daily life consists of making decisions every day. In many cases, real-world decision problems can be formulated as mathematical optimization problems. Optimization can be defined as the science of determining the "best" solutions to certain mathematically defined problems, which are often models of physical reality [13]. Optimization became an independent subject in the late 1940s, when *linear programming* methods were investigated [55]. Nowadays, modern optimization methods can solve large scale problems and are essential parts in many computer-assisted applications. To use optimization, an *objective* must be identified, which is a quantitative measure of the performance of the system under study [38]. This objective could be profit, time or any other quantity that can be represented by a single number. More complex optimization problems deal with more than one objective. These problems will be discussed in the next section. The general form of optimization problems is

$$\begin{aligned} &\min f(\mathbf{x}) \\ &\text{s.t. } \mathbf{x} \in S, \end{aligned} \tag{2.1}$$

where $\mathbf{x} \in \mathbf{R}^n$ is a *decision variable*, $f(\mathbf{x})$ an *objective function* and $S \subset \mathbf{R}^n$ a *constraint set* or *feasible region* [55].

A classification of optimization problems can be done in several ways, according to different aspects of the problem itself or their solutions. In the following, classification criteria are given, which are discussed in more detail by Nocedal and Wright [38].

Discrete and continuous optimization. In *discrete optimization*, the variables used in the objective function(s) are restricted to assume only discrete values, such as the integers (e.g., a number of objects). The problem is then known as an *integer programming* problem [60]. By contrast, *continuous optimization* finds a solution from an uncountably infinite set, typically a set of vectors with real components [38].

Unconstrained and constrained optimization. Problems can be classified by the existence of constraints on the variables. In *unconstrained optimization* problems of the form

$$\begin{aligned} &\min f(\mathbf{x}) \\ &\text{s.t. } \mathbf{x} \in \mathbf{R}^n, \end{aligned} \tag{2.2}$$

often (natural) constraints are disregarded if they have no effect on the solution [38]. Thus, there is no restriction on the search space. *Constrained optimization* problems arise from models that include explicit constraints on the variables. These constraints may be simple bounds or more complex constraints such as nonlinear inequalities that represent relationships among the variables [38][58]. A problem is called *linear programming* (LP), if it involves the optimization of a linear objective function, subject to linear equality and inequality constraints [35]. LP is an important special case of optimization problems, because many practical problems can be formulated as linear programming problems and it therefore arises in a vast number of fields and applications.

Local and global optimization. *Local optimization* algorithms seek a solution that is only locally optimal. They do not always find the best of all optima, which is the *global solution.* The global optimum of the whole domain is usually difficult to identify and even more difficult to locate [38]. Although global solutions are necessary in many applications, local optimization algorithms are important as they are comparatively fast and, in addition, many global optimization algorithms proceed by solving a sequence of local optimization methods [55]. To give an example, linear programming problems fall in the category of *convex programming*, in which all local solutions are also global solutions [38].

Stochastic and deterministic optimization. In many situations, decision makers wish to solve optimization problems which depend on parameters which are unknown. *Stochastic optimization* algorithms use these quantifications of the uncertainty to produce solutions that optimize the expected performance of the model [51]. In *deterministic optimization* problems the model is fully specified. Deterministic algorithms are exact in so far as a repeated execution will yield the same output given the same input.

Single-objective and multi-objective optimization. Problems can also be classified according to the nature of the objective function. The general form (equation 2.1) states a *single-objective* problem, since only one objective function is to be minimized. However, in many real-world decision making problems, optimization techniques are applied to sets of functions, which represent multiple criteria. *Multi-objective optimization* often means to compromise conflicting goals. In this case, there will always be more than one optimal solution [59]. As this work mainly deals with multi-objective optimization, these problems will be discussed in more detail in the following section.

The applicability of optimization methods is widespread, reaching into many activities in which numerical information is processed, and much research has been done on optimization methods. To describe all techniques that were introduced in the last 50 years would go beyond the scope of this work. Focusing on problems with multiple objectives, the following section will provide an overview of multi-objective optimization algorithms.

2.3 Multi-Criteria Optimization

In many real-world problems, there is more than one criterion crucial before coming to a decision. These tasks are called *multi-criteria* or *multi-objective* optimization problems.

2.3.1 Objective Functions

Miettinen [37] defines multi-objective optimization problems as follows. The starting point is a problem of the form

$$\begin{aligned} &\text{minimize } \{f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})\} \\ &\text{s.t. } \mathbf{x} \in S, \end{aligned} \tag{2.3}$$

where $S \subset \mathbf{R}^n$ denotes a feasible region. The word *minimize* means that all objective functions f_i are minimized simultaneously. Here we have k objective functions $f_i : \mathbf{R}^n \to \mathbf{R}$. These form an *objective vector*

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \vdots \\ f_k(\mathbf{x}) \end{pmatrix}. \tag{2.4}$$

The statement above refers to a minimization problem only. If an objective function f_i has to be maximized, it is equivalent to minimize $-f_i$. The goal is to find a *decision vector*

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \tag{2.5}$$

such that every objective function attains its optimum.

2.3.2 Pareto Optimality

If the functions are not conflicting, a solution exists that is optimal with respect to every objective function. In real-world problems, this trivial case occurs very rarely. Usually, multi-criteria optimization means finding a compromise by trading off contradicting objectives against each other. In this case, there will always be more than one solution.

Pareto optimality defines the front of solutions that can be reached by trading-off the conflicting objects in an optimal manner [59]. This concept is based on *efficiency*. A point $\mathbf{x} \in S$ is efficient if its criterion vector is not dominated by the criterion vector of some other point in S. In other words, a point in a minimization problem is efficient if it is not possible to decrease one of the objectives without necessarily increasing at least one of the others [54]. Figure 2.6 shows Pareto efficient points in a two-dimensional minimization problem. The set of these points is called *Pareto set* or *Pareto frontier*. The concept of Pareto optimality is used in chapter 4, where an algorithm is presented, which computes Pareto frontiers in an interactive environment.

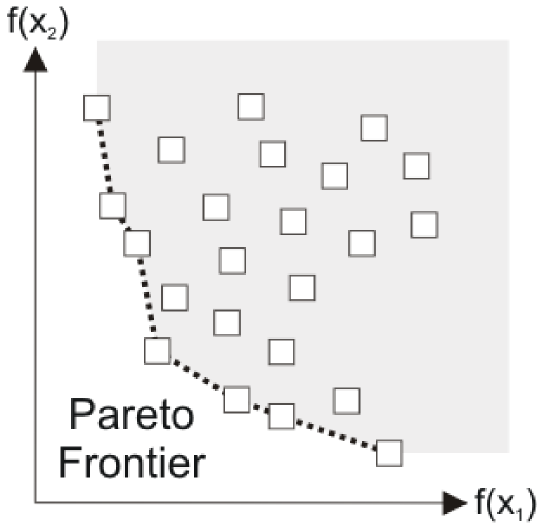


Figure 2.6: Efficient points define the Pareto frontier in a two-dimensional minimization problem.

2.3.3 Solutions to Global Optimization

A classification of algorithms to solve global multi-objective optimization problems is given by Weise [59]. Figure 2.7 emphasizes the multitude of the approaches.

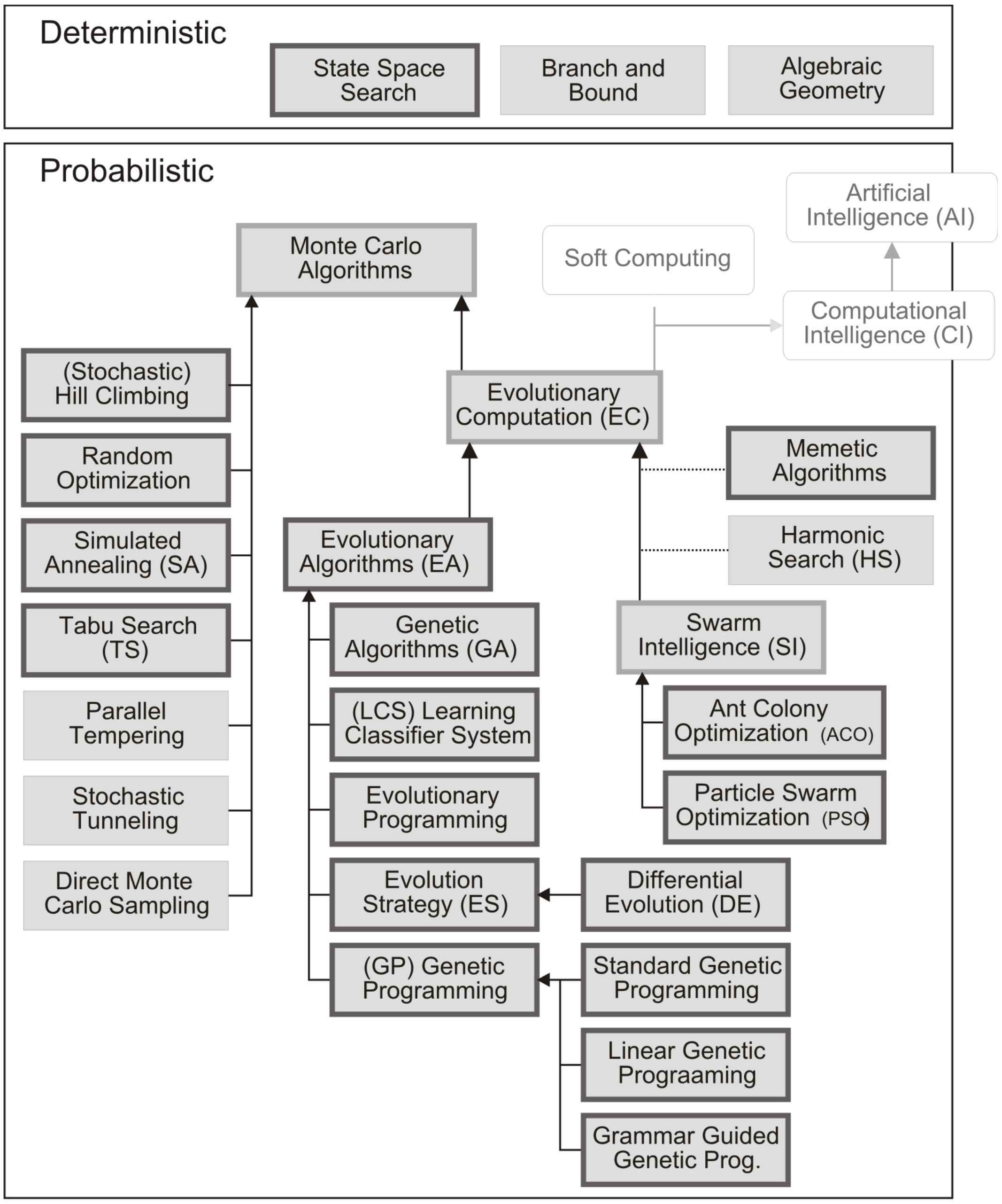

Deterministic
State Space Search
Branch and Bound
Algebraic Geometry
Probabilistic
Artificial Intelligence (AI)
Monte Carlo Algorithms
Soft Computing
Computational Intelligence (CI)
(Stochastic) Hill Climbing
Evolutionary Computation (EC)
Random Optimization
Memetic Algorithms
Simulated Annealing (SA)
Evolutionary Algorithms (EA)
Harmonic Search (HS)
Tabu Search (TS)
Genetic Algorithms (GA)
Swarm Intelligence (SI)
Parallel Tempering
(LCS) Learning Classifier System
Ant Colony Optimization (ACO)
Stochastic Tunneling
Evolutionary Programming
Particle Swarm Optimization (PSO)
Direct Monte Carlo Sampling
Evolution Strategy (ES)
Differential Evolution (DE)
(GP) Genetic Programming
Standard Genetic Programming
Linear Genetic Prograaming
Grammar Guided Genetic Prog.

Figure 2.7: Classification of global optimization algorithms. Image from Weise [59].

Weise divides optimization algorithms into two basic classes: deterministic and probabilistic algorithms. A *deterministic* algorithm is an algorithm which behaves predictably. Given a particular input, it will always produce the same output, and the algorithm will always pass through the same sequence of states [11]. On the other hand, an algorithm is *probabilistic* (also called *randomized*) if its behavior is determined not only by its input but also by values produced by a random-number generator, so random choices are made during the course of the algorithm [11]. In case of probabilistic optimization algorithms, they try to make a trade-off between guaranteed correctness of the solution and shorter runtime. This does not mean the results could be incorrect, but they may not be the global optima [59].

An important class of algorithms is *evolutionary computation*. It encompasses algorithms that are based on a set of multiple solution candidates (called population) which are iteratively refined. *Evolutionary algorithms* (EA), which form a subset of evolutionary computation, are well-suited for optimization problems and have grown in popularity over the last years. Evolutionary algorithms are randomized search algorithms using biology-inspired mechanisms like mutation, crossover, natural selection and survival of the fittest [59]. The basic idea behind evolutionary algorithms is to copy the process of Darwinian evolution in order to find solutions for hard problems. Figure 2.8 shows the basic process cycle of such an algorithm.

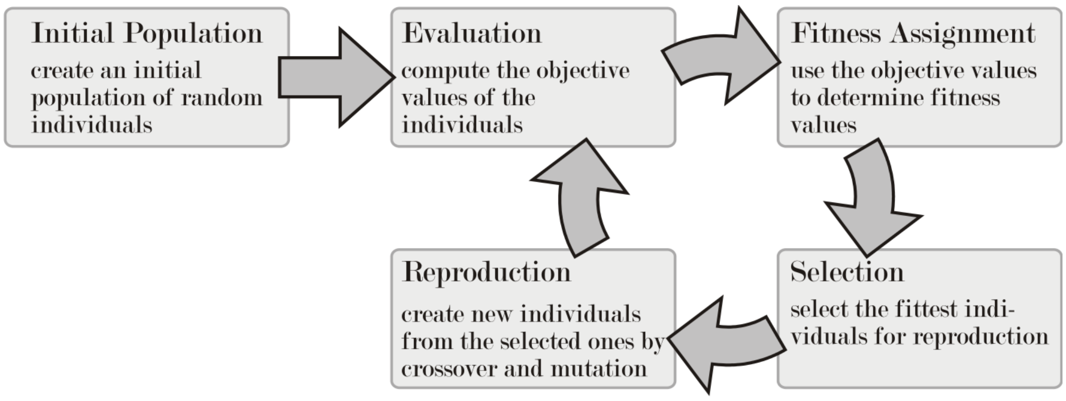

Initial Population
create an initial population of random individuals
Evaluation
compute the objective values of the individuals
Fitness Assignment
use the objective values to determine fitness values
Selection
select the fittest individuals for reproduction
Reproduction
create new individuals from the selected ones by crossover and mutation

Figure 2.8: The basic cycle of evolutionary algorithms. The process of finding solutions is based on selection and reproduction. Image from Weise [59].

2.4 Interactive Optimization

Many real-world optimization problems cannot be totally automated. In this case, user interaction is necessary for refining the optimization problem. Interactive optimization (a.k.a. *human-in-the-loop optimization* or *human-guided search*) is a field of optimization based on user feedback. There has been a considerable amount of work on automatic optimization systems. However, interactive approaches are very rare. Current optimization systems typically solve simplified formulations of real-world problems and produce solutions which sometimes are difficult for users to understand or trust. Interactive optimization allows users to explore many possible solutions in order to better understand the trade-off between possible solutions and then choose a solution based on their understanding of the domain.

Approaches using an interactive process leverage the strengths of both humans and computers. Research in interactive optimization directly addresses the questions of how people can effectively interact with optimization systems and how far the user can affect the solutions to the current problem [7]. Simple user interaction is used in *interactive evolution*. These algorithms generate solutions via biologically inspired methods and use human evaluation to produce better solutions in the next iteration step [56]. Other systems provide more interactivity by allowing users to control search parameters or to manually modify computer-generated solutions [48].

A reason to involve people in the optimization process is to leverage their abilities in areas in which humans outperform computers, such as visual perception or learning from experience. On the other hand, the major problems with interactive optimization are human fatigue in the interaction process and the fact that human evaluation is slow and subjective [56].

2.4.1 Cycle of User Interaction

In the introduction, the loop of interaction was mentioned. This loop describes the steps in the human-computer interaction. Chimani et al. [7] define the cycle of user interaction as three activities: inspection, modification and reoptimization of the current solution. These steps are illustrated in figure 2.9 and briefly discussed in the following.

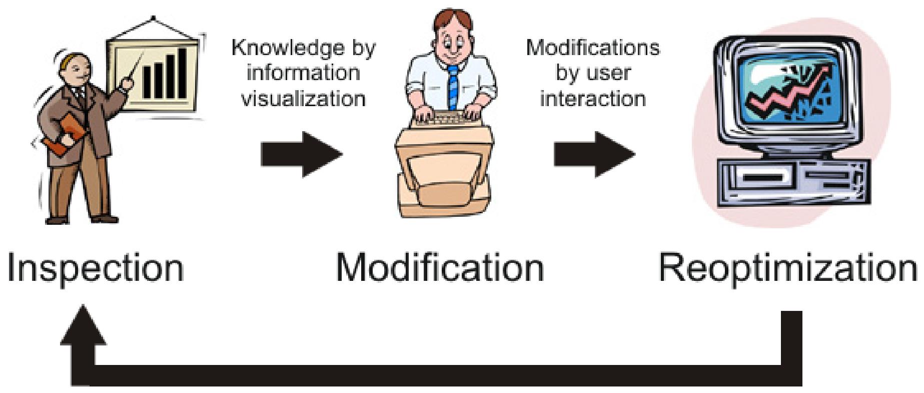

Knowledge by information visualization
Modifications by user interaction
Inspection
Modification
Reoptimization

Figure 2.9: The cycle of user interaction is defined by inspection, modification and reoptimization.

Inspection. In this step, the user browses through the solution and checks if all constraints are satisfied. Information visualization assists the user in this task by providing interaction techniques.

Modification. Once the quality of the current solution is determined and the user is not satisfied with the results, the modification state is applied. The user modifies the optimization parameters in order to get a better solution. This step is the major difference to automatic methods and it is in particular necessary if user has real-world knowledge that is not modeled in the problem specification.

Reoptimization. After the modification phase, the optimization algorithm is restarted using the modified parameters. To investigate if this leads to a better solution, the cycle of user interaction turns to the inspection step again.

2.4.2 Selected Approaches

Interactive optimization systems add new contributions to the research topics of optimization and information visualization. In this section, some approaches and applications are presented that make use of interactive optimization concepts.

Human-guided search. Human-guided simple search uses a hill-climbing algorithm [47] for finding local minima. By visualization and interaction techniques, the human user identifies promising regions of the search space for the computer to explore [2]. The idea is to escape non-optimal local minima by human guidance. Human-guided tabu search [27] follows the same approach using a more complex optimization algorithm. The HuGS platform [28], a toolkit for interactive optimization, provides visual metaphors that allow users to focus and constrain the exploration of the search space.

Interactive evolution. As mentioned above, interactive evolution algorithms are based on an iterative process. Subjective human evaluation is used to control the progress of the iteration. Thus, it is an evolutionary computation technique whose fitness function is replaced by a human user [56]. Interactive examples are provided by the University of Kent as online experiments [14]. Oliver et al. [39] deal with the problem of automatically generating the style and layout of web pages. They use a genetic algorithm, which takes into account user preferences by selecting solutions that he favors according to their graphical representation.

Interactive partitioning. Lesh et al. [32] apply interactive optimization on k-way network partitioning, which is the NP-hard problem of partitioning the nodes of a network into k disjoint subsets with the goal to minimize the number of hyperedges spanning two or more subsets. The purpose of the interaction component is to select groups of nodes on which to concentrate the computer's search.

Interactive design optimization. Aspects of design are typically ignored in optimization models because they are difficult to model with mathematics. However, they are very important in areas such as architectural design. Michalek and Papalambros [36] describe an interactive design optimization approach for architectural layouts. An interactive design CAD tool allows the designer to add, delete and modify objectives (e.g., minimize wasted space), units (e.g., bedrooms) and constraints during optimization to refine the problem definition. The designer may interact with the optimization problem in three ways: defining the problem, guiding the search and exploring the design space.

