Chapter 1: Introduction

Practical optimization is the art and science of allocating scarce resources to the best possible effect. Optimization techniques are called into play every day in questions of industrial planning, resource allocation, scheduling, decision-making, etc. For example, how does a global petroleum refiner decide where to buy crude oil, where to ship it for processing, what products to convert it to, where to sell those products, and at what prices? A maximum-profit optimization model is used to solve this problem. How does an airline know how to route its planes and schedule its crews at minimum cost while meeting constraints on airplane flight hours between maintenance and maximum flight time for crews? A minimum-cost optimization model is used.

Many of the large scale optimization techniques in general use today can trace their origins to methods developed during World War II to deal with the massive logistical issues raised by huge armies having millions of men and machines. Any techniques that promised to improve the effectiveness of the war effort were desperately needed, especially in the face of limited numbers of people, machines, and supplies. What is the optimum allocation of gasoline supplies among competing campaigns? What is the best search and bombing pattern for anti-submarine patrols?

The fundamentals of the first practical, large-scale optimization technique, the simplex method, were developed during the war. The simplex method was perfected shortly after the war when the first electronic computers were becoming available. In fact, the early history of computing is closely intertwined with the history of practical optimization. In the early years, the vast majority of all calculation on electronic computers was devoted to optimization via the simplex method!

The entire history of large-scale practical optimization is very short. In fact, the inventor of the simplex method, George Dantzig, died only very recently in 2005. He told an amusing story at the 1997 International Symposium on Mathematical Programming. A graduate student of optimization, on hearing that he was present at the meeting, rushed up to him and pumped his hand excitedly, exclaiming "Professor Dantzig! I'm so glad to meet you! I thought you were dead!"

The field itself is an exciting hotbed of innovation, and a terrific field for researchers. Spectacular breakthroughs are still happening, like the development of interior point methods for linear programming in the late 1980s. More recently, ideas abound as researchers from mathematical backgrounds trade thoughts with researchers from a computer science tradition. An important example is the emergence of constraint programming as a powerful paradigm for optimization. One of the largest companies in the field of practical optimization, Ilog, is also one of the newest, and has achieved its position by combining the best of the mathematical techniques with the best of the constraint programming techniques from computer science.

New optimization techniques are arriving daily, often stimulated by fascinating insights from other fields. Genetic algorithms, for example, use an analogy to chromosome encoding and natural selection to "evolve" good optimization solutions. Optimization techniques play a role in training artificial neural networks used in artificial intelligence research for pattern recognition. "Swarm intelligence" research looks at using independent software agents to collectively solve various optimization problems. Perhaps you have ideas of your own: maybe you can be the next George Dantzig….

Today, optimization methods are used everywhere in business, industry, government, engineering, and computer science because optimization problems arise just as regularly in these fields as they did during World War II. An edge in maximizing profits or minimizing costs can often mean the difference between success and failure in business. In academia, the teaching of optimization takes place in several different departments and under different course names. You will find optimization taught in university departments of engineering, computer science, business, mathematics, and economics; and in courses going under the names of optimization, mathematical programming, operations research, industrial engineering, and algorithms.

Here are a few more examples of optimization problems:

- How should the transistors and other devices be laid out in a new computer chip so that the layout takes up the least area?
- What is the smallest number of warehouses, and where should they be located so that the maximum travel time from any retail sales outlet to the closest warehouse is less than 6 hours?
- How should telephone calls be routed between two cities to permit the maximum number of simultaneous calls?

The process of optimization is shown schematically in Figure 1.1. You typically begin with a real problem, full of details and complexities, some relevant and some not. From this you extract the essential elements to create a model, and choose an algorithm or solution technique to apply to it. In practical problems, the computer will carry out the necessary calculations.

There is an unavoidable loss of realism as you move down the diagram, from *real world problem* to *algorithm, model, or solution technique*, and finally to *computer implementation*. An important part of successful applied optimization is skill in determining what is and what is not important during this process of abstraction. For example, suppose that production efficiency increases slightly nonlinearly as production volume increases on your assembly line. Is the nonlinearity significant, or is it accurate enough to use a much simpler linear approximation? And where do you get the "profit per item produced" that your model requires? In reality it is an estimate produced by the accounting department who consider many factors in arriving at the number, including the pay rates for workers mandated by the last union contract, the estimated prices of raw materials and energy, and the estimated sales based on forecasts provided by the marketing department. The trick is to learn, through experience, how to match messy

real world situations with the correct optimization algorithms, and how to combine algorithms to create new solvers as the situation may demand.