Interactive decision making. Pirkul et al. [41] describe the visual interactive decision support tool *VisOpt*, which is designed to solve P-median optimization problems (e.g., locating P facilities relative to a set of customers). The system provides a graphical representation of the problem and allows easy manipulation of solution characteristics through a point and click approach. What-if questions can be answered by clicking on visual elements, which makes it easier for the decision maker to define the optimization parameters.

2.5 Memory-Based Reasoning

Memory-based reasoning (MBR) is an approach from the field of machine learning that is used to discover valuable information from existing data. A new problem is solved by finding similar cases in the past, and reusing them in the new problem situation. For example, a salesman could infer the profit of this year's sales from the known profit of a past year having comparable conditions. MBR emphasizes a collection of cases as a large memory, and reasoning as a process of searching in this memory [1].

MBR is used in many pattern recognition and machine learning applications. The rapidly growing amount of information continually produced and distributed makes it difficult to retrieve relevant information and to draw conclusions from known data concerning future cases. MBR is a quantitative method that predicts a new case by retrieving similar cases from the past. It uses the past cases to predict the solution to the current problem [6]. MBR belongs to a class of methods named *case-based reasoning* (CBR). This class covers a range of different methods for organizing, retrieving, utilizing and indexing the knowledge retained in past cases [1].

The key question is how to find reference cases to predict the solution of the current problem and how similar one data instance is to another. This yields an application-related degree of *similarity* [23]. A simple approach in classification is the nearest neighbor (NN) method. NN classifies an instance X according to the class of the stored instance whose Euclidean distance to X is minimal [42]. In this case, the Euclidean distance defines the similarity feature. In this work, a simple MBR approach is used to estimate data values concerning predefined targets. The goal is to approximate data with respect to given input parameters. The background of this approach is to avoid costly simulations. In many real-world applications, data is collected by simulations of a system that reacts on input parameters. These simulations can be time-consuming and expensive. To get a "preview" of the data values with respect to modified input parameters, a data estimation approach is used, which is based on distances and interpolation of existing values.

2.6 The Bulk Analyzer

Bulk Analyzer is an information visualization system aiming at multi-dimensional, large data sets. It was developed at the VRVis research center [57], where this work was done. The system combines concepts and techniques commonly used in information visualization toolkits. Due to an extensible framework, new views can be added to the system by the use of a plug-in mechanism. It supports various input formats and visualization techniques like histograms, scatter plots (in 2D and 3D), parallel coordinates, categorical data, hierarchies, curves and even time series. Important concepts of the toolkit are given in the following sections.

2.6.1 Linking and Brushing

A strength of Bulk Analyzer is the *linking and brushing* concept over multiple views. The user selects subsets of the data set and the system automatically highlights these values in all views. This often provides more information than considering the visualizations independently [25]. Furthermore, the toolkit provides *composite selections.* Selections can be specified by queries using the Boolean operations AND, OR and NOT. For example, a selection can be formed by two interval selections on different attributes, combined by a logical AND. By this concept, the selected subset can be restricted or expanded in a flexible way.

2.6.2 Layers

The Bulk Analyzer supports four data layers, which determine certain subsets of the data set. By different visual representations of these layers, the *focus and context* concept is obtained. The *all entries* layer represents all data items currently imported. The *current selection* layer specifies the current focus specified by a query mentioned above. An arbitrary number of queries can be defined in Bulk Analyzer, but only one of them can be the current focus at a time. All other selections belong to the *context* layer. The *superfocus* layer represents highlighted values which currently lie beneath the mouse cursor. The visual properties of each layer (e.g., color, transparency, ...) can be set independently. Figure 2.10 illustrates an example of this, showing three different views comparing car properties. The scatter plot shows horsepower and miles per gallon (MPG), the histogram accumulates MPG values using 50 bins and the parallel coordinates view compares the attributes horsepower, MPG, acceleration and the number of cylinders. The gray values denote all values that are not part of a focus or context layer. The red values show the current selection, which is specified by an interval on the right-most axis of the parallel coordinates view. The green values are part of the context layer. Finally, the blue values belong to the superfocus layer, which is given by the mouse position in the histogram view. Changes in any layer updates the visual layer representation in all views.

2.6.3 Used Data Sets

This section shortly describes the data sets used in this work to illustrate the theoretical background. For simple exemplifications, the cars data set [12] is used. The data set was collected by Ernesto Ramos and David Donoho in 1982 and deals with automobiles. It includes eight variables for 406 different cars and it is a quite popular data set in various kinds of publications. The variables include vehicle weight, horsepower, time to accelerate from 0 to 60 mph, miles per gallon, number of cylinders, engine displacement, model year and origin (American, European, Japanese).

To demonstrate the results using real-world data, a more complex data set is used, which originates from simulation data of an engine manufacturer and is described in detail at the beginning of the case study in chapter 7. It includes input and output parameters and is therefore particularly suitable for the data estimation task in chapter 6.

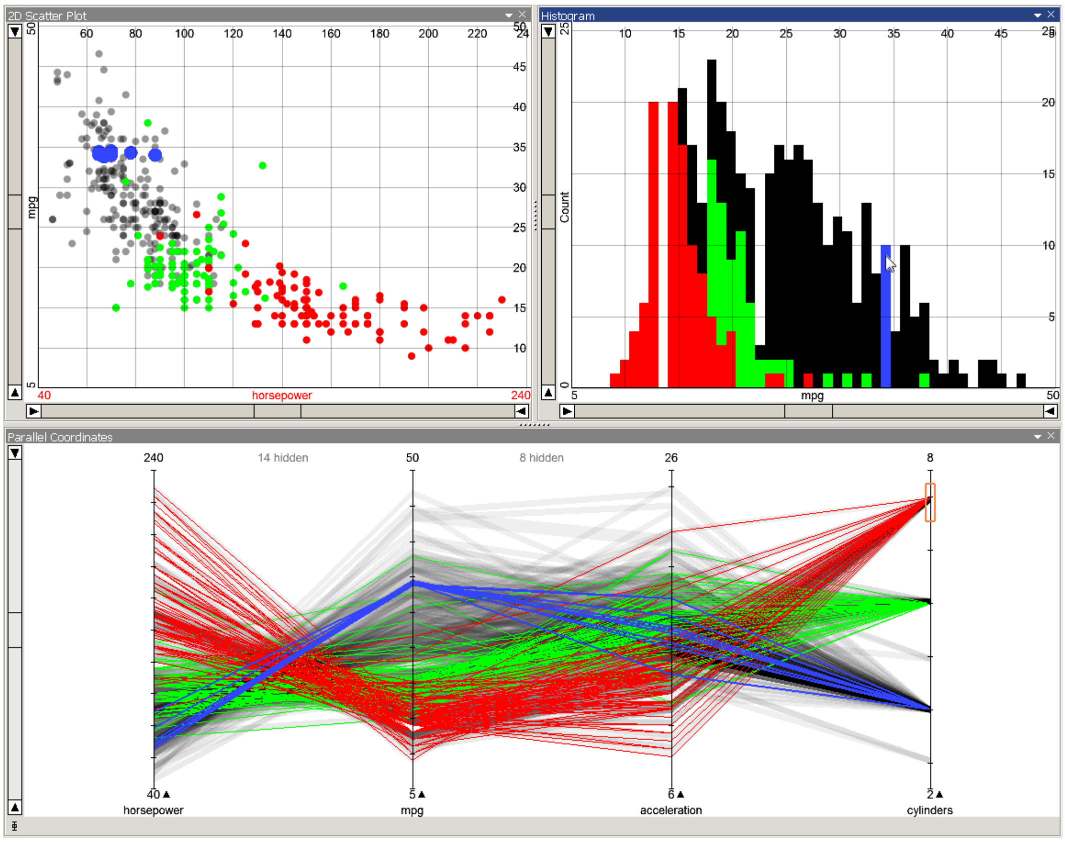


Figure 2.10: Layers in the Bulk Analyzer system. Used views: 2D scatter plot, histogram and parallel coordinates depicting car attributes. Layers: The all entries layer is gray-colored. The current selection layer (red) and the context layer (green) are defined by interval selections in the parallel coordinates view. The superfocus layer (blue) is given by the mouse position in the histogram view. Data courtesy of Donoho and Ramos [12].

Chapter 3

Multi-Objective Problems

An important goal of this work is to provide interactive means for finding solutions to problems involving more than one objective function. The objectives, which often are in conflict, may describe very different aspects of the solution. These *multi-objective* or *multi-criteria problems* come in the context of optimization. Collette and Siarry [9] define optimization problems as the search for a minimum or maximum (the optimum) of one or more functions, which define the *objectives* of the optimization problem.

Optimization is discussed in the next chapter, which uses *minimization*, *maximization* and a third objective type *goal*, which is a minimization of $|f(x) - g|$ in mathematical terms, with g denoting the goal value. From a conceptual standpoint, a goal objective means a certain value on a specific dimension, which describes the objective in a numerical way.

The concept of goal values is reused in the chapters on distance computation and data estimation. In these topics, goals are seen as points in the n-dimensional space. Although there is a conceptual difference between multi-dimensional objectives in optimization and n-dimensional goals in the subsequent chapters, the generation and manipulation of goal values is realized in a uniform manner. The large overlap of the two concepts, especially with regard to interaction, led to this combined handling of goals. Therefore, this concept is a kind of link between the different topics dealt with in this work.

3.1 Defining Objectives

In Operations Research, objectives are handled in a more sophisticated way. The parameters of multi-objective problems are differentiated between attributes, objectives and goals [8]. *Attributes* mean properties or characteristics of alternatives. *Objectives* denote specific and measurable requirements to meet, thus a concrete purpose. For instance, a problem could have the attributes effort and benefit, where the aim of minimizing effort and maximizing benefit are objectives. Intentions that are not specific enough to be measured, are usually referred to as *goals*. Goals are abstract purposes toward which an endeavor is directed, whereas objectives are concrete intentions. In the example above, a goal could be to make money. The present text uses the term *objective* for the minimization or maximization of values in a specific dimension. The term *goal* is not used as the vague intention mentioned above, but as a definition of a concrete value in a particular dimension, thus a goal value can be seen as an objective in numerical terms. For this reason, the term *objective* is used for both the actual objectives (minimization, maximization) and the goals. In the chapters on distance computation and data estimation, goals denote points in the n-dimensional space and appoint *targets* of the actual problem.

For the studies in this work, it is possible to define objective functions for arbitrary dimensions. When dealing with parallel coordinates, this task is intuitive, since each considered dimension is depicted by a particular axis. Section 3.3 shows how this ability is provided in the Bulk Analyzer system.

3.2 Assessment Ranges

To provide more control over the optimization's result, the objective concept was extended. Objectives are not only defined by their target dimension and type (minimization, maximization, goal) in the Bulk Analyzer system. Additionally, *assessment points* represent distances from the goal value, which allow for assessing the "quality" of individual cases with respect to the according objective. Each objective has three assessment points. The *target* defines a range around the goal value, where resulting values are considered desirable. The *tolerated* range encloses the target range. Values located in this range are considered acceptable but not optimal. Finally, the *undesirable* point has the highest distance from the goal value and defines a range, in which results are unacceptable. Solution points outside these ranges are ignored, if assessment ranges are activated.

These ranges are defined by the user, separately for each objective. By this concept, it is possible to split up the solution set into four quality sections. Values inside the target range of each objective are cases of best quality, tolerated values are acceptable, undesirable values are unacceptable and all other cases have worst quality. For instance, a constraint could be to highlight only those values of the solution set, which are contained within the tolerated range of all objectives.

In case of goal objectives, assessment ranges are placed symmetrically around the goal value. When dealing with maximization and minimization objectives, assessment ranges only make sense in one direction. The next section shows, how this is realized in the Bulk Analyzer system.

3.3 Integration into Bulk Analyzer

If an objective is defined for a dimension, one of three types can be set as objective function: maximization, minimization and goal. Goal objectives make it possible to specify arbitrary numerical goal values to the optimization. In the Bulk Analyzer system, the user may specify objectives in two ways. In a dialog, numerical input is supported by selecting a target dimension, choosing the objective type and entering a value in case of a goal objective. Assessment ranges are specified by entering the distance to the goal value for each range. The advantage of the dialog-based objective specification is to provide precise input in form of numerical values. Additionally, a graphical approach was implemented, which offers the ability to change the goal value and the assessment ranges interactively in the parallel coordinates view. The graphical objective representation was designed with regard to interaction. The user may modify the parameters directly in the graphical data representation, allowing an uninterrupted working process without opening dialogs. The drawback of this approach is that precise input is not always possible.

Assessment ranges are depicted by various colors. The green area represents the target, yellow indicates the tolerated range and red is assigned to undesirable. Figure 3.1 shows the visualization of three objectives. The optimization consists of a maximization on the acceleration axis, a minimization on the number of cylinders and a goal value of 138 on the horsepower axis. Note that in case of a goal objective, the assessment ranges extend in both directions. The goal value and the assessment points can be changed interactively by dragging.

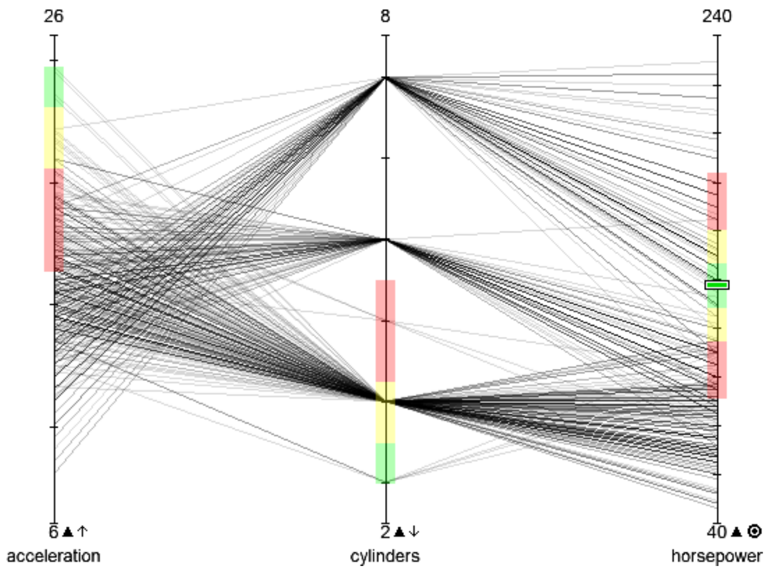


Figure 3.1: Objectives in a parallel coordinates visualization showing attributes of cars. Acceleration is maximized, the number of cylinders is minimized and horsepower is set to a goal of 138. Data courtesy of Donoho and Ramos [12].

Chapter 4

Interactive Optimization

Many problems in today's applications do not have only one solution, which fits best for all criteria. Especially in the case of multi-criteria optimization, objectives are typically contradicting each other, leading to a set of solutions. Therefore, identifying "optimal" parameterizations requires trading off objectives against each other. Among other approaches, involving the user in this process can help to attain reliable solutions fast. Pursuing this motivation, this chapter describes an interactive approach to multi-criteria optimization combining computation-based algorithms with visualization of multi-variate data.

4.1 Goals

The goal of optimization algorithms is to find the optimal solution to a specific problem. Although optimization usually is used analytically (which means it operates on functions), this chapter describes a multi-criteria optimization approach, which operates on discrete values. Especially when dealing with a large number of data elements in high dimensions, the use of computation-based techniques is essential. Many optimization algorithms exist and the informative value of the results also depends on the given problem.

A point in the solution set is optimal if it maximizes the potentiality to reach the decision maker's goals. Sometimes the effort is too high to find optimal points, so in practice a "near-optimal" solution is often satisfying. Therefore, the final solution often differs from the optimal solution in multi-criteria optimization problems [54].

The following section specifies a way to determine optimal solutions in the Pareto sense. Also, an algorithm will be presented that finds the Pareto optimal solution to a given optimization problem.

4.2 Pareto Algebra

As a starting point of solving this kind of problems, we make use of Pareto algebra, named after Vilfredo Pareto (figure 4.1), an Italian sociologist, economist and philosopher. Among contributions in other fields of science, one of his main interests was the analysis of individuals' choices. He introduced the concept of *Pareto optimality* (also named *Pareto efficiency*) [10]. The starting point is an optimization

Figure 4.1: Vilfredo Pareto (1848 - 1923). Image from Wikipedia [10].

problem with potentially conflicting objectives. With respect to Pareto optimality, a solution is said to be optimal, if it is impossible to find a solution which improves one or more objectives without worsening any other. As an example, imagine we are looking for a powerful car, which saves fuel at the same time. Figure 4.2 shows the two dimensions of the cars data set already mentioned in chapter 2. Thus, we are interested in finding those cars which have maximum horsepower and maximum MPG (miles per gallon). This states a typical optimization problem with multiple criteria. In this example, there are two functions to maximize simultaneously:

$$\max_{x \in D} f_1(x) \tag{4.1}$$

$$\max_{x \in D} f_2(x) \tag{4.2}$$

where f_1 stands for horsepower and f_2 means MPG (with D denoting the considered domain). The figure clearly indicates that there is more than one item in the Pareto optimal solution set (the red points depict the solution). In fact, most real-world problems lead to multiple solution points when dealing with Pareto optimality.

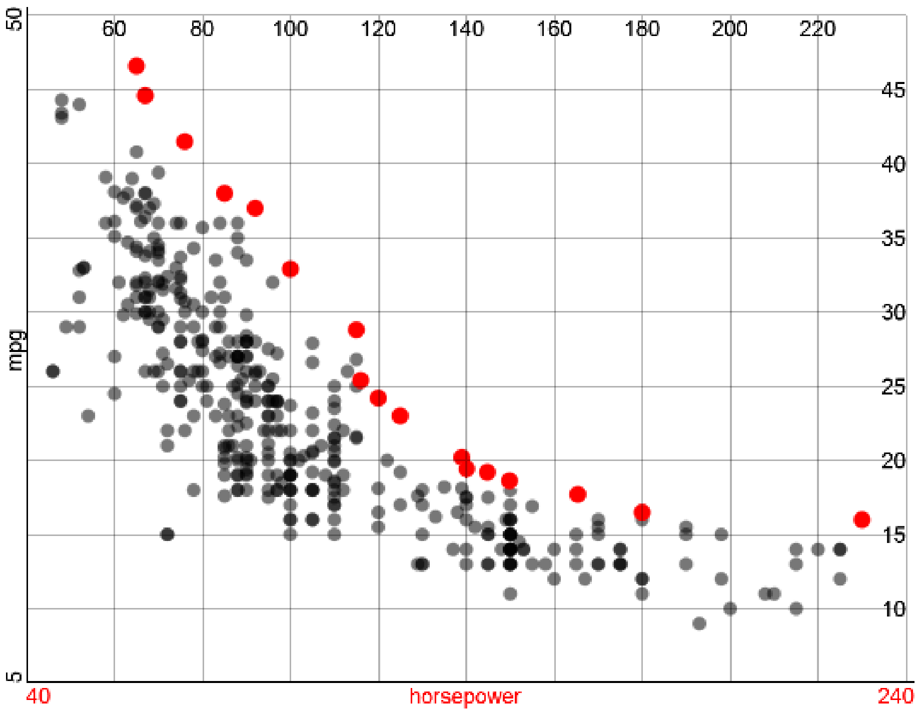


Figure 4.2: Pareto optimal solution in a scatter plot comparing cars with the goal to maximize horsepower and MPG simultaneously. Data courtesy of Donoho and Ramos [12].

A point in D is optimal if it fits best for the user's purposes, thus it maximizes the utility function. To be optimal, a point must be *efficient*, which means that it is not possible to increase one objective without decreasing at least one of the others. Inefficient solutions are not candidates for optimality.

In the following, mathematical definitions are given that are part of *Pareto algebra* [17].

Definition 4.1 *A quantity is a set Q with a partial order $\preceq$.*

We assume here that smaller values are preferred over larger ones. This does not fit to our maximization problem above, but we will see in the next section that it is easy to convert a general optimization problem to a minimization problem.

Definition 4.2 *A configuration space S is the Cartesian product $Q_1 \times Q_2 \times \ldots \times Q_n$ of a finite number of quantities.*

Definition 4.3 *A configuration $c = (c_1, c_2, \ldots, c_n)$ is an element of the configuration space $Q_1 \times Q_2 \times \ldots \times Q_n$.*

In the example above a configuration would be a point in the scatter plot, therefore a car having concrete horsepower and MPG values.

Definition 4.4 *If $c_1, c_2 \in S$, then $c_1 \preceq c_2$ if and only if for every Q_k of S, $c_1(Q_k) \preceq c_2(Q_k)$. If $c_1 \preceq c_2$, then c_1 is said to dominate c_2.*

Dominance of one configuration over another describes a partial order. If $c_1 \preceq c_2$, then c_1 is at least as good as c_2. There could be quantities, which are better, but none of them will be worse. Configurations that are not dominated by any other configuration in the configuration space are called *Pareto points*. These form a set of configurations, which are best with respect to all quantities and it is called *Pareto optimal set* or *Pareto frontier*. It usually represents the trade-off between conflicting goals. In literature, often the term "best compromises between the criteria" is used.

Furthermore, disregarding all points that are contained within the Pareto frontier and searching for the Pareto points again, obtains a second Pareto frontier. All these points are dominated by the points in the first frontier and dominate all remaining points. If this method is continued until all points are assigned to Pareto frontiers, a classification of data entries is obtained, that subdivides the data like peeling onion skins. The next section presents a deterministic algorithm, which computes all Pareto frontiers for a given data set of arbitrary dimensions.

4.3 A Deterministic Algorithm Computing All Pareto Frontiers

4.3.1 Data Preparation

The algorithm assumes that the data is stored in a two-dimensional array containing the data entries for each objective (dimension).

```
data[number of objectives][number of entries]
```

As the algorithm itself minimizes all objectives, all involved dimensions have to be transformed to minimization problems first. A maximization problem is equivalent to a minimization problem after inverting the sign of all values of the respective data column. For objectives, where a goal value g is given, each value x is transformed to

$$x' = |x - g| \tag{4.3}$$

in order to translate the objective to an equivalent minimization problem. The result of the algorithm is saved to an array that holds the Pareto frontier index for each data item. This array is initialized to -1 for each entry.

```
frontiers[number of entries]
```

4.3.2 The Algorithm

The algorithm presented here is designed to find all Pareto frontiers. It terminates when all entries are assigned to a frontier (algorithm 1).

Algorithm 1 Computes all Pareto frontiers

```
1: procedure ComputeParetoFrontiers
2:     frontier index ← 1
3:     repeat
4:         ComputeFrontier(frontier index)
5:         frontier index ← frontier index + 1
6:     until all entries are assigned to a frontier
7: end procedure
```

Algorithm 2 computes whether entries belong to the specified frontier. The procedure takes a frontier index as parameter and searches for entries that belong to this frontier by iterating through all items. The first check is if the current item's frontier index is already found (line 3). In this case, it is skipped, since we know that it surely does not belong to the current frontier. Then the algorithm analyzes all items after the currently observed one (line 5). Again, items are skipped if they belong to a different frontier. If there are two items, that potentially belong to the considered frontier, a sub procedure is called that provides the information, whether one item is dominating the other (line 7).

Algorithm 2 Computes membership to a Pareto frontier

```
 1: procedure ComputeFrontier(frontier index)
 2:     for $i = 1$ to number of entries do
 3:         if frontiers[$i$] is $-1$ then
 4:             currentFrontier ← $true$
 5:             for $j = i + 1$ to number of entries do
 6:                 if frontiers[$j$] is $-1$ then
 7:                     result ← CheckDomination($i$, $j$)
 8:                     if result is "$j$ dominates $i$" then
 9:                         {$i$ does not belong to the current frontier, so stop here}
10:                         currentFrontier ← $false$
11:                         break for-loop
12:                     end if
13:                 end if
14:             end for
15:             if currentFrontier is $true$ then
16:                 {$i$ belongs to the current frontier}
17:                 frontiers[$i$] ← frontier index
18:             end if
19:         end if
20:     end for
21: end procedure
```

The procedure *CheckDomination* (algorithm 3) compares two data items with respect to Pareto algebra as described in the last section. It takes two indices as input parameters and iterates through all objectives (line 4). If the value at the first index of the current objective is less than the value at the second index, but greater for any other objective (line 7), none of the items is dominating the other, since for domination no value must be “worse”. The same holds for the other way round (line 13). After all objectives have been considered, a domination state can be assigned (lines 19-28).

The advantage of this algorithm over other solutions is that it is deterministic. It always assigns the correct Pareto frontiers, independent of the data characteristics. In each step a predefined way to proceed is given, since it does not use any random components. Another benefit is that it computes the result frontier by frontier. This way, a Pareto frontier can be visualized while the next one is still being processed, assuming that the underlying system supports multiple threads. This concept was used in the Bulk Analyzer system and it proved beneficial not to block the user’s work while the calculation is running.

A disadvantage of the shown algorithm is its quadratic runtime, which will be analyzed later in this work. When dealing with millions of data items, this algorithm will not help very much, since the computation time will be prohibitively long. In this case, the exact solution should be disregarded in favor of an acceptable runtime, if a solution “near the optimum” is satisfactory.

Algorithm 3 Gets domination information for two entries

```
procedure CheckDomination(first index, second index)
    index1Smaller ← $false$
    index2Smaller ← $false$
    for all objectives do
        if data[objective][first index] < data[objective][second index] then
            index1Smaller ← $true$
            if index2Smaller is $true$ then
                {none of the entries is dominating the other}
                return “no domination”
            end if
        else if data[objective][first index] > data[objective][second index] then
            index2Smaller ← $true$
            if index1Smaller is $true$ then
                {none of the entries is dominating the other}
                return “no domination”
            end if
        end if
    end for
    if index1Smaller is $true$ and index2Smaller is $false$ then
        {$i$ dominates $j$ in all dimensions}
        return “$i$ dominates $j$”
    else if index1Smaller is $false$ and index2Smaller is $true$ then
        {$j$ dominates $i$ in all dimensions}
        return “$j$ dominates $i$”
    else
        {none of the entries is dominating the other}
        return “no domination”
    end if
end procedure
```

4.3.3 Runtime Studies

For the following runtime analysis we denote n as the number of entries in each data column and m as the number of used objectives.

The *ComputeFrontier* procedure has to check every entry against every entry that comes after it. The outer loop takes n calls, the inner loop $\frac{n}{2}$ in average. The function is called m times in the main procedure, thus we have a complexity of $m \cdot n \cdot \frac{n}{2}$ so far. The *CheckDomination* procedure iterates through all objectives again, therefore the total complexity is $m \cdot n \cdot \frac{n}{2} \cdot m$. The number of objectives m is constant and usually there will be much more data entries than objectives, so the crucial element is n. This leads to the notation $c \cdot n^2$ and the quadratic runtime $O(n^2)$. The quadratic runtime holds for the best case and the worst case as well here.

This is not the fastest way to compute Pareto frontiers, but it is an easy and straightforward algorithm. However, if just an approximation is needed instead of the exact solution, genetic algorithms improve performance. Coello et al. [8] give examples of how to solve multi-objective problems using evolutionary algorithms.

4.3.4 Improvements

One improvement that was made in the implementation for the system was additional data preparation. When testing the algorithm with real-world data, it became clear that multiple data points have equal values on all considered dimensions used as objectives. All these entries will have the same Pareto frontier index, so it is not necessary to compute the frontier index for each of them. The implementation marks all entries with identical values for all objectives in a previous step and starts the frontier index calculation just for one of them. The resulting frontier index is assigned to the remaining points afterward. Although the task of marking multiple data entries also causes extra costs, the performance of the algorithm increases, if there are many entries lying on the same point in the m-dimensional space, with m denoting the number of objectives.

4.4 Integration into Bulk Analyzer

This section covers the challenge of integrating the algorithm described above into the information visualization system Bulk Analyzer. The intention was to provide a framework to define objectives, to compute the Pareto frontiers and to visualize the results. A major design goal was interactivity, so objectives should be able to be modified in an interactive way.

4.4.1 Integration into the System

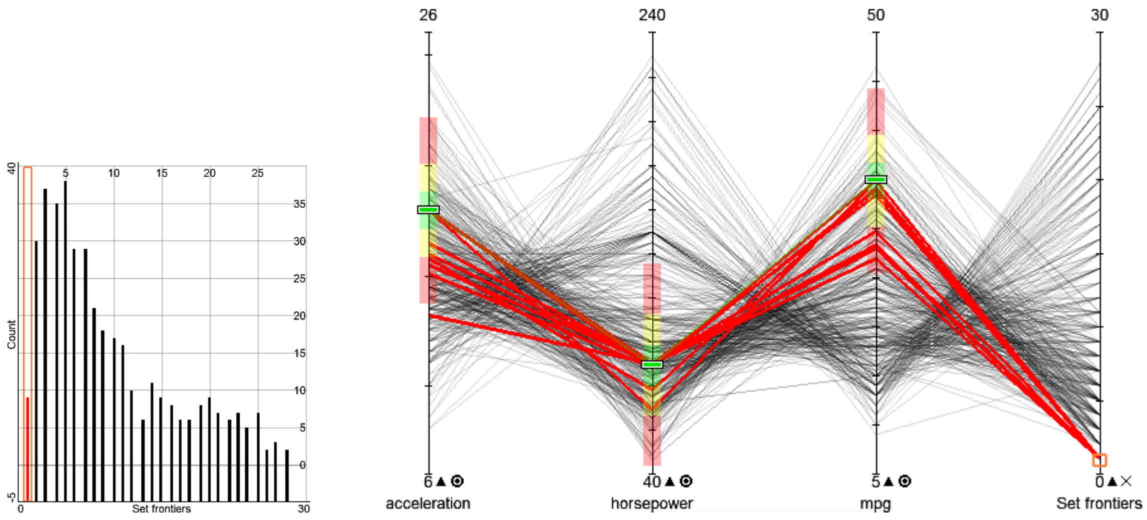
The algorithm was implemented as drafted in section 4.3. In a first version, the resulting frontier indices were just stored in an array. Thresholding was used to set the focus mask for entries, where the frontier index is smaller than or equal to a value defined by the user. This approach turned out to be too limited, as there was no possibility to reuse the computed values in any other way.

In a second version, the frontier indices are stored in a separate data dimension. This way, the values computed by the algorithm can be accessed by any component in the system. The content of this dimension depends on the optimization parameters, which are defined by objectives. If any parameter is changed, the values are optionally updated automatically or by clicking a button. We will return to this concept in chapter 5.

Each Pareto frontiers dimension can be used like any other attribute. In particular, it can be shown as a new axis using parallel coordinates and the individual Pareto frontiers can be selected directly within the view via brushing. In figure 4.3, the first frontier of an optimization is selected. The left image shows the number of entries contained within each Pareto frontier in a histogram. The first frontier is selected and highlighted in parallel coordinates, where the right-most axis shows the appropriate frontiers dimension itself.

4.4.2 Frontier Selection and Assessment Restriction

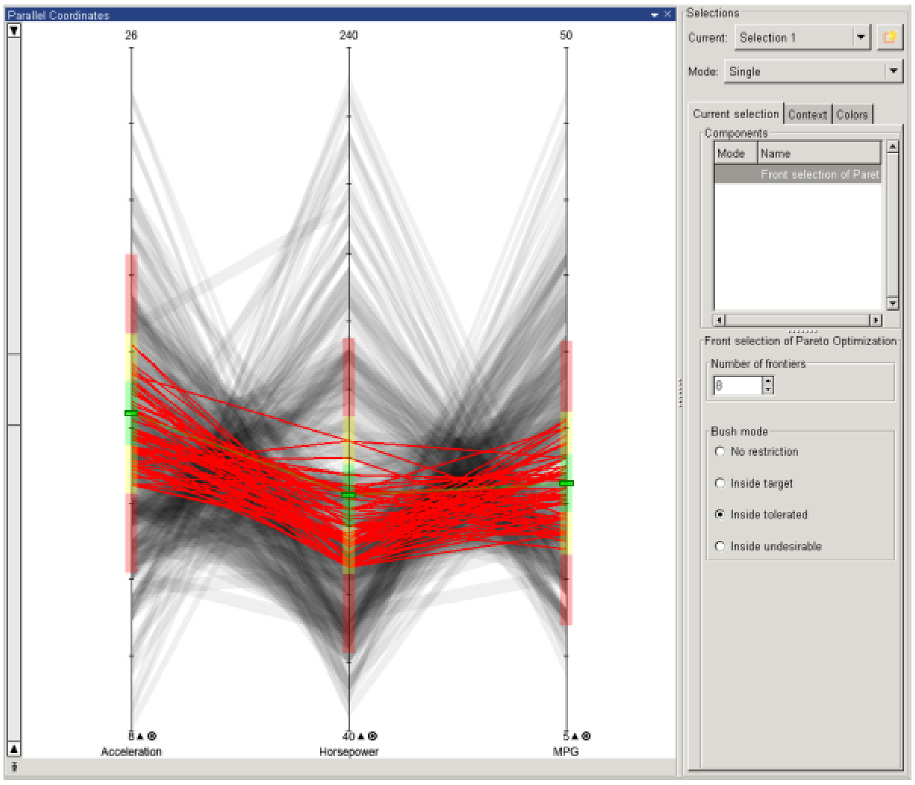
Apart from Pareto frontiers dimensions themselves, an additional selection component was added to the system. It provides the ability of changing the selected frontier (all entries of frontiers smaller than or equal to a user-defined threshold)



(a) (b)

Figure 4.3: Selection of a Pareto frontier of an optimization: (a) Histogram showing the number of data items in each Pareto frontier; (b) The optimization in parallel coordinates. The right-most axis represents the Pareto frontier dimension. The highlighted values denote the first frontier. Data courtesy of Donoho and Ramos [12].

and optionally also considers the assessment ranges. Sometimes there are entries in the set of Pareto points that lie far away from one or more goals. To make sure, that only goal-relevant values are highlighted, the Pareto frontier selection can be restricted to one of the user-defined assessment ranges. Although it is also possible to achieve this restriction by a combination of interval selections on the axes, this approach is more intuitive. For each objective, the user is able to define ranges, in which results are tolerated or not. In figure 4.4, the first eight Pareto frontiers are selected, restricted to the tolerated range (the yellow area on the objectives). Changes on the assessment ranges take effect immediately; there is no need to start a new frontier calculation, since all frontiers are already computed and stored in the Pareto frontiers dimension.



Current: Selection 1

Mode: Single

Number of frontiers: 8

Bush mode: No restriction, Inside target, [✓ Inside tolerated], Inside undesirable

Figure 4.4: Component handling frontier selection and assessment ranges. In this example, the frontier selection is restricted to the tolerated range of each objective. Data courtesy of Donoho and Ramos [12].

4.4.3 Generating Objectives by Selections

As described above, optimizations can be defined by explicitly specifying goals on data dimensions, which can be used to select values matching these objectives. Sometimes it is also helpful to go the other way round. The goal is to obtain a set of objectives (i.e., an optimization), which describe a given selection as good as possible. The purpose of this is to assess non-selected data entries with respect to properties characterizing the given selection in order to identify entries, which are similar to the selected ones. Analyzing statistical properties of the given selection determines the parameters of the objectives. For all selected values on a specific axis, the arithmetic mean

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \qquad (4.4)$$

is taken as goal value (the objective type is always *goal* here). The assessment ranges are derived from the sample standard deviation

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}. \tag{4.5}$$

With this, the optimization properties are set as follows.

$$goal = \bar{x} \tag{4.6}$$

$$target = s \tag{4.7}$$

$$tolerated = 2 \cdot s \tag{4.8}$$

$$undesirable = 3 \cdot s \tag{4.9}$$

This way, the selected data can be "described" by multiple objectives. If s is high, the selected values span a wide range on the respective dimension. In this case, the assessment distances will also be high. If s is low, the assessment distances will have lower values. This ensures a more evenly distribution of tolerated and non-tolerated values among the considered dimensions. The user may choose the dimensions, for which objectives are created, in a dialog.

4.4.4 Coloring by Frontiers

To integrate the Pareto frontier information into parallel coordinates, the view was extended. Although it is possible to visualize the membership of data items to Pareto frontiers by adding the frontiers dimension as an additional axis, coloring of polylines was implemented to perceive the information directly in the view. The goal was to map one extra dimension to the view without adding another axis. The colors are set according to the values in the desired dimension. This requires a mapping from data values to RGB color values, which is done by predefined transfer functions. Figure 4.5 shows an example of a transfer function mapping small values to blue, moderate values to green and high values to red.

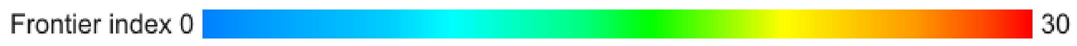


Figure 4.5: Sample transfer function. Low values are mapped to blue, high values to red.

For the purpose of this chapter, the Pareto frontiers dimension is mapped to color values. A benefit of colored polylines is the improved perception of connected lines. In figure 4.6, bluish values determine low Pareto frontier indices, which means, that these items are "good" with respect to the underlying optimization. This example illustrates the advantage of storing the optimization result to an independent dimension. Since any dimension can be chosen as color attribute, no special handling is needed to map the Pareto frontier indices to colors.

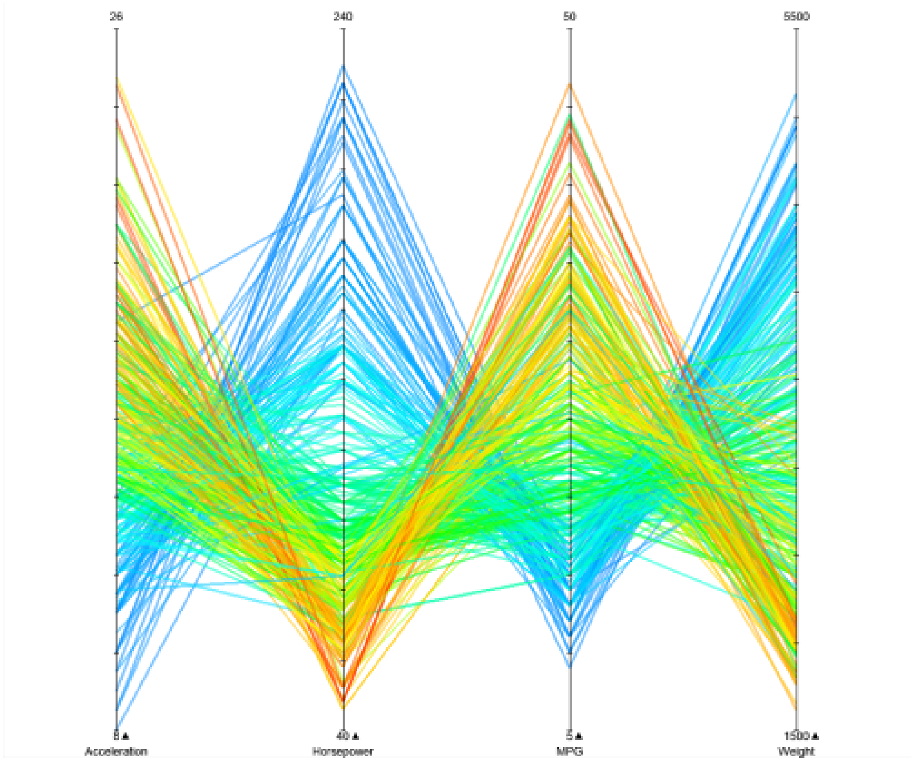


Figure 4.6: Pareto frontier dimension used as color information. This optimization minimizes acceleration, maximizes horsepower and minimizes MPG. Data courtesy of Donoho and Ramos [12].

4.5 Interaction Loop

Interaction was a key aspect of the approach as examined in this thesis. The loop of interaction contains user input, Pareto frontier computation and visualization. Every time, the user modifies one or more objectives, the algorithm is recomputing all Pareto frontiers either automatically or by clicking a button. This computation is executed in a separate thread, in order not to delay the user's work. The result is computed frontier-by-frontier; this ensures that each frontier can be selected once it is determined. As mentioned, the objectives can be manipulated directly within a parallel coordinates visualization. By dragging, goal values can be modified as well as the assessment distances.

The algorithm described in this chapter is used to compute the Pareto frontiers. The result of this computation (one Pareto frontier index for each data item) is stored into a separate data dimension. This way, the Pareto frontier indices can be used like any other attribute.