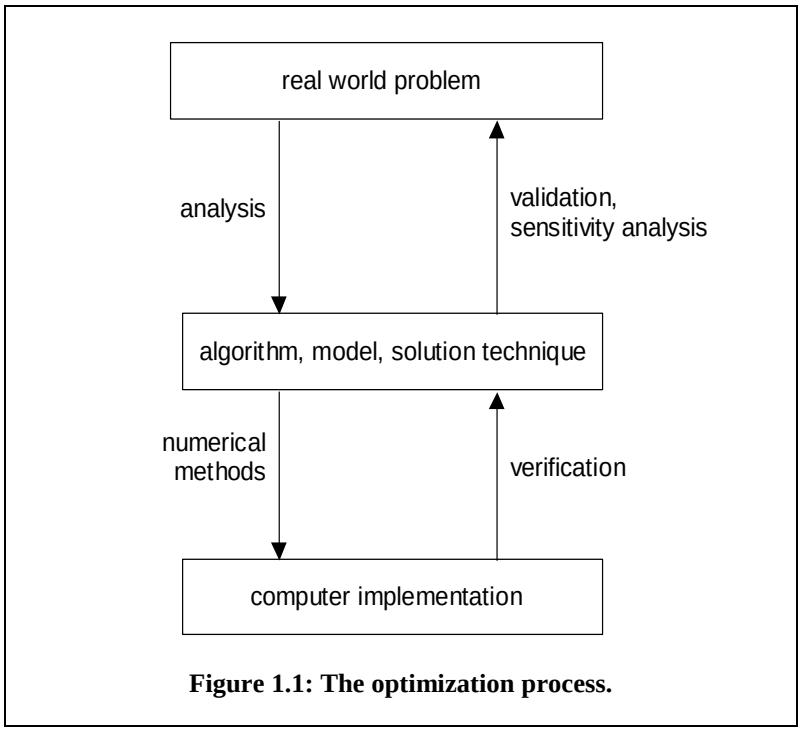

Figure 1.1: The optimization process.

The arrows in Figure 1.1 indicate the normal process of the optimization cycle. Moving from the *real world problem* to the *algorithm, model, or solution technique* is known as **analysis.** It is here that the main work of abstracting away irrelevant details and focusing on the important elements takes place. In many cases, the process of analysis is extremely valuable by itself, even without carrying out any subsequent optimization. Just laying bare the essential elements of a problem is often extremely valuable to the client, and leads to insights into how to approach the problem. Many of the skills needed here are in courses titled *systems analysis* in software engineering or computer science programs. Skills needed include the ability to interact with people to find information, and expertise with tools for expressing your understanding of the situation in a manner that is clear, complete and easily understood by nontechnical folks. I highly recommend a course in systems analysis for anyone who will be doing practical optimization.

Moving from the *algorithm, model, or solution technique* to the *computer implementation* is generally the province of **numerical methods** (and other computer science techniques). This covers issues such as calculation accuracy when using digital computers, efficient implementations of matrix inversion techniques, and the like. An in-depth knowledge is not normally necessary, but some familiarity is helpful when trying to adjust the control parameters of the solvers that you may be using.

Moving from *computer implementation* back to the *algorithm, model, or solution technique* is called **verification**. The main idea is to make sure that the computer implementation is actually carrying out the algorithm as it is supposed to. Again, this will not be of great concern for users of well-tested commercial optimizers.

You will be greatly concerned with **validation and sensitivity analysis**, the process of moving between the *algorithm, model, or solution technique* and the *real world problem*. It is here that the loop is completed and you finally compare the results you are obtaining

with the real situation. Are the results appropriate? Do they make sense? Does the model need to be modified, or another solution technique chosen? If so, then the loop begins again.

Validation is the process of making sure that the model or solution technique is appropriate for the real situation. **Sensitivity analysis** looks at the effect of the specific data on the results. For example, consider again how the "profit per item produced" is estimated. Suppose that you are not very confident that the estimate is accurate. Sensitivity analysis asks how sensitive the final results are to variations in data such as the profit per item produced. It may show, for example, that the final solution changes very little even with a large variation in profit per item produced. In this case you heave a sigh of relief. On the other hand if it turns out that the results change dramatically when the estimated profit per item produced changes very slightly, then you have cause for worry, especially if there are millions of dollars depending on the outcome of your analysis! In this case you will need to run a number of scenarios showing how things will turn out at various values of profit per item produced in order to arrive at a final recommendation.