The visualization part provides the possibility to handle the Pareto frontiers as selection. The user specifies the number of frontiers to be highlighted and the visualization system restricts the selection. This selection can be treated like any other selection, particularly it can be combined with other selections. The visualization front-end also allows the user to set the assessment ranges. If the selection should be restricted to a predefined area around the objectives, the values *target*, *tolerated* or *undesirable* can be chosen (see chapter 3 for further informations on assessment ranges). This process is shown in figure 4.7.

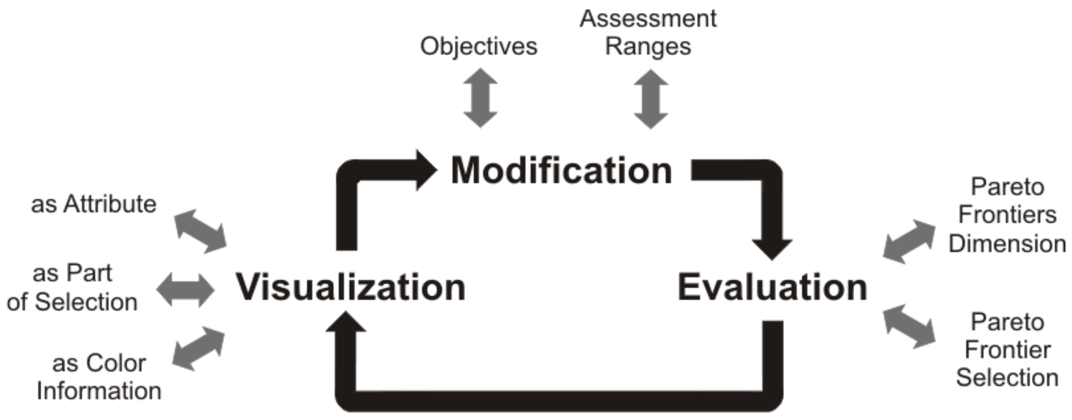


Figure 4.7: Loop of interaction. The interaction cycle consists of modification of objectives and assessment ranges (user input), evaluation (computation of Pareto frontiers) and the visualization of the results. These results can be shown as additional attribute (e.g., as an axis in parallel coordinates) or as a selection of entries. Furthermore, the results can be reused as color information.

Chapter 5

Distance Computation

The previous chapter dealt with Pareto optimization, which is a measure of fitness with respect to predefined goals. To discuss another approach based on multi-dimensional goals, distance computation techniques are presented, which were implemented in the Bulk Analyzer system.

5.1 Goals

The Pareto frontier calculation algorithm shown in chapter 4 assigns each data value to its appropriate Pareto frontier. The drawback of Pareto optimization is that entries can be possibly useless, although they belong to the first Pareto frontier, as they are "good" for a single goal, but not for all. Another way of determining the fitness of data is to compute the distance from each entry to a predefined multi-dimensional goal. This way, there are no classes of entries anymore, as when dealing with Pareto frontiers. Instead, a numerical distance value is assigned to each item, which determines the closeness to the target.

In the following sections, several ways of calculating these distance values are presented, as well as the integration into the Bulk Analyzer system.

5.2 Context to Optimization

Although optimization and distance computation are different approaches, this thesis points out analogies concerning goal definition and the characteristic of the solution. The Pareto optimization approach considers the "real" objectives minimization and maximization, whereas only numerical goal values are used in this chapter. The definition of these goals is done in the same way as described in chapter 3.

The result of the computation is one distance value for each data entry. Again, this result is stored in a separate data dimension and can be used as an additional attribute. Both the calculation of Pareto frontiers and distance values considered in this chapter describe the relationship from one data item to others as well as the closeness to the predefined multi-dimensional goal.

5.3 Scaling Data

Different data attributes mean different value ranges. As an example, the cars data set includes the attributes weight and cylinders. Weight ranges from 1500 to 5500 lbs, whereas cylinder values lie between 3 and 8. This discrepancy must be taken into account before calculating an overall distance value. Thus, all considered dimensions must be mapped to a common data range first in order to get comparable values. The following sections describe various scaling methods that were examined in the context of this thesis. Results and differences in quality between the various methods will be shown in a case study in chapter 7. Note that also the goal value has to be scaled in order to fit to the underlying data after the transformation.

5.3.1 Minimum/Maximum Scaling

Each dimension is scaled linearly to the range $[-1, 1]$. Let x_{min} and x_{max} denote the minimum and maximum, respectively of a given dimension x, then the coefficients for the scaling equation are computed by

$$k = \frac{2}{x_{max} - x_{min}} \text{ and} \tag{5.1}$$

$$d = -1 - k \cdot x_{min}. \tag{5.2}$$

These coefficients are used to project each entry to its scaled equivalent by the linear function

$$x_i' = k \cdot x_i + d. \tag{5.3}$$

This transformation projects all points of the dimension x to the interval $[-1, 1]$. However, this approach is very sensitive to outliers. If there are values that are numerically distant from the rest of the data, the range of relevant data will be very small, which could lead to numerical problems in successive calculations.

5.3.2 Statistical Normalization

This approach scales the data by the statistical values *mean* and *standard deviation.* These values are calculated by

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \text{ and} \tag{5.4}$$

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2.} \tag{5.5}$$

The aim of normalization is to obtain the distribution of values with mean ≈ 0 and variance ≈ 1 for each attribute. To normalize the data, the following equation is used.

$$x_i' = \frac{x_i - \bar{x}}{s} \tag{5.6}$$

5.3.3 Linear Interpolation

Another idea is to interpolate the data with respect to assessment points. The user may define distances, which specify the scale the values will be interpolated to. To illustrate this concept, figure 5.1 shows the interpolation of a value x to the user-defined scale. The interpolation is defined by the goal and the assessment points of the considered dimension (horizontal axis) and the user-defined distances d_{Target}, $d_{Tolerated}$ and $d_{Undesirable}$ (vertical axis). These values form the lines in the illustration (symmetric about the goal value), along which the value x is projected onto the scaled value x'.

Since the user-defined distances are equal for all dimensions to be scaled, the projected values will have a common data range.

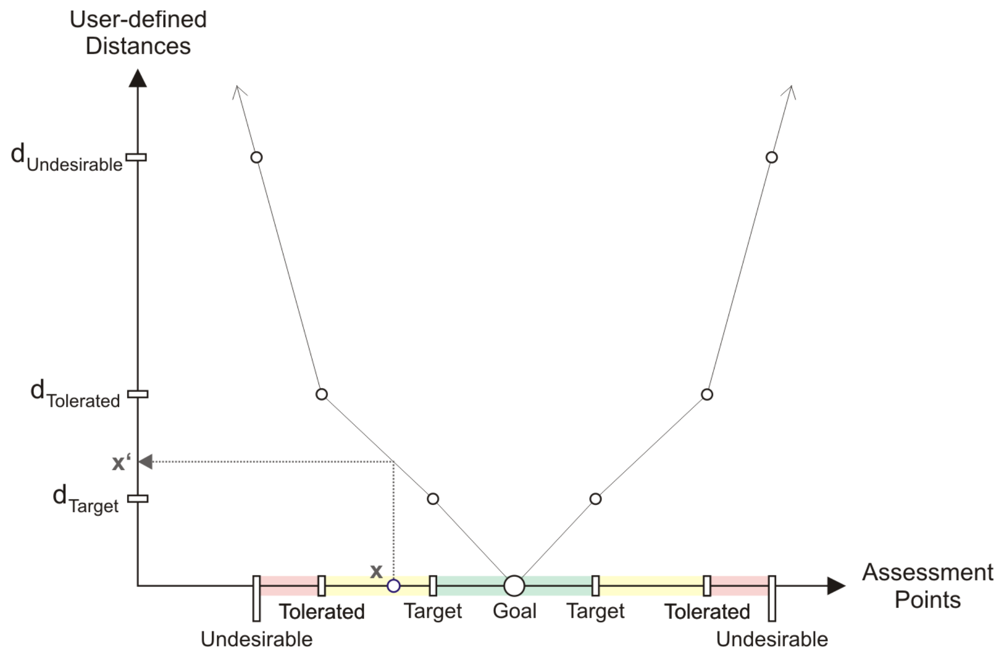


Figure 5.1: Scaling by linear interpolation. The vertical axis shows user-defined distances, which define the interpolation properties with respect to the assessment points of the goal. This interpolation is used to project each value x onto the scaled value x'.

The advantage of this approach is that the user is able to influence the results. Defining different linear interpolation properties (i.e., different slopes of the lines in the illustration) makes it possible to provide separate projections for the values in each assessment range.

5.3.4 Division by Average / Division by Goal

The last scaling approach is division. Division by average normalizes all considered dimensions to a mean of 1.

$$x_i{}' = \frac{x_i}{\frac{1}{n}\sum_{k=1}^{n} x_k} \tag{5.7}$$

Division by goal scales values around the goal to ≈ 1.

5.4 Distance Computation

Once the data is scaled to a common value range, distances can be computed. As mentioned before, the goal value also has to be transformed, when the data is projected by scaling, as denoted by *goal′*. The distance computation is done by a subtraction from the goal value.

$$d_i = |x_i{}' - goal'|^{exponent} \tag{5.8}$$

Before aggregation, an optional non-linear exponent can be defined. If this exponent is > 1, small distances (< 1) are decreased, whereas large distances (> 1) are increased. The default exponent is 1. The effect of exponents > 1 is that the distance computation gets more sensitive to outliers, whereas exponents < 1 decrease this sensitivity.

5.5 Aggregation

In order to obtain a single distance value for each data record, a final step is required where the distance values of the individual dimensions are aggregated. Several methods were implemented in the context of this thesis. In chapter 7, a case study is presented comparing the methods.

Maximum distance. Defines the maximum distance over all objectives as a result value. Denoting obj as the set of all objectives, the solution distance d_i for each data entry i is

$$d_i = \max_{obj} d_{obj,i}. \tag{5.9}$$

Average distance. Calculates an average distance value over all used objectives. In the following, m denotes the number of objectives.

$$d_i = \frac{1}{m} \sum_{obj=1}^{m} d_{obj,i} \tag{5.10}$$

Summed distance. Sums all distances belonging to one data item.

$$d_i = \sum_{obj=1}^{m} d_{obj,i} \tag{5.11}$$

Weighted summed distance. Specifying a weight to each objective allows for assigning different priorities to the dimensions. Usually, the sum of all weights is 1 to avoid normalization problems. Denoting the weight of the objective obj with w_{obj}, the distance value of the i^{th} data entry is retrieved by

$$d_i = \sum_{obj=1}^{m} (d_{obj,i} \cdot w_{obj}). \tag{5.12}$$

5.6 Integration into Bulk Analyzer

As in the case of the optimization task in chapter 4, the result (i.e., the distance per entry) is stored in an own data dimension. Thereby, the distance values can be accessed by the system like any other data attribute. In the parallel coordinates view, for instance, a new axis can be added to show and select distance values. In figure 5.2, the eight lowest entries on a distance dimension (the right-most axis) are selected to improve the perceptibility of data items having low distance. Based on two goal values, the “closest” values are computed. Statistical normalization was chosen as scaling method, the resulting distances were accumulated by average.

Distances can also be used as a color map. Figure 5.4 shows a 2D scatter plot depicting the attributes horsepower and MPG. The distance dimension is mapped to color by the use of the transfer function in figure 5.3. Points close to the target are dark blue, indicating low distance. The higher the distance to the target, the more changes the data points’ color to green, yellow and finally red.

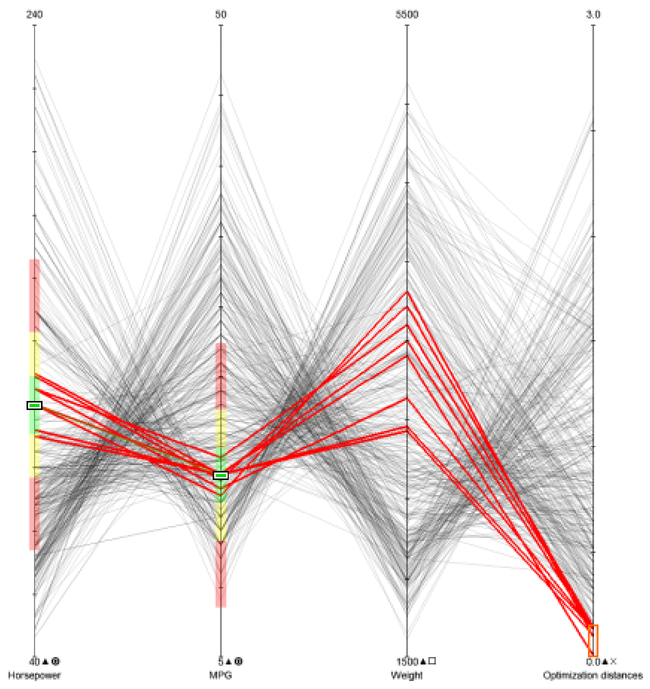


Figure 5.2: Selection on a distance dimension with goal values 120 on horsepower and 18 on MPG in a parallel coordinates visualization. The eight lowest distance values are selected. Data courtesy of Donoho and Ramos [12].

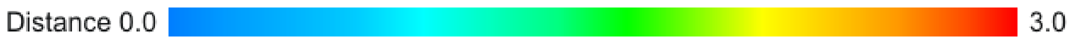


Figure 5.3: Transfer function used in figure 5.4. Low values are mapped to blue, high values to red.

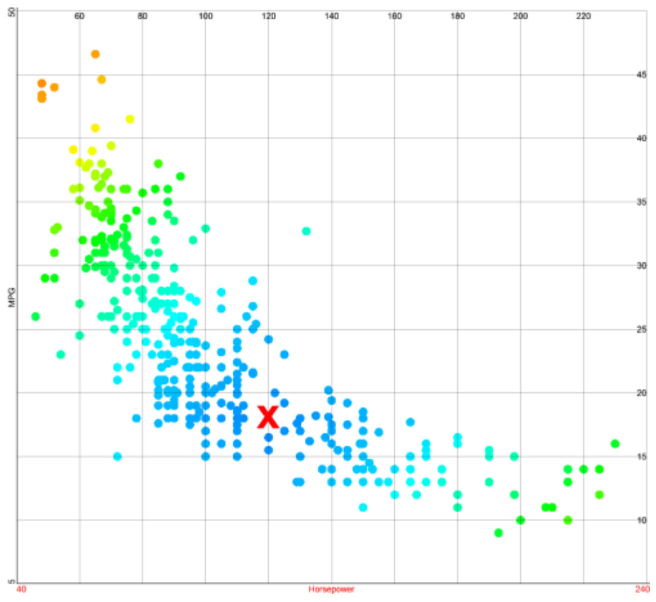
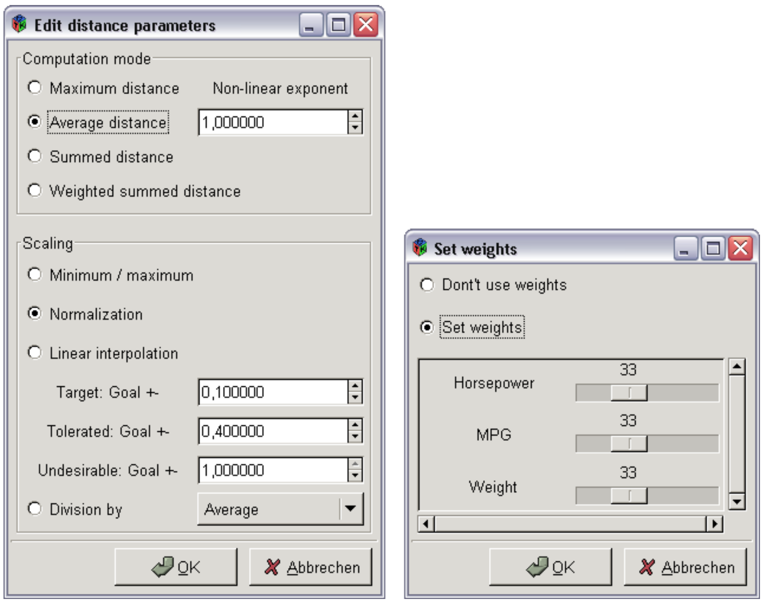


Figure 5.4: 2D scatter plot of the attributes horsepower and MPG using distances as color information. The target (120 on horsepower, 18 on MPG) is depicted as a red cross. Data courtesy of Donoho and Ramos [12].

Figure 5.4 demonstrates the correctness of the distance computation. Normally, attributes are colored that are not part of the distance definition to obtain additional information from the computed distances.

The input parameters of the distance computation are represented by points in the n-dimensional space. In this thesis, these points are referred to as goals. Goals are defined as described in chapter 3. This also means that user interaction is similar to interactive optimization in chapter 4, although the semantics differs, since we do not have objectives in distance computation, but n-dimensional goal points.

Once a distance dimension has been created using a particular set of goals, the properties of the distance computation can be changed via an input dialog (figure 5.5 left). Another dialog is used to define objective weights (figure 5.5 right), which are used by one aggregation mode as explained in section 5.5. All involved attributes are listed in a window. The weight of each attribute can be set by a slider control. Whenever a weight is changed, the other weights are adjusted to assure an overall weight of 100% by evenly adding, respectively subtracting percentage points.


Edit distance parameters

Computation mode
- Maximum distance
- [✓ Average distance] — Non-linear exponent: 1,000000
- Summed distance
- Weighted summed distance

Scaling
- Minimum / maximum
- [✓ Normalization]
- Linear interpolation
  - Target: Goal +- 0,100000
  - Tolerated: Goal +- 0,400000
  - Undesirable: Goal +- 1,000000
- Division by: Average

OK | Abbrechen

Set weights
- Don't use weights
- [✓ Set weights]

Horsepower	33
MPG	33
Weight	33

OK | Abbrechen

Figure 5.5: Dialogs handling distance computation parameters (left) and objective weights (right).

Chapter 6

Data Estimation

The previous chapters mainly dealt with approaches to assess data entries with respect to multiple objectives. The goal of the work as described in this chapter is to compute and visualize estimated data values.

6.1 Goals

Normally, data is collected by measurements or generated by simulations. Numerical simulations are typically very complex and time-consuming, so it might be helpful to provide approximations of (not yet simulated) parameterizations in order to get a quick “simulation preview”. Although it is unavoidable to start a new simulation in order to get exact results, an approximation can help decision-makers to identify beneficial parameterizations.

In this work, approximation is used to estimate simulation results at positions of the parameter space, which have not yet been simulated. Based on statistical methods and distances (see chapter 5), values are determined, which depend on predefined input parameters. This describes a *predictive modeling problem* [20], which is a common task in data mining. The goal of these problems is to predict the value of one or more variables from known values (of potentially other variables).

6.2 Estimation Computation

Approximation is used in this work to compute a data point that represents an estimation of a simulation based on predefined targets. The idea behind this approach is based on the concept of similarity. Results are determined by analogous situations in the past (e.g., from preceding simulations). In AI research this task is referred to as *memory-based reasoning* (MBR). The intention is to use a large memory of examples as a reasoning base. Further details on MBR are given by Stanfill and Waltz [53].

6.2.1 Estimation Targets

The estimation targets, for which the values are approximated, are set via goals, as it was done in the previous chapters. These goals are n-dimensional points, defining input values used for the estimation. In the following, a way is discussed how to estimate values for all dimensions that fit best according to these targets. Due to the conceptual overlap, the interactive definition of goals as described in chapter 3 is reused for data estimation.

6.2.2 Weights

The implemented estimation method uses distances as computation base. The precalculated distance dimension contains the distance of each data entry to the user-defined estimation target. These distances are transformed to weights and stored in a separate data dimension. The weights are used when summing up the existing values in order to obtain the desired result. Data values close to the target (which have small distance values) thus preponderate over values that are numerically distant from the defined parameters.

The first step is to calculate the weight per entry by transforming high distances to small weights and vice versa. This is done by

$$w_i = \frac{1}{d_i + 1}. \tag{6.1}$$

By this, a weight of 1 is assigned to values having distance 0, whereas weights are getting smaller with increasing distance. In order to get useful results, this values have to be normalized. By dividing each weight by the sum of all weights, an overall sum of 1 is achieved.

$$w_i{}' = \frac{w_i}{\sum_{k=1}^{n} w_k} \tag{6.2}$$

6.2.3 Estimated Values

The next step is to compute a value for each data dimension, which represents the result of the estimation. This is done by computing the weighted sum for each dimension m.

$$I_m = \sum_{i=1}^{n} (x_{m,i} \cdot w_i{}') \tag{6.3}$$

This yields a vector I that represents one estimated result per dimension. This result value is highlighted in a parallel coordinates visualization by an orange polyline, as figure 6.1 demonstrates. In this example, a two-dimensional estimation target is set (Input 1 and Input 2 in the left image).

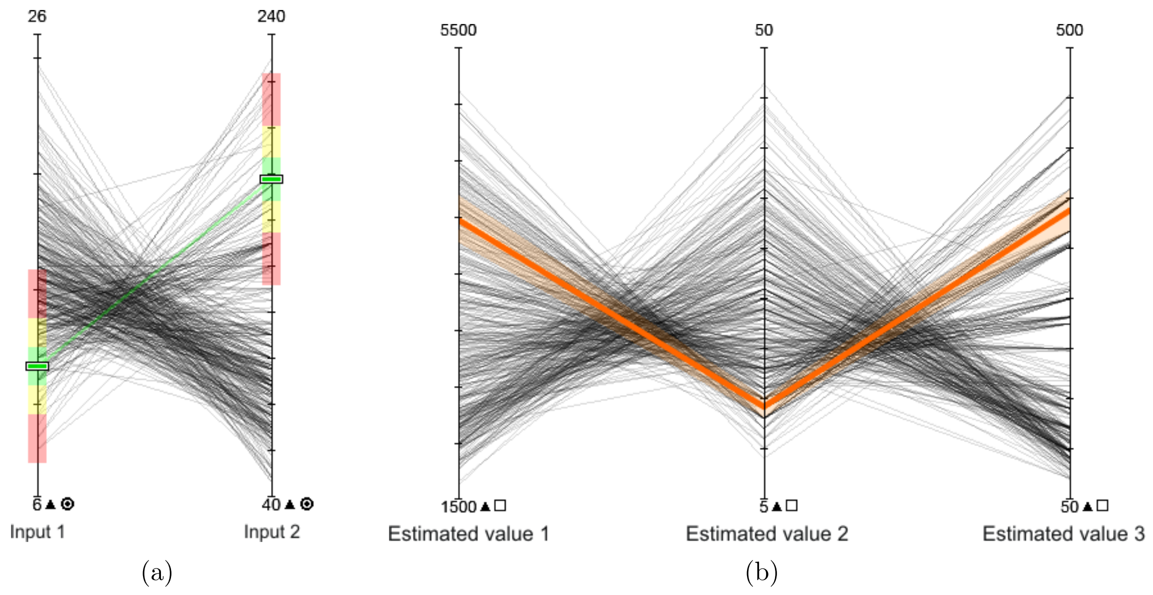
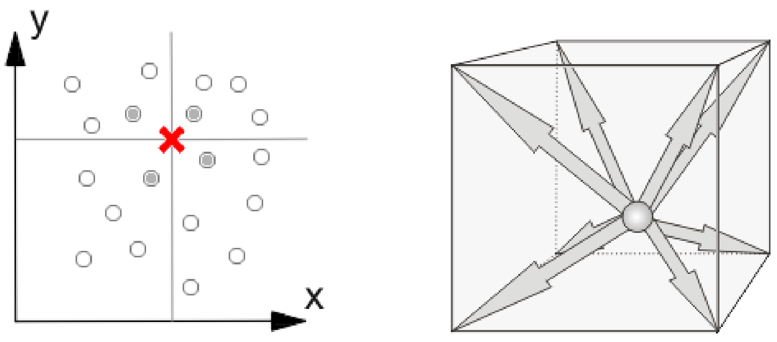


Figure 6.1: Data estimation in parallel coordinates: (a) Two estimation targets define the input parameters of the estimation; (b) The estimated values are depicted as orange polyline.

6.2.4 Global versus Local Approximation

The computation of weights used for the estimation can be done in several ways. In a first approach, the estimated values were determined by taking all values into account, which means that for all values a weight > 0 was calculated, since all values have a distance ≥ 0. When the results were analyzed, it became clear that values, which are very far from the predefined estimation target, influenced the resulting values too much. The effect of this was that the estimated values only changed slightly when the target was modified, because values having high distance interfered in the computation.

Thus it turned out that the estimation approach is not a global problem, since the relevant values are located around the estimation target. Considering this in a second try, the calculation of the weights was transformed to a local problem. The goal was to assign weights only to relevant points around the target, whereas all other weights are set to zero. This was done by selecting the closest value in each direction, which leads to 2^m values that will be taken into account, when m denotes the number of goals, which represent the estimation target. Figure 6.2 shows the selection of relevant points in two and three dimensions.



(a) (b)

Figure 6.2: Choosing the relevant points for the approximation. In the two-dimensional space (a), the four closest values are taken into account; in three dimensions (b), the eight closest ones (one per direction).

However, the results were still not satisfying, since this approach of choosing relevant values is too restrictive. If the chosen values are outliers, the approximation will also not represent the underlying data correctly.

In a final attempt, a compromise was made between the previous approaches. For the computation of weights, all values are taken into account that are located within a certain interval. This interval is defined by the closest value and additionally an amount of five percent of the dimension's range in each direction, as figure 6.3 demonstrates in parallel coordinates. The upper bound of the interval is defined by the first crossing point upwards plus an amount of five percent of the axis' range; the same procedure is done downwards for the lower bound. All values in this interval are considered for the estimation, which means that weights are computed for these points only. The weights for all other points are set to zero.

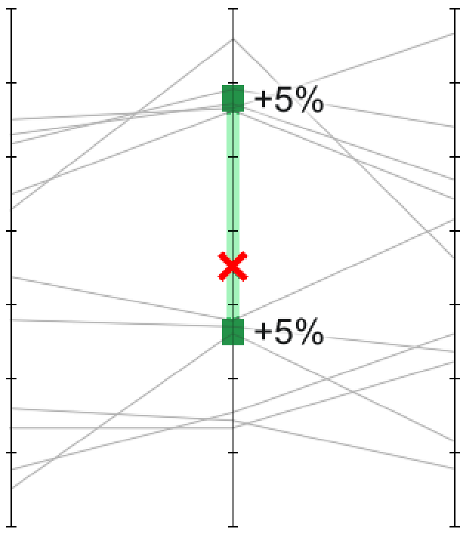


Figure 6.3: Choosing the relevant point range for the computation of weights. In each dimension, an interval is defined that includes the closest points up- and downwards plus an amount of five percent of the data range in each direction. The goal value is depicted by a red cross.

6.3 Estimation Confidence

Additionally to the estimated values, the confidence of the estimation is assessed by the weighted average distance between the estimated and the actual data values. One confidence value is computed per dimension, which indicates the significance of the estimated value in the respective dimension. It is calculated by the following equation with m denoting the m^{th} dimension.

$$S_m = \frac{1}{n} \sum_{i=1}^{n} \left(|x_{m,i} - I_m| \cdot {w_i}' \right) \tag{6.4}$$

According to the estimated values, this yields a vector containing one confidence value per dimension. Low values indicate a high confidence in the result and vice versa. Low values mean that the majority of high weighted data values is concentrated around the estimated value, which indicates a robust result.

6.4 Integration into Bulk Analyzer

The data estimation was designed to work on top of a distance computation. Therefore, the parameters used as estimation target are defined via n-dimensional goals. Chapter 5 describes a way to generate distances based on these goal values. Once the distance dimension is computed, another dimension is built on this basis, containing the weights used for the computation of the estimated values. Inside the parallel coordinates visualization, an estimation result may be added as an attribute. In the Bulk Analyzer system, attributes are used to set (visual) properties of a view. For example, other attributes of the parallel coordinates view are the used axes or the dimension used for coloring. An arbitrary number of estimations can be added to the view, and a color is assigned to each of them (figure 6.4).

The view automatically computes the estimated value and the confidence for each dimension using the precalculated weights. The estimated values are depicted by a bold polygonal line, the confidence by a semitransparent area as additional visual elements, as figure 6.5 illustrates. If the semitransparent band is thin in one dimension, the level of confidence in the approximation is high and vice versa.

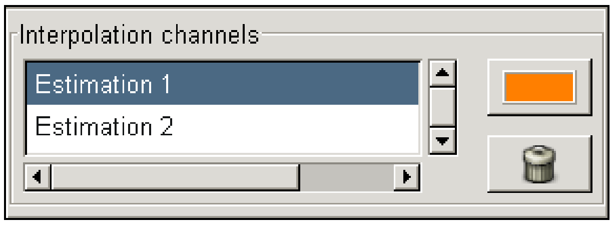


Figure 6.4: Data estimation management window. Estimations are added as attribute; the user interface provides the possibility to change the assigned colors and to remove them.

Each time the user modifies the estimation target, the underlying distance dimension changes its contents as well as the weights assigned to it. This also effects all estimations generated from this weight dimension. This approach forms an intuitive way of getting additional information by the analysis of existing data. In the following chapter, data estimation is illustrated in detail in a case study.

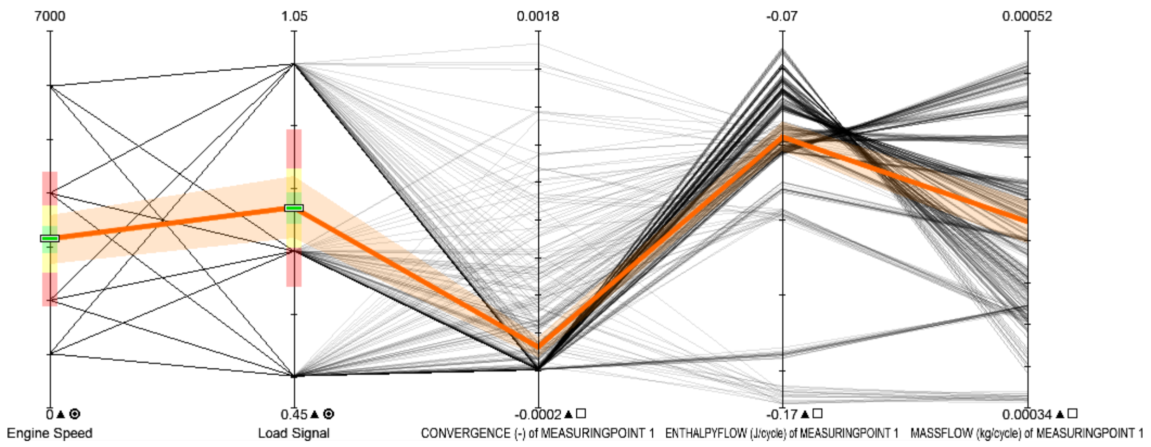


Figure 6.5: Data estimation using real-world simulation data of an engine manufacturer. The target is defined by goals on two input parameters of the simulation (3,150 on engine speed and 0.77 on load signal). The orange polyline shows the approximated simulation result, bordered by the semitransparent confidence information. Data courtesy of AVL [19].

Chapter 7

Case Study

This chapter illustrates the usefulness of the techniques as described in this thesis by an interactive visual analysis of simulation data of a car engine.

7.1 Used Data Set

The investigated data set is provided by AVL, a company offering power train engineering and testing solutions in both the automotive and nonautomotive industries. The data consists of multiple simulation runs, which have been generated by an engine simulation program named AVL BOOST. This program computes one-dimensional gas dynamics for intake- and exhaust-systems in order to model one or more complete engine cycles. The simulations' results can be used to predict engine performance, acoustics and effectiveness [19].

BOOST offers a graphical editor to build a model using engine elements like cylinders, junctions and restrictors, which are connected by pipes (figure 7.1). From an abstract point of view, each simulation run is specified by multiple input parameters and it generates a typically large number of — mostly time-varying — results as output. In the BOOST engine model, such results as pressure, temperature, air/fuel ratio, and many more, are computed at predefined measuring points in the simulation model. The distinction between input and output parameters is also reflected in the data set.

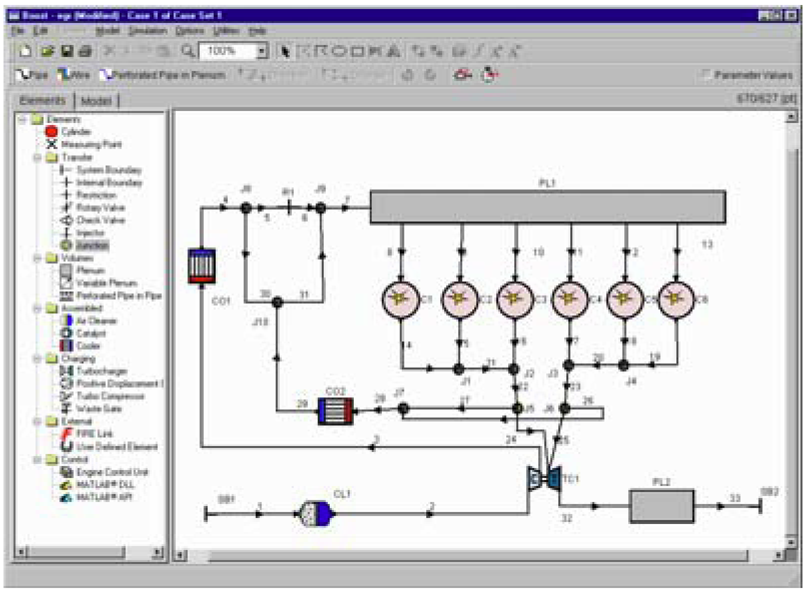


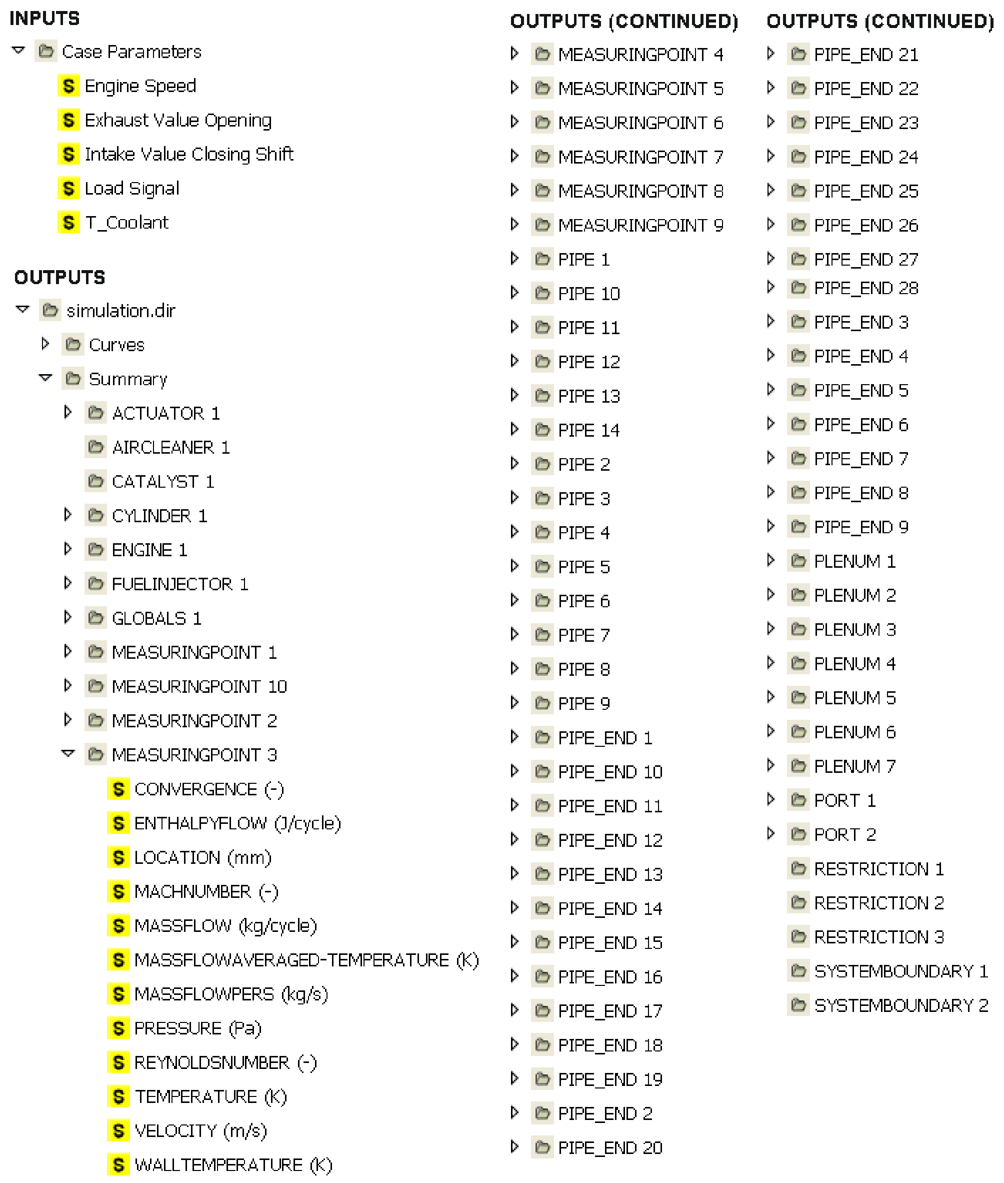
Figure 7.1: Graphical editor for the engine simulation tool AVL BOOST. Iconic elements define the engine model. Image from AVL [18].

The data set used in this chapter consists of 504 entries (i.e., simulation runs) with 1,038 dimensions. 569 attributes are defined as 1D curves. The techniques introduced in this thesis can not directly be applied to them and 1D curve attributes are therefore not used in this case study. Five dimensions are input parameters of the simulation model. The remaining 464 attributes represent the result of a simulation as scalar values. Figure 7.2 shows an extract of the data set's structure. Attributes referring to the same position within the model are organized in folders. For example, the expanded output folder *Measuring Point 3* contains twelve attributes, which define the result of the simulation at a specific measuring point.

7.2 Goals

The case study in this chapter is going to demonstrate optimization, distance computation and data estimation. The results are illustrated in the context of the data analysis and visualization tool Bulk Analyzer.

Three sections build the main parts of the case study in this chapter. First, an optimization task is defined that demonstrates the search for optimal configurations in order to maximize the engine power under additional engine parameters. Several ways to visualize the results are presented. The second part deals with distance computation, showing a way to find configurations “close” to predefined parameters. The methods presented in this work are analyzed and evaluated. The third part shows a way to predict simulation runs by the use of a data estimation approach based on precalculated distance values.



INPUTS

- Case Parameters
  - Engine Speed
  - Exhaust Value Opening
  - Intake Value Closing Shift
  - Load Signal
  - T_Coolant

OUTPUTS

- simulation.dir
  - Curves
  - Summary
    - ACTUATOR 1
    - AIRCLEANER 1
    - CATALYST 1
    - CYLINDER 1
    - ENGINE 1
    - FUELINJECTOR 1
    - GLOBALS 1
    - MEASURINGPOINT 1
    - MEASURINGPOINT 10
    - MEASURINGPOINT 2
    - MEASURINGPOINT 3
      - CONVERGENCE (-)
      - ENTHALPYFLOW (J/cycle)
      - LOCATION (mm)
      - MACHNUMBER (-)
      - MASSFLOW (kg/cycle)
      - MASSFLOWAVERAGED-TEMPERATURE (K)
      - MASSFLOWPERS (kg/s)
      - PRESSURE (Pa)
      - REYNOLDSNUMBER (-)
      - TEMPERATURE (K)
      - VELOCITY (m/s)
      - WALLTEMPERATURE (K)

OUTPUTS (CONTINUED)

- MEASURINGPOINT 4
- MEASURINGPOINT 5
- MEASURINGPOINT 6
- MEASURINGPOINT 7
- MEASURINGPOINT 8
- MEASURINGPOINT 9
- PIPE 1
- PIPE 10
- PIPE 11
- PIPE 12
- PIPE 13
- PIPE 14
- PIPE 2
- PIPE 3
- PIPE 4
- PIPE 5
- PIPE 6
- PIPE 7
- PIPE 8
- PIPE 9
- PIPE_END 1
- PIPE_END 10
- PIPE_END 11
- PIPE_END 12
- PIPE_END 13
- PIPE_END 14
- PIPE_END 15
- PIPE_END 16
- PIPE_END 17
- PIPE_END 18
- PIPE_END 19
- PIPE_END 2
- PIPE_END 20

OUTPUTS (CONTINUED)

- PIPE_END 21
- PIPE_END 22
- PIPE_END 23
- PIPE_END 24
- PIPE_END 25
- PIPE_END 26
- PIPE_END 27
- PIPE_END 28
- PIPE_END 3
- PIPE_END 4
- PIPE_END 5
- PIPE_END 6
- PIPE_END 7
- PIPE_END 8
- PIPE_END 9
- PLENUM 1
- PLENUM 2
- PLENUM 3
- PLENUM 4
- PLENUM 5
- PLENUM 6
- PLENUM 7
- PORT 1
- PORT 2
- RESTRICTION 1
- RESTRICTION 2
- RESTRICTION 3
- SYSTEMBOUNDARY 1
- SYSTEMBOUNDARY 2

Figure 7.2: AVL BOOST Dataset. Output dimensions belonging together are grouped in folders. Data from AVL.