This book concentrates on the top two boxes in Figure 1.1, and on the processes of moving between them. You will be exposed to some of the details of the algorithms and their computer implementations so that you have enough understanding to deal with unexpected outcomes. For success in practical optimization, you will need three things:

- a broad knowledge of the classical generic optimization problems,
- a knowledge of the available solution algorithms and their implementations in commercially available solvers,
- practice in the process of analyzing and solving problems (as in Figure 1.1).

A good approach to acquiring these essential elements is to:

- use this book to learn about the solution algorithms, and for examples of problem formulations,
- obtain practice in the optimization process through homework examples and examinations which consist entirely of applied problems, i.e. purely theoretical exercises of the mechanics of the solution algorithms are to be avoided.

Of course, textbook word problems are vastly simpler than the complexities, irrelevancies, and political difficulties that arise in real situations, but they do provide some practice in analysis and model formulation.

This book is designed as a one-term introduction to the most important topics in applied optimization. It is impossible to cover all of optimization in a one-term course: there are entire yearlong courses on each of the individual topics that we will cover! This book provides a fairly broad survey at a medium depth. There are numerous topics that you should be aware of, but which cannot be covered in an introductory book like this one; brief sketches of these topics appear throughout the book.

The main goals of a course using this book should be to equip students to:

1. recognize problems that can be tackled using the tools of applied optimization,
2. formulate optimization problems correctly and appropriately,
3. solve optimization problems, primarily by selecting and applying the correct solvers, but also possibly by writing special software or hiring experts.

These abilities will be an excellent addition to your skills toolkit, and especially useful as the world becomes more complex and computer-centric. Note that a course in optimization or operations research is required in most MBA courses, in business, industrial engineering (and many other engineering programs), and economics. Such courses are usually a recommended option in computer science as well. These courses are included in all those programs precisely because the material finds so many practical applications.

A brief overview of the main topics covered in the book follows.

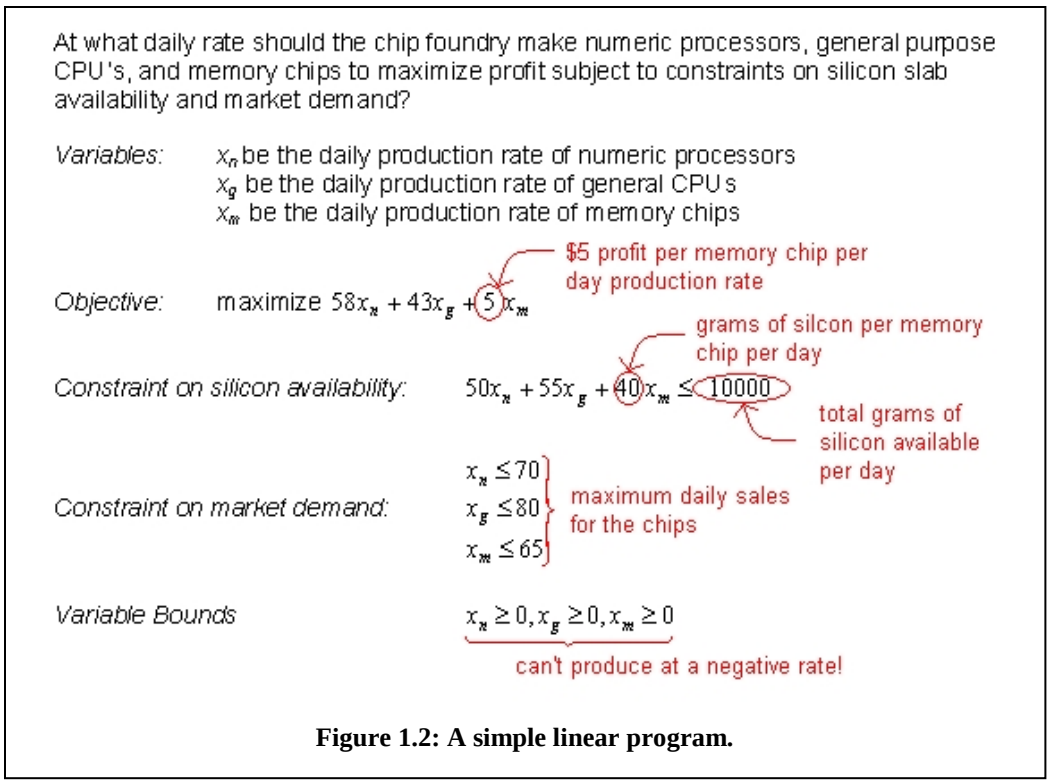

Figure 1.2: A simple linear program.