7.3 Pareto Optimization

This example considers six attributes of a BOOST data set. *Power* states the power output of the simulated engine (in kW). *Fuel mass flow* defines the injection rate of the fuel injector (in kilogram per second). *Engine speed* denotes the number of full rotations completed in one minute the simulated engine provides (in revolutions per minute). *Fuel-energy* describes the energy content of fuels (in Joules). Furthermore, two attributes are taken from measuring point 2: *pressure* (in Pascal) and *temperature* (in Kelvin). These attributes are added to the parallel coordinates view.

The first goal is to maximize engine power. Therefore, an objective is defined that maximizes the *power* axis. Simultaneously, *pressure* is minimized, *temperature* at measuring point 2 is set to a goal of 311 K and *fuel mass flow* is set to 0.002 kg/s for demonstration purposes (figure 7.3).

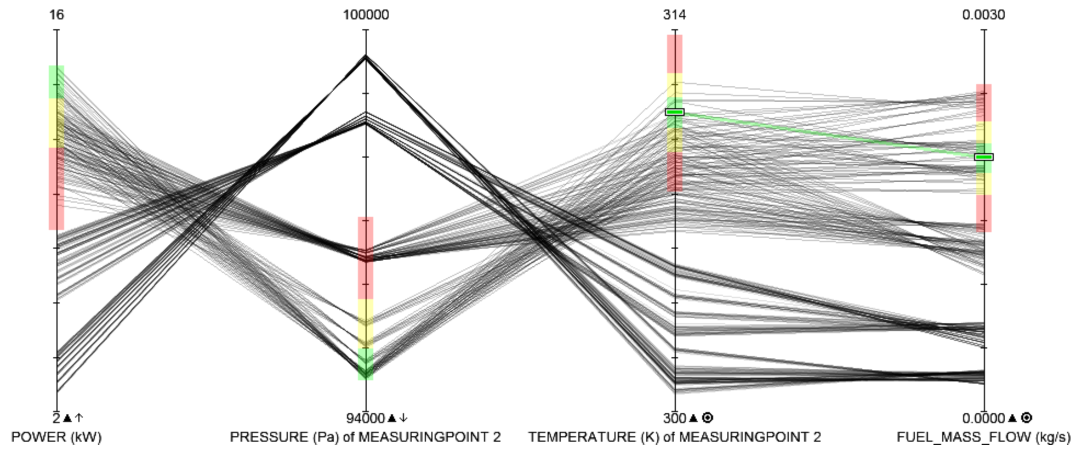


Figure 7.3: A maximization objective is set on *power*, *pressure* is minimized, *temperature* is set to a goal of 311 K and a goal of 0.002 kg/s is set on *fuel mass flow*. Data courtesy of AVL [19].

After the objectives are set, the optimization algorithm can be started. The algorithm creates the new attribute *set frontiers*, which will contain the Pareto frontier index of each simulation run as soon as the computation is done. In figure 7.4, this new data dimension is added to the parallel coordinates view as the rightmost axis. Furthermore, all data items belonging to the first Pareto frontier are selected (the current selection layer is colored red in this example). The first Pareto frontier contains 34 data items, which are not dominated by any other item in the Pareto sense. The frontier includes high values for engine speed. As explanation, the visualization shows that power is distinctly correlated with engine speed. The visualization also indicates that fuel-energy values around 1,750 J and 2,150 J are optimal for this configuration.

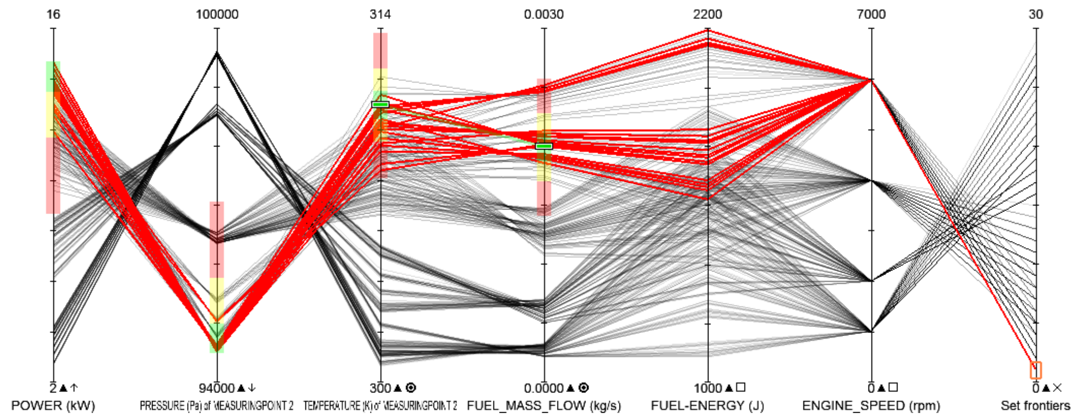


Figure 7.4: Selection of the first Pareto frontier in parallel coordinates. The current selection layer is defined by a selection of the first Pareto frontier on the *set frontiers* axis. Data courtesy of AVL [19].

The new dimension containing Pareto frontier indices can be used like any other attribute in the Bulk Analyzer system. Therefore, it is also possible to use it as color information for the parallel coordinates view, in order to improve the perception of the results. Figure 7.5 shows the results using a color gradient from red (low indices) to blue (high indices). The image clearly indicates the separation of regions which are near the optimum and those which are non-optimal.

The Bulk Analyzer visualization system also allows to use a certain dimension to be displayed as interpolated background color in the 2D scatter plot view. The background color is set to the mapped values of the underlying data, where the color between the data points is interpolated. To illustrate this, the input parameter *engine speed* is plotted against another input parameter *intake value closing shift*, in order to provide a visual exploration of the parameter space (figure 7.6). To find the Pareto optimal configuration with respect to the objectives above, it is possible to select the first Pareto frontier in parallel coordinates, which also highlights the points in the scatter plot by the linking and brushing concept. Another way is to use

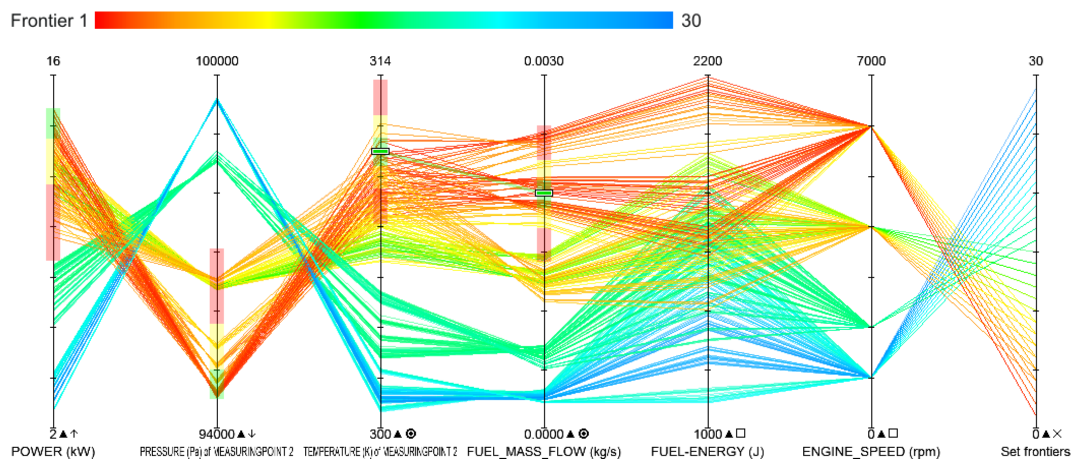


Figure 7.5: Pareto frontier indices as color information in parallel coordinates. Red polylines denote items having a low Pareto frontier index regarding to the current set of objectives; Green, yellow and red polylines indicate increasing frontier indices. Data courtesy of AVL [19].

the Pareto frontier indices as background color information. Again, red areas denote regions near the optimum, whereas blue areas indicate non-optimal regions. This way, the engineer quickly detects optimal settings by the analysis of the parameter space.

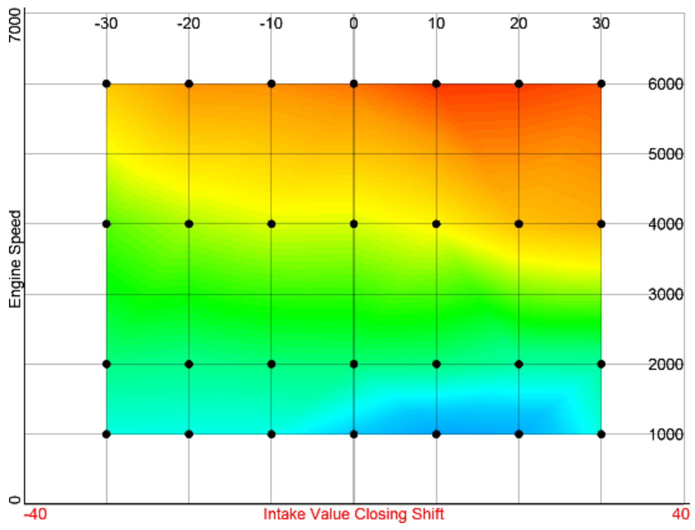


Figure 7.6: Pareto frontier indices as interpolated background color in a 2D scatter plot comparing input parameters of the simulation. Red areas denote regions having low Pareto frontier indices, blue areas indicate non-optimal regions. Data courtesy of AVL [19].

To give another example in the field of sales and marketing, the cars data set mentioned in chapter 2 is analyzed. Due to the high fuel price in 2008 and the upcoming financial crises, the demand for lightweight and petrol-saving cars increases [44]. To fulfill the customers' needs, the data set is explored using a parallel coordinates visualization and Pareto optimization. To save fuel, a maximization objective is set on MPG (miles per gallon). The weight is minimized as we are not looking for heavy, gas-guzzling cars. Furthermore, the intention is to sell new cars, so the maximum value of the model year axis is targeted. Figure 7.7 shows the results. The first Pareto frontier and, in contrast, the last frontier are selected. One noticeable

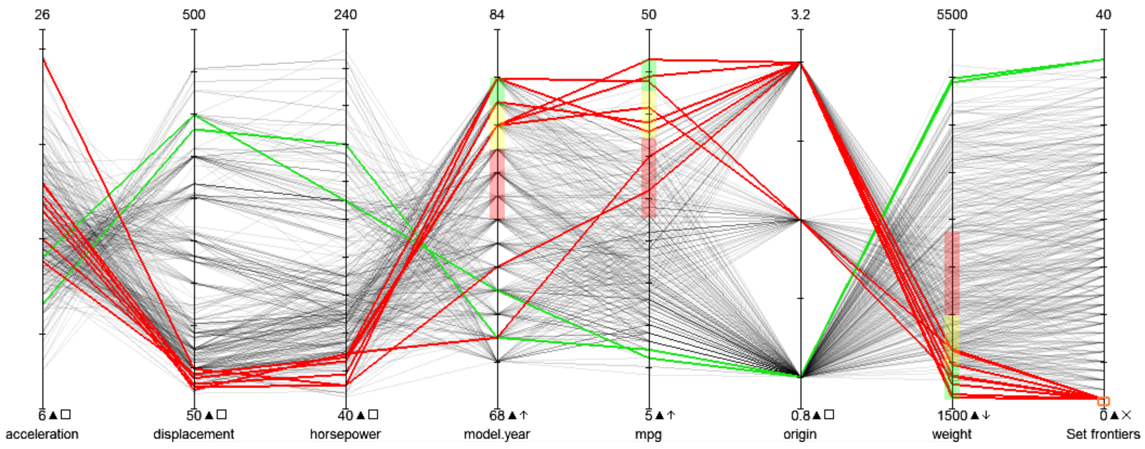


Figure 7.7: Pareto optimization analyzing cars in a parallel coordinates visualization. The "best" configurations (data items belonging to the first Pareto frontier) are red, the "worst" configurations (data items belonging to the last Pareto frontier) are green. Data courtesy of Donoho and Ramos [12].

issue is the origin of cars. The categorical axis *origin* maps American cars to 0, European cars to 1 and Japanese cars to 2. The first Pareto frontier (i.e., the "best" configurations in this optimization task) contains two European, six Japanese, but no American cars. Many US cars have a high Pareto frontier index, which indicates that customers would rather buy European or Japanese cars.

7.4 Distance Computation

For the following demonstration, four goals are set in parallel coordinates: 12 kW on the *power* axis, 99,000 Pa on the *pressure* axis, 310 K on the *temperature* axis and 0.0008 kg/s on the *fuel mass flow* axis. Using these (conflicting) objectives, distance values are computed as described in chapter 5. Figure 7.8 shows three distance computations (dimension *set distances*), each with different distance parameters. As mentioned, the distance computation approach consists of data scaling, the distance computation per dimension and aggregation to retrieve an overall distance. The following table shows the parameters used in this example.

Figure	Scaling mode	Computation mode (aggregation)
Top	Minimum/maximum	Maximum distance
Middle	Normalization	Average distance
Bottom	Division by goal	Weighted summed distance with weights
		60% on *power*
		10% on *pressure*
		10% on *temperature*
		20% on *fuel mass flow*

The right-most axis in each image contains the computed distance values. A distance of zero would mean that one or more data items exactly meet the target. The various computation modes focus on different aspects of data aggregation. The top image uses the maximum distance of all dimensions as overall distance. If a data item is very close to a goal in one dimension, but distant in another dimension, the overall distance will be high. In the middle image, the average distance is calculated, which is the arithmetic mean of the distances in all considered dimensions. The bottom image uses weighted sum aggregation. A weight is specified for each goal, which can be seen as “importance”. The distance on each dimension is weighted by the specified value and the resulting overall distance is the sum of these weighted distances.

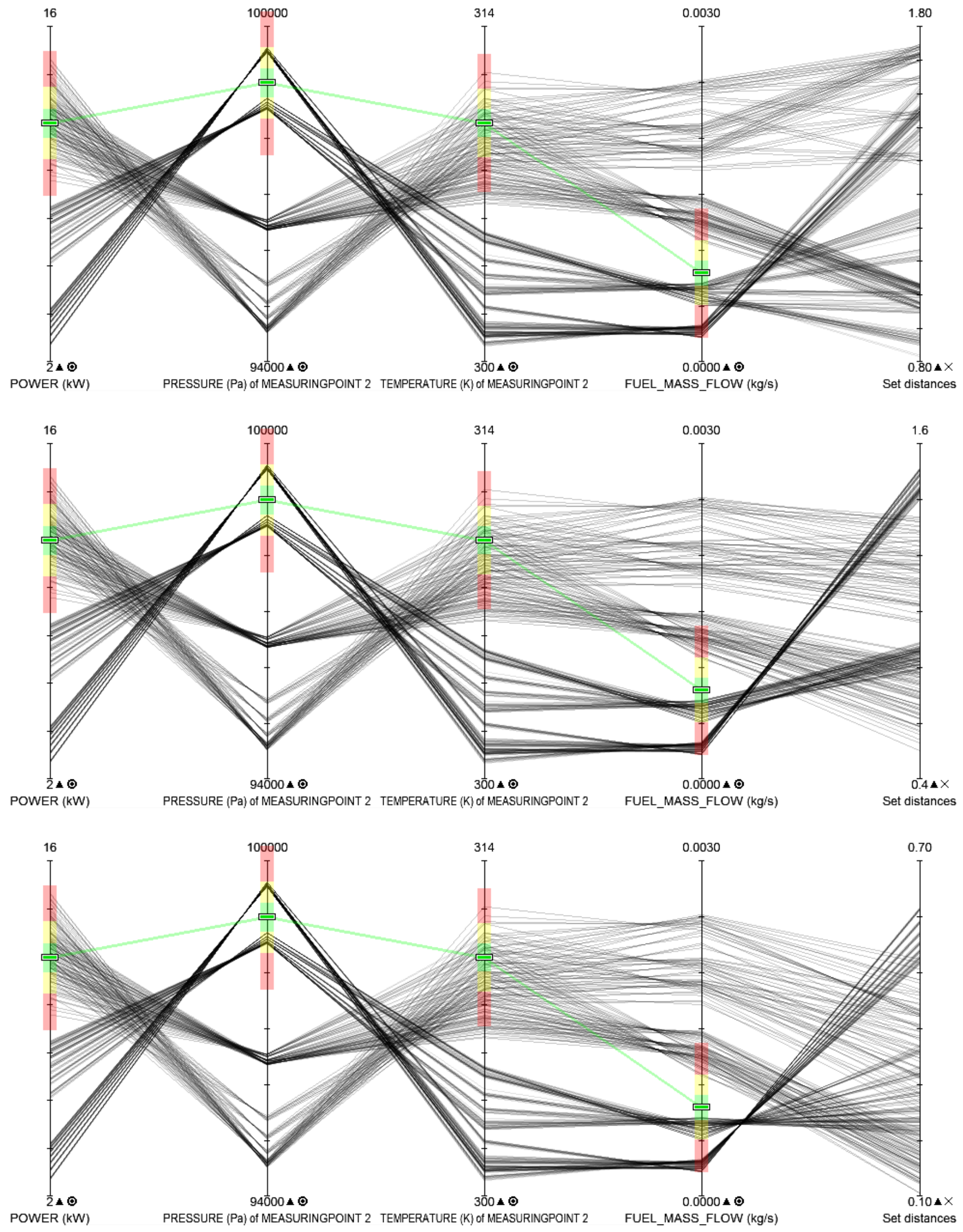


Figure 7.8: Comparison of distance computation modes. All images use the same goals for *power* (12 kW), *pressure* (99,000 Pa), *temperature* (310 K) and *fuel mass flow* (0.0008 kg/s). The used computation modes are listed in the table above. The right-most axis (*set distances*) contains the computed distance values. Data courtesy of AVL [19].

To demonstrate the effect of the non-linear exponent, which can be specified as an additional factor, three scatter plots are compared in figure 7.9. In each image, the *power* goal is set to 7.5 kW and the *temperature* goal to 305 K. The default distance parameters are used, which are normalization for scaling and average distance aggregation. The left image uses an exponent of 0.5, so the square root of all distances is computed before they are aggregated. In this case, all distances are closer to 1 after the computation and the number of small distances (the red area in the image) decreases. For the middle image, the default exponent 1 is used, which means no modification. The right image uses an exponent of 4. Distances < 1 are getting smaller, whereas higher distance values are expanded. In this example, many original distances are < 1, therefore, the red area in the image expands.

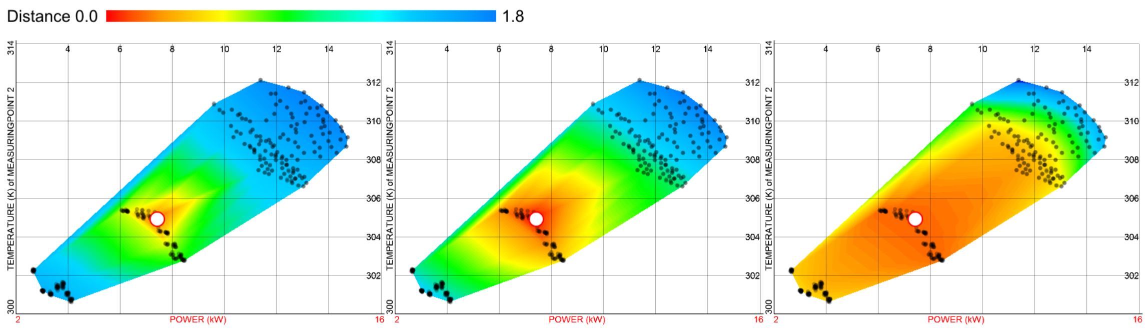


Figure 7.9: Effect of the non-linear exponent in distance computation. The white circles denote the two-dimensional goal. The left image uses an exponent of 0.5, which causes a reduced number of small distances; therefore the bluish area (high distance values) expands. The middle image shows the default distances (exponent 1). The right image uses an exponent of 4, which causes a concentration on the lower bound of the distances' range; therefore the red area (low distances) expands. Data courtesy of AVL [19].

7.5 Data Estimation

The final part of this case study deals with data estimation. The goal is to approximate data based on estimation targets by analyzing existing values. Although goals can be specified on arbitrary dimensions, two input parameters of the simulation data set are chosen here to demonstrate the interactive input-computation-output activity with real-world data.

The output dimensions *enthalpy flow*, which is a description of thermodynamic potential of a system (in Joules per cycle), *pressure* and *temperature* are added to the parallel coordinates view. The dimensions *engine speed* and *load signal* are added as input parameters. The goals on these input parameters are set to values that do not yet exist in the data set. The user specifies input values by setting multi-dimensional goal points and gets an idea of how the system behaves for that parameterization. As figure 7.10 shows, the approximated value on the *pressure* axis lies between two clusters. In this case, it is not clear, whether the value will tend to one of these clusters or not. Thus, there is a high uncertainty, expressed by the semitransparent band.

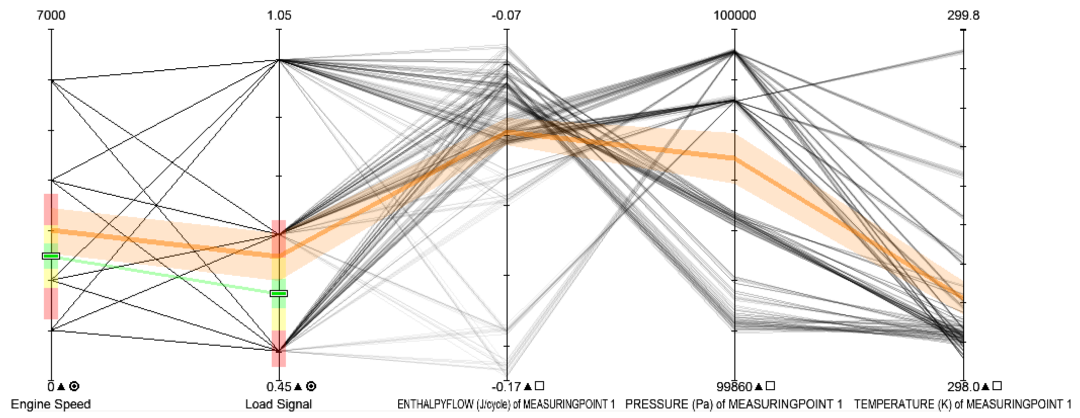


Figure 7.10: Data estimation in parallel coordinates. The two-dimensional estimation target consists of 2,500 rpm on *engine speed* and 0.6 on *load signal.* The orange polyline suggests the behavior of the system with respect to the modified input. The semitransparent band around this value indicates the confidence in the approximation. Data courtesy of AVL [19].

As mentioned in chapter 6, the approximation approach is based on distances, which are used as weights in the approximation process. Only values close to the estimation target are taken into account. The weights are determined by their overall distance to the goals. All other weights are set to zero. To illustrate the weights, figure 7.11 shows two more data estimation examples. The one-dimensional goal on *engine speed* is set to 4,500 rpm in the upper image and 6,500 rpm in the lower image. The weights are assigned to the right-most axis. The light polylines indicate values that are not relevant for the approximation process and thus having a weight of zero. The black polylines denote values close to the target. Small distances mean high weights and vice versa.

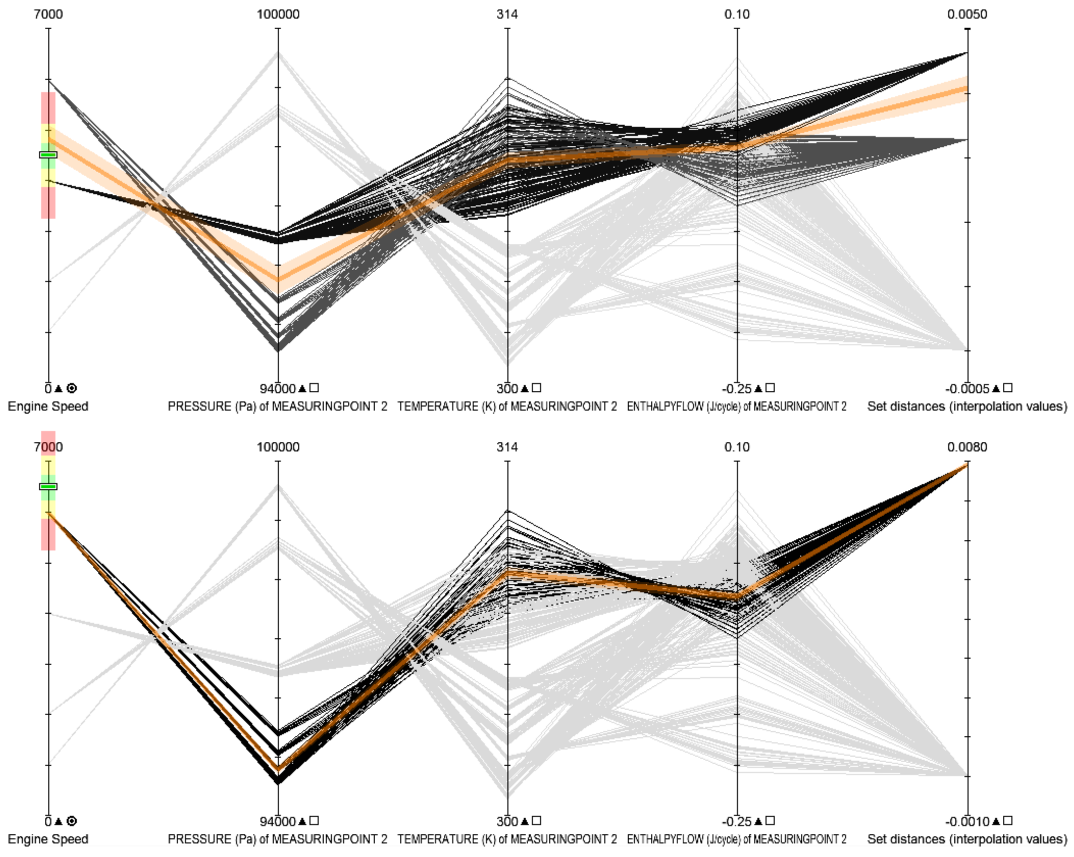


Figure 7.11: Weights used for data estimation in parallel coordinates. The light gray polylines denote values, which are not relevant for the approximation. The black polylines indicate values close to the target and therefore having a weight > 0. Data courtesy of AVL [19].

Chapter 8

Summary

Extracting important information from the large amount of data available today is essential in many fields of application. It is a common approach to apply application-specific algorithms to retrieve certain data characteristics. The integration of these algorithms into information visualization helps to identify and explore exact problem-specific solutions, which often cannot be provided by basic visualization techniques. This work presents three interactive approaches in the context of optimization, distance computation and data estimation.

8.1 Introduction

Visualization uses the human perception to give insight into data. It denotes the process of transforming data into images on the screen with the goals of exploration, analysis and presentation [5][24]. If visualization alone is not sufficient, it is a common approach to apply advanced data analysis algorithms on the data set to retrieve specific information.

The contribution of this work is to integrate particular automatic methods into information visualization. The goal is to achieve a tight interactive process between the user, the domain-specific computation and visualization. Figure 8.1 shows this *loop of interaction.* Information visualization helps to explore the current solution. The user analyzes the solution and uses the new knowledge to modify the algorithm parameters, if the results are not satisfying. Once the interaction process is finished, the computation is restarted and the new solution is visualized. This concept is used in the context of interactive optimization, distance computation and data estimation.

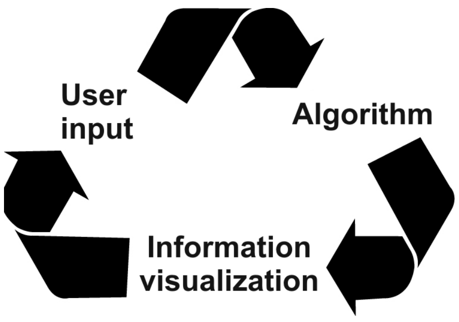


Figure 8.1: Loop of interaction. The tight coupling of user input, automated data processing and information visualization plays an important role in this work.

8.2 Related Work

When dealing with large data sets, data representations in form of tables are of limited use due to the lack of visual structures that can be perceived well by the human [5]. Information visualization tries to provide insight even for large and complex data. The illustrations in this work focus on *parallel coordinates*, a technique that maps multi-dimensional data onto a two-dimensional display screen [22]. Parallel coordinates use n axes to represent n data dimensions and aligns these axes in a parallel way. Data items are depicted by polygonal lines (*polylines*), which intersect each axis at the mapped position of the value in the particular dimension (figure 8.2).

8.2.1 Interactive Optimization

The first part of this work deals with multi-criteria optimization. Optimization can be defined as the search for the "best" solutions to mathematically defined problems [13]. In general, this means to maximize or minimize one or more functions. There is a wide variety of optimization problems, which can be classified by the variable type used in the objective function (discrete or continuous), the existence of constraints,

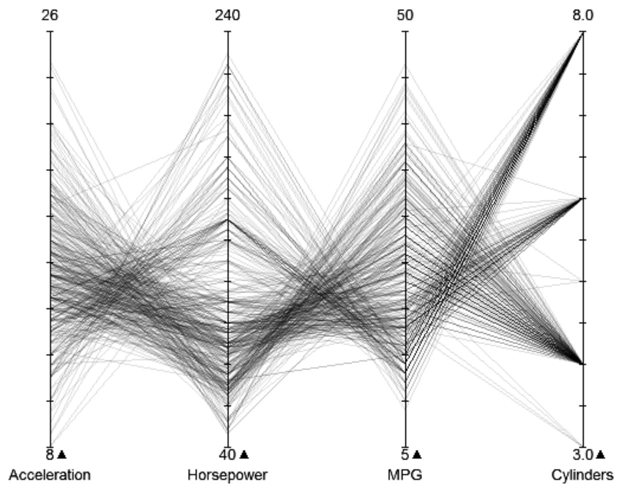


Figure 8.2: Parallel coordinates comparing cars. Data courtesy of Donoho and Ramos [12].

the nature of the solution (local or global), the used solution approach (stochastic or deterministic) or the number of objectives (single-objective or multi-objective) [38]. This work focuses on global solutions to multi-objective optimization problems.

Furthermore, *Pareto optimality* is used to find a global solution, which is based on *efficiency*. A point $\mathbf{x} \in S$ is efficient if its criterion vector is not dominated by the criterion vector of some other point in the feasible region $S \subset \mathbf{R}^n$. Thus, a point is efficient if it is not possible to increase one of the objectives without necessarily decreasing at least one of the others [54]. The set of efficient points is called *Pareto set* or *Pareto frontier* (figure 8.3) and usually represents the trade-off between conflicting goals.

Much research has been done on solving optimization problems. Therefore, there is a wide variety of algorithms and automatic optimization systems. However, many real-world optimization problems cannot be totally automated. In this case, user interaction is necessary for refining the optimization problem. Interactive optimization approaches are based on user feedback and leverage human abilities, such as visual perception or learning from experience. Users may explore many possible solutions and then choose a solution based on their understanding of the domain. This approach also helps to find a solution for multi-objective problems, where no unique solution exists and the user chooses the “subjective best” one.

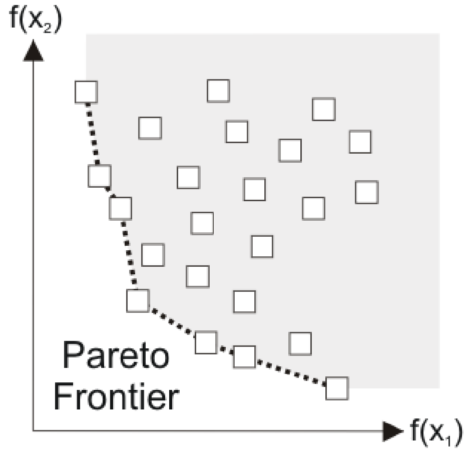


Figure 8.3: Efficient points define the Pareto frontier in a two-dimensional minimization problem.

Interactive approaches leverage the strengths of both humans and computers. Chimani et al. [7] define the cycle of user interaction as three activities: inspection, modification and reoptimization of the current solution. *Inspection* means to browse through the current solution with the assistance of information visualization. *Modification* is the main interaction step, in which optimization parameters are changed if the current solution is not satisfying. After this, the *reoptimization* step restarts the computation, which leads to the inspection step again. For example, *interactive evolution* algorithms generate solutions via biologically inspired methods and use human evaluation to produce better solutions in the next iteration step [56]. Other approaches that make use of the concept of interactive optimization are human-guided search [2][27], interactive partitioning [32], interactive design optimization [36] and interactive decision making [41].

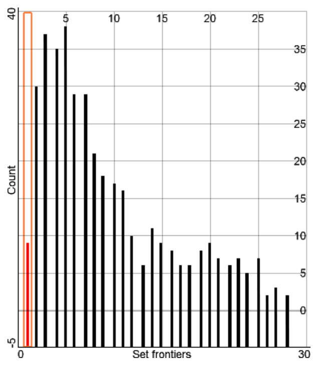
8.2.2 Memory-Based Reasoning

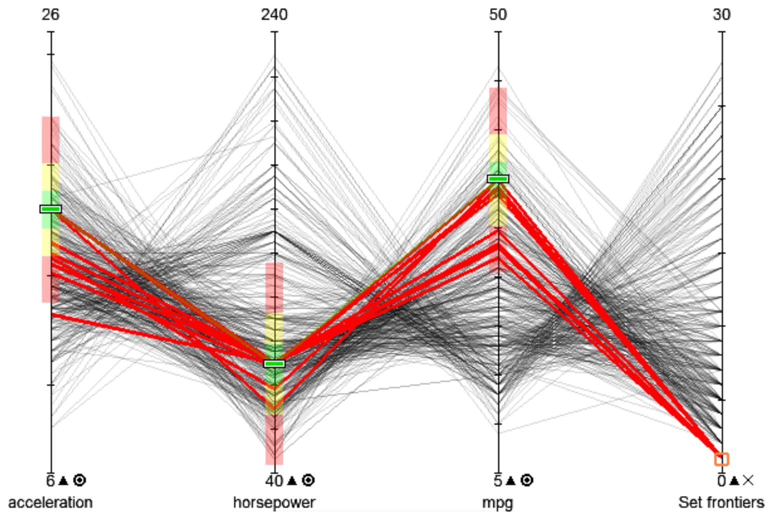
Memory-based reasoning (MBR) is an approach from the field of machine learning with the goal to predict unknown information from existing values. A problem is solved by finding similar cases in the past, and reusing them in the new problem situation [6]. This work shows a simple MBR approach to estimate data values according to given input parameters. Using these parameters, data is estimated by interpolation of existing values. In real-world applications, data is often generated by simulations. The goal of the approximation approach discussed in this work is to get a data "preview" of not yet simulated parameterizations based on already known values without starting costly simulation runs.

8.3 Interactive Optimization

Optimization is a mathematical discipline that concerns the finding of the optimum of given problems (e.g. to minimize or maximize functions) [54]. Especially when dealing with large, high-dimensional data sets, the use of automated optimization techniques is essential. However, many problems cannot be fully defined as mathematical problems. Often human evaluation is necessary to come to a solution. In this case, interactive optimization approaches can help if there are many solutions to choose from or problems are difficult to model with mathematics. For example, in design optimization the solution is based on subjective human interaction [36]. This work uses a global optimization algorithm in an interactive environment. The algorithm is based on Pareto optimality. It computes all Pareto frontiers with respect to user defined objectives, which can be manipulated interactively in parallel coordinates. Each objective is defined as maximization, minimization or as an arbitrary goal value. This represents a multi-objective optimization problem, which can be defined and modified directly within a parallel coordinates visualization. Figure 8.4 shows a parallel coordinates visualization comparing cars [12]. An optimization is defined containing objectives on acceleration, horsepower and miles per gallon. The right-most axis contains the Pareto frontier index of each data item and the first frontier is selected. Additionally, the histogram in the left image shows the number of data items in each frontier.

The visualization system allows the user to modify the objectives by dragging. At each user interaction, the computation is restarted and the results are updated. This generates a tight loop of interaction defined by user input, Pareto frontier computation and information visualization.





(a) (b)

Figure 8.4: Selection of a Pareto frontier of an optimization: (a) Histogram showing the number of data items in each Pareto frontier; (b) The optimization in parallel coordinates. The right-most axis represents the Pareto frontier dimension. The highlighted values denote the first frontier. Data courtesy of Donoho and Ramos [12].

8.4 Distance Computation

Another way to determine the fitness of data with respect to predefined n-dimensional goals is to compute distances. Various approaches are given in this work to compute a distance value for each multi-dimensional data item. The computation of distances involves three steps. The first step is to scale the data in order to unify potentially different data ranges of the dimensions. The second step is the computation of the distances to the predefined goal in each dimension. In the last step, the distances of all dimensions are aggregated to obtain an overall distance value for each data item.

This work describes four scaling methods. *Minimum/maximum scaling* transforms all values to the range $[-1, 1]$. *Statistical normalization* uses the statistical values mean and standard deviation for the data transformation. *Linear interpolation* is used to project the values to a common range by the help of user-defined distances. *Division* divides all values by either the average (mean value) or the goal value.

The distances are then computed using the scaled values x_i' and the scaled goal $goal'$ for each considered dimension. Additionally, a non-linear exponent can be specified to influence the resulting distances concerning outlier sensibility.

$$d_i = |x_i' - goal'|^{exponent} \tag{8.1}$$

The final aggregation of the distances d_i can also be done in several ways. In the following, m denotes the number of goals.

Maximum distance

$$d_i = \max_{obj} d_{obj,i}. \tag{8.2}$$

Average distance

$$d_i = \frac{1}{m} \sum_{obj=1}^{m} d_{obj,i} \tag{8.3}$$

Summed distance

$$d_i = \sum_{obj=1}^{m} d_{obj,i} \tag{8.4}$$

Weighted summed distance

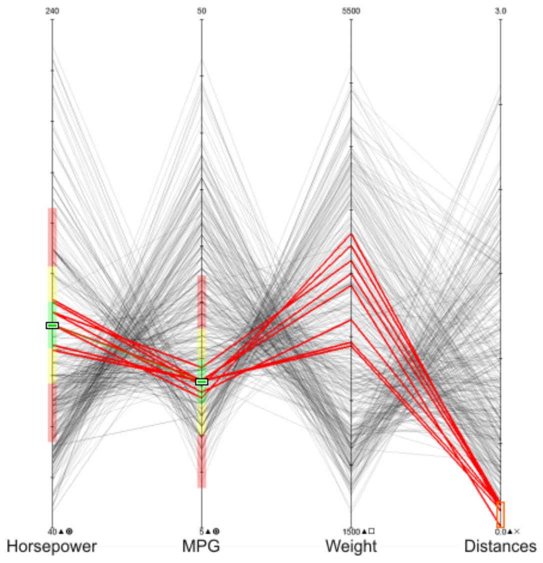
$$d_i = \sum_{obj=1}^{m} (d_{obj,i} \cdot w_{obj}). \tag{8.5}$$

Summed distance

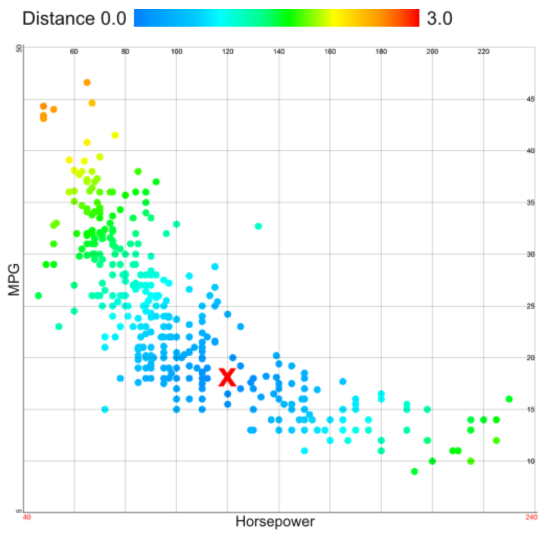
$$d_i = \sum_{obj=1}^{m} d_{obj,i} \tag{8.6}$$

Figure 8.5 shows the results using the cars data set [12]. Statistical normalization was chosen for data scaling and the average distance was used as aggregation mode. The left image shows a parallel coordinates visualization containing computed distances (the right-most axis). The eight lowest distances are selected to improve the perceptibility of data items having low distance to the defined goals. In the right image, the distances are used as color information in a 2D scatter plot.

Using a color transfer function, blue points mean low distance in this example, red points mean high distance.



(a)



(b)

Figure 8.5: Distance computation using goals on horsepower (120) and MPG (18): (a) Parallel coordinates comparing horsepower, MPG and weight. The eight lowest distances are selected on the distance dimension (right-most axis); (b) Distance values used as color information in a 2D scatter plot. Horsepower is plotted against MPG; the two goals define a point in the plane, which is depicted by a red cross. Data courtesy of Donoho and Ramos [12].

8.5 Data Estimation

Data sets are in most cases either collected by measurements or generated by simulations. These simulations can be very complex and time-consuming. If input parameters of the simulations are changed, it is sometimes helpful to provide approximations of not yet simulated parameterizations in order to get a "preview" of results at that position of the parameter space without starting costly simulations. This work shows an approach, which is based on the analysis of existing data values. Estimation targets are defined by a multi-dimensional goal. A weight is assigned to each data item by the help of distances to this goal. High distances to the goal mean low weights, low distances mean high weights. These weights are normalized and then used for the data estimation. In the following equation, w_i' denotes the normalized weight assigned to the i^{th} data item, n denotes the number of data items, and m means the m^{th} dimension.

$$I_m = \sum_{i=1}^{n} (x_{m,i} \cdot w_i') \tag{8.7}$$

This vector I represents the estimated values for all result dimensions of the simulation. The weighted average distance between the estimated and the actual data values can additionally be used to assess the confidence of the estimation.

$$S_m = \frac{1}{n} \sum_{i=1}^{n} (|x_{m,i} - I_m| \cdot w_i') \tag{8.8}$$

Low values indicate a high confidence in the result of the respective dimension and vice versa. Figure 8.6 shows an estimation in parallel coordinates. The data set is provided by AVL [19], a company offering power train engineering and testing solutions in both the automotive and nonautomotive industries. The estimation target is defined on two input parameters of the simulation: 3,150 rpm on engine speed and 0.77 on load signal. The orange polyline shows the result of the approximation. The semitransparent band around this polyline denotes the confidence in the approximation. On the third axis (convergence), the band is thin, which means that the prediction is rather robust in this dimension.

Interaction is a key concept in this approach. Once the estimation target is modified by dragging, the computation is restarted automatically and the result is updated. This loop of interaction makes it possible to quickly explore "virtual" simulations on arbitrary input parameters.

8.6 Conclusion

The major goal of this work is to show the benefit of integrating automatic methods into information visualization in an interactive manner. The approaches in this work deal with multiple objectives. Direct manipulation of objectives inside the parallel coordinates visualization is a key aspect to provide convenient interaction. Future

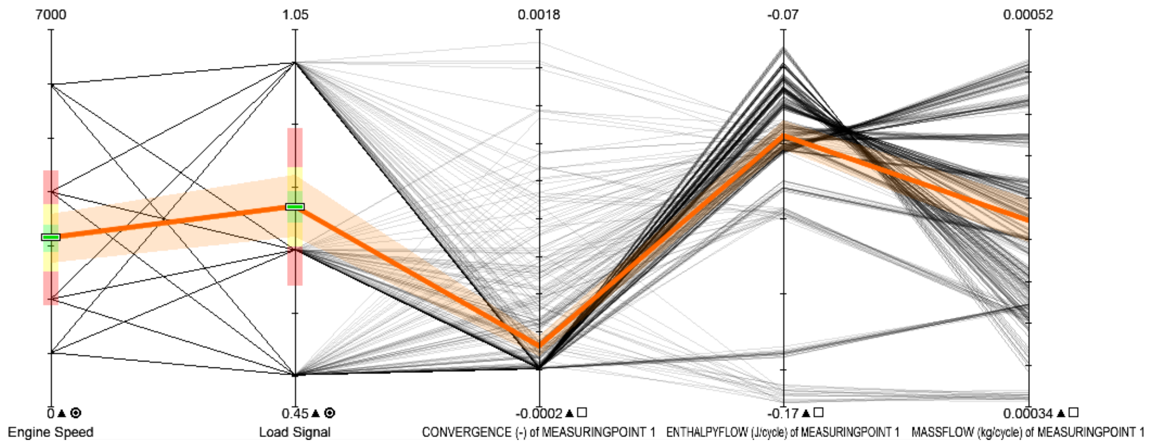


Figure 8.6: Data estimation using real-world simulation data of an engine manufacturer. The target is defined by goals on two input parameters of the simulation (3,150 on engine speed and 0.77 on load signal). The orange polyline shows the approximated simulation result, bordered by the semitransparent confidence information. Data courtesy of AVL [19].

extensions could provide visual and functional representations of objectives in other views, like a movable goal point in a scatter plot.

The optimization approach uses an algorithm based on Pareto optimality. A considerable amount of work has been done in the field of optimization, however, interactive approaches are rare. The approach studied in this work is a contribution to interactive optimization. An extension to the shown algorithm could be to define weights on the objectives in order to set different priorities to different dimensions.

The computation of distances with respect to multi-dimensional targets can be used in various applications. Further implementations could use distance values for application-specific algorithms (e.g., in the field of cluster analysis). The data estimation approach in this work uses distances to compute weights for existing data values, which are used in the approximation process. Interactive optimization, distance computation and data estimation support decision makers to explore relevant subsets of large data sets.

Chapter 9

Conclusion and Future Work

Automatic methods to explore large data sets have been used in many fields of application. One drawback of automated approaches is that the problem must be fully specified before application. The major contribution of this work is tight interactive combination of information visualization and problem-solving algorithms. User feedback in terms of objective modification allows to explore the solution space in order to get results under subjective human guidance. The user is able to specify the algorithm parameters directly in the visualization of the data set. The system reacts on this modification and restarts the computation. This concept defines an intuitive loop of interaction. In an analysis dialog, the user observes the current solution, interprets the results and then formulates a strategy of how to proceed.

This work focuses on multiple objectives, which occur in many fields of application. Direct manipulation of objectives inside the visualization provides convenient interaction. The manipulation and visual representation of objectives has been integrated in a parallel coordinates visualization. Future extensions could provide visual and functional representations of objectives in other views. For example, a movable goal point in a 2D scatter plot could be used to define targets simultaneously in two dimensions.

The optimization approach uses an algorithm based on Pareto optimality, which is a common way to deal with multi-objective problems. The algorithm is deterministic and computes the global solution to the defined optimization problem. Since goals can be defined as arbitrary values, this approach is very flexible and intuitive at the same time. Only limited research was done on interactive optimization and many approaches in this field were designed for specific applications. This approach is a contribution to interactive optimization in a general sense. Future improvements could provide the ability to define weights on the objectives, which would allow for ranking the values inside each Pareto frontier. Another extension could be to set implicit optimization constraints to restrict the solution set.

The concept of distance offers additional information about the data set under observation. Distance values with respect to multi-dimensional targets can be used in various situations. For example, the implemented data estimation approach uses distances as weighting basis. Distances can be seen as degree of closeness and similarity. Further implementations could reuse these values to solve problems by application-specific algorithms and visualizations. In the field of cluster analysis, for instance, similarity is often according to a distance measure.

The goal of this work is to show the benefit of integrating automatic methods into information visualization in an interactive manner. This motivation led to an interactive approach dealing with multi-objective problems, which are defined as modifiable goals in parallel coordinates. This approach helps users to find solutions in an interactive process. Interactive optimization, distance computation and data estimation support decision makers to extract important information from large data sets in order to increase the efficiency of their applications.

Chapter 10

Acknowledgments

This work has been done at the VRVis Research Center in Vienna, Austria, whose business partner AVL List GmbH kindly provided datasets used during research.

I want to thank Harald Piringer for his assistance and his ideas in this interesting field of research. He always was very supportive in conceptual and technical questions and I have learned a lot about academic research and writing during the work on this thesis.

I also want to thank Professor Eduard Gröller, who supervised this thesis in an uncomplicated and constructive manner.

Last but not least, special thanks go to Anja Zachhuber for her patience and everything else for all these years.

Bibliography

[1] Agnar Aamodt and Enric Plaza. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications*, 7(1):39–59, 1994.

[2] David Anderson, Emily Anderson, Neal Lesh, Joe Marks, Brian Mirtich, David Ratajczak, and Kathy Ryall. Human-Guided Simple Search. *Proceedings of the 17th National Conference on Artificial Intelligence and 12th Conference on Innovative Applications of Artificial Intelligence*, pages 209–216, 2000.

[3] Mihael Ankerst, Stefan Berchtold, and Daniel A. Keim. Similarity Clustering of Dimensions for an Enhanced Visualization of Multidimensional Data. *Proceedings of the IEEE Symposium on Information Visualization 1998*, pages 52–60, 1998.

[4] Mihael Ankerst, Daniel A. Keim, and Hans-Peter Kriegel. 'Circle Segments': A Technique for Visually Exploring Large Multidimensional Data Sets. *Proc. Visualization '96, Hot Topic Session, San Francisco, CA*, 1996.

[5] Stuart K. Card, Jock D. Mackinlay, and Ben Shneiderman. *Readings in Information Visualization: Using Vision to Think.* Morgan Kaufmann Publishers, San Francisco, USA, 1999.

[6] Venkata U. B. Challagulla, Farokh B. Bastani, and I-Ling Yen. A Unified Framework for Defect Data Analysis Using the MBR Technique. *Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence*, pages 39–46, 2006.

[7] Markus Chimani, Neal Lesh, Michael Mitzenmacher, Candy L. Sidner, and Hidetoshi Tanaka. A Case Study in Large-Scale Interactive Optimization. *International Conference on Artificial Intelligence and Applications*, pages 24–29, 2005.

[8] Carlos A. Coello, Gary B. Lamont, and David A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems.* Springer, New York, USA, 2nd edition, 2007.

[9] Yann Collette and Patrick Siarry. *Multiobjective Optimization. Principles and Case Studies.* Springer, Berlin, Germany, 2003.

[10] Wikipedia contributors. *Vilfredo Pareto*, 2008. Online available at `http://en.wikipedia.org/wiki/Vilfredo_Pareto`; visited on April 9, 2008.

[11] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms.* MIT Press, Cambridge, USA, 2nd edition, 2001.

[12] David Donoho and Ernesto Ramos. *PRIMDATA: Data sets for use with PRIM-H, Cars dataset*, 1983. Online available at StatLib Datasets Archive (`http://lib.stat.cmu.edu/datasets`); visited on May 3, 2008.

[13] Roger Fletcher. *Practical Methods of Optimization.* John Wiley & Sons, Ltd., Chichester, UK, 1980.

[14] Forensic Imaging Group, University of Kent, UK. *Online Experiments.* Online available at `http://www.survey.visionmetric.com`; visited on September 10, 2008.

[15] Ying-Huey Fua, Matthew O. Ward, and Elke A. Rundensteiner. Hierarchical Parallel Coordinates for Exploration of Large Datasets. *Proceedings of the IEEE conference on Visualization '99*, pages 43–50, 1999.

[16] George W. Furnas. Generalized Fisheye Views. *Proceedings of the CHI Conference on Human Factors in Computing Systems '86*, pages 16–23, 1986.

[17] Marc Geilen, Twan Basten, Bart Theelen, and Ralph Otten. An Algebra of Pareto points. *Proceedings of the Fifth International Conference on Application of Concurrency to System Design*, pages 88–97, 2005.

[18] AVL List GmbH. *BOOST – Engine Cycle and Gas Exchange Simulation (Product Description).* Online available at `http://www.avl.com/boost`; visited on August 26, 2008.

[19] AVL List GmbH. *AVL BOOST — Advanced Engine Cycle, Aftertreatment and Duct Acoustic Simulation (Data Sheet)*, 2008. Online available at `http://www.avl.com/boost`; visited on August 6, 2008.

[20] David Hand, Heikki Mannila, and Padhraic Smyth. *Principles of Data Mining.* MIT Press, Cambridge, USA, 2001.

[21] Helwig Hauser, Florian Ledermann, and Helmut Doleisch. Angular Brushing of Extended Parallel Coordinates. *IEEE Symposium on Information Visualization (InfoVis 2002)*, pages 127–130, 2002.

[22] Alfred Inselberg and Bernard Dimsdale. Parallel coordinates: a tool for visualizing multi-dimensional geometry. *Proceedings of the 1st conference on Visualization '90*, pages 361–378, 1990.

[23] Simon Kasif, Steven Salzberg, David Waltz, John Rachlin, and David Aha. Towards a Framework for Memory-Based Reasoning. *NECI Technical Report*, pages 95–132, 1995.

[24] Daniel A. Keim. Visual Techniques for Exploring Databases, Invited Tutorial. *Int. Conference on Knowledge Discovery in Databases (KDD'97)*, 1997.

[25] Daniel A. Keim. Information Visualization and Visual Data Mining. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):1–8, 2002.

[26] Daniel A. Keim, Florian Mansmann, Jörn Schneidewind, and Hartmut Ziegler. Challenges in Visual Data Analysis. *Proceedings of the Tenth International Conference on Information Visualization (IV 2006)*, pages 9–16, 2006.

[27] Gunnar W. Klau, Neal Lesh, Joe Marks, and Michael Mitzenmacher. Human-Guided Tabu Search. *Proceedings of the 18th National Conference on Artificial Intelligence*, pages 41–47, 2002.

[28] Gunnar W. Klau, Neal Lesh, Joe Marks, Michael Mitzenmacher, and Guy T. Schafer. The HuGS Platform: A Toolkit for Interactive Optimization. *Proceedings of the Workshop on Advanced Visual Interfaces*, pages 324–330, 2002.

[29] Ernst Kleiberg, Huub Van de Wetering, and Jarke J. Van Wijk. Botanical visualization of huge hierarchies. *Proceedings of the IEEE Symposium on Information Visualization 2001*, pages 87–94, 2001.

[30] Robert Kosara, Fabian Bendix, and Helwig Hauser. Parallel Sets: Visual Analysis of Categorical Data. *Transactions on Visualization and Computer Graphics*, 12(4):558–568, 2006.

[31] Michael D. Lee, Rachel E. Reilly, and Marcus A. Butavicius. An Empirical Evaluation of Chernoff Faces, Star Glyphs, and Spatial Visualizations for Binary Data. *Proceeding of the Australian Symposium on Information Visualisation*, pages 1–10, 2003.

[32] Neal Lesh, Joe Marks, and Maurizio Patrignani. Interactive Partitioning (System Demonstration). *Proceedings of the 8th International Symposium on Graph Drawing*, pages 31–36, 2000.

[33] Peter Lyman and Hal R. Varian. *How Much Information*, 2003. Online available at `http://www.sims.berkeley.edu/how-much-info-2003`; visited on April 8, 2008.

[34] Jock D. Mackinlay. Opportunities for Information Visualization. *IEEE Computer Graphics and Applications*, 20(1):22–23, 2000.

[35] Jiří Matoušek and Bernd Gärtner. *Understanding and Using Linear Programming.* Springer, New York, USA, 2006.

[36] Jeremy J. Michalek and Panos Y. Papalambros. Interactive Design Optimization of Architectural Layouts. *Engineering Optimization*, 34(5):485–501, 2002.

[37] Kaisa Miettinen. *Nonlinear Multiobjective Optimization.* Kluwer Academic Publishers, Boston, USA, 1999.

[38] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization.* Springer, New York, USA, 1999.

[39] A. Oliver, O. Regragui, N. Monmarch, and G. Venturini. Genetic and Interactive Optimization of Web Sites. *Eleventh International World Wide Web Conference*, 2002.

[40] Wei Peng, Matthew O. Ward, and Elke A. Rundensteiner. Clutter Reduction in Multi-Dimensional Data Visualization Using Dimension Reordering. *Proceedings of the IEEE Symposium on Information Visualization 2004*, pages 89–96, 2004.

[41] Hasan Pirkul, Rakesh Gupta, and Erik Rolland. VisOpt: A Visual Interactive Optimization Tool for P-Median Problems. *Decision Support Systems*, 26(3):209–223, 1999.

[42] John Rachlin, Simon Kasif, Steven Salzberg, and David W. Aha. Towards a Better Understanding of Memory-Based Reasoning Systems. *Proceedings of the 11th International Conference on Machine Learning*, pages 242–250, 1994.

[43] Ramana Rao and Stuart K. Card. The Table Lens: Merging Graphical and Symbolic Representations in an Interactive Focus+Context Visualization for Tabular Information. *Proceedings of the CHI Conference on Human Factors in Computing Systems '94*, pages 318–322, 1994.

[44] John Reed. *Manufacturers adapt with smaller vehicles.* Published on Financial Times Online on October 1, 2008; Online available at `http://www.ft.com`; visited on January 15, 2009.

[45] Theresa-Marie Rhyne. Does the difference between information and scientific visualization really matter? *IEEE Computer Graphics and Applications*, 23(3):6–8, 2003.

[46] Geraldine E. Rosario, Elke A. Rundensteiner, David C. Brown, and Matthew O. Ward. Mapping Nominal Values to Numbers for Effective Visualization. *Proceedings of the IEEE Symposium on Information Visualization 2003*, pages 113–120, 2003.

[47] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 2nd edition, 2003.

[48] Stacey D. Scott, Neal Lesh, and Gunnar W. Klau. Investigating Human-Computer Optimization. *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*, pages 155–162, 2002.

[49] Ben Shneiderman. Tree Visualization with Tree-Maps: 2-d Space-Filling Approach. *ACM Transactions on Graphics*, 11(1):92–99, 1992.

[50] Harri Siirtola. Direct Manipulation of Parallel Coordinates. *Proceedings of the CHI Conference on Human Factors in Computing Systems '00*, pages 119–120, 2000.

[51] James C. Spall. *Introduction to Stochastic Search and Optimization.* John Wiley & Sons, Inc., New York, USA, 2003.

[52] Robert Spence and Mark Apperley. Data base navigation: An office environment for the professional. *Behaviour and Information Technology*, 1(1):43–54, 1982.

[53] Craig Stanfill and David Waltz. Toward Memory-Based Reasoning. *Communications of the ACM*, 29(12):1213–1228, 1986.

[54] Ralph E. Steuer. *Multiple Criteria Optimization: Theory, Computation, and Application.* John Wiley & Sons, Inc., New York, USA, 1986.

[55] Wenyu Sun and Ya-xiang Yuan. *Optimization Theory and Methods: Nonlinear Programming.* Springer, New York, USA, 2006.

[56] Hideyuki Takagi. Interactive Evolutionary Computation: Fusion of the Capabilities of EC Optimization and Human Evaluation. *Proceedings of the IEEE*, 89(9):1275–1296, 2001.

[57] VRVis Center for Virtual Reality and Visualization Research, Ltd. Additional information online available at `http://www.vrvis.at`; visited on May 7, 2008.

[58] Tao Wang. *Global Optimization for Constrained Nonlinear Programming.* Phd thesis in computer science, University of Illinois, 2001.

[59] Thomas Weise. *Global Optimization Algorithms - Theory and Application.* Thomas Weise, 2nd edition, 2008. Online available as e-book at `http://www.it-weise.de`; visited on July 8th 2008.

[60] Laurence A. Wolsey. *Integer Programming.* John Wiley & Sons, Inc., New York, USA, 1998.

[61] Hong Zhou, Xiaoru Yuan, Huamin Qu, Weiwei Cui, and Baoquan Chen. Visual Clustering in Parallel Coordinates. *Computer Graphics Forum (Proceedings of EuroVis '08)*, 27(3):1047–1054, 2008.