Linear programming is the most widely used of the major techniques for constrained optimization. "Programming" in this sense does not imply computer programming – it is an old word for "planning". In linear programming, all of the underlying models of the real-world processes are linear, hence you can think of "linear programming" as "planning using linear models".

While linear models (e.g. each refrigerator produced consumes exactly 5 kWh of electricity and 60 kilograms of steel) might seem too simple to model many real world problems, they actually have a vast range of applications. Many production problems are adequately modeled with linear relations, and many routing problems are linear, for

example. The largest practical constrained optimization problems that are regularly solved are linear programs. Airline scheduling problems may have hundreds of thousands of constraints and a million variables, for example.

A portion of a typical linear programming problem is shown in Figure 1.2. Note that all of the relationships, the single objective function and the multiple constraints, are all linear relationships. The problem is solved by determining the values of the variables that maximizes the profit relationship in this case.

Breakthroughs in the solution of linear programs include the introduction of the simplex method in 1947, and Karmarkar's interior point method in 1984. Research on improved methods continues to this day.

Networks are natural models of many real situations. A network model consists of nodes and connecting lines called arcs, as sketched in Figure 1.3. The nodes in Figure 1.3 represent cities in the eastern part of Canada, and the lines between the nodes may variously represent roads or railways or telephone lines, depending on the application.

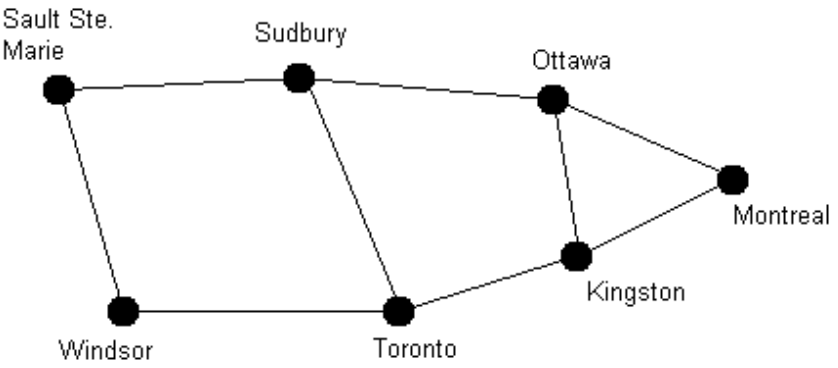

Figure 1.3: A simple network diagram.

Examples of some questions that might be posed, given the network model in Figure 1.3:

- Label each of the arcs with the distance between the cities that it connects. What is the shortest route between Sault Ste. Marie and Montreal? This may be a simple problem to solve in Figure 1.3, but now imagine solving it on a network diagram of a million-transistor layout: brute force enumeration of all of the possible routes simply won't work! This is the classic *shortest route problem*. If the arcs are labeled with travel times, then it is the fastest-route problem.
- Let the arcs represent telephone lines, and label each arc with the maximum capacity of the phone line in terms of number of simultaneous calls. Now how should you route calls from Windsor to Ottawa so as to maximize the number of simultaneous calls that can be handled? Note that calls can go via several routes at once: Windsor-Toronto-Kingston-Ottawa and Windsor-Sault Ste. Marie-Sudbury-Ottawa and Windsor-Toronto-Sudbury-Ottawa, etc. This is the *maximum flow problem*. The related *minimum-cut* problem can be used to answer

questions about where larger-capacity telephone lines should be constructed so that more calls can flow simultaneously between two cities.

- Label the arcs with the cost of constructing a high-speed fiber-optic computer line between cities that it connects. Which of the fiber-optic lines should be constructed so that all of the cities can be connected together at minimum total cost? This is the *minimum spanning tree problem*.

There is a huge array of other problems that can be tackled by network formulations and solution methods.

We will also take a brief look at the PERT methodology. This is a network-based technique for managing complex projects consisting of many activities, some of which have precedence relationships, and some of which can run in parallel. Typical questions include: What is the shortest time in which the project can be completed? How do limited resources affect the completion time? PERT techniques have many applications beyond project management, for example in computer chip design where there are questions of signal propagation time through a network of transistors.

In linear programming, all of the variables are real-valued, that is they can take on fractional values such as 3.7 or 19.331. In **integer programming** the variables are restricted to taking on integer values, or perhaps only binary (0 or 1) values. Many problems are integer, for example:

- Given the members of a swim team and knowledge of their typical times for various strokes, how do you assign the members of the team to the legs of a multi-stroke relay to achieve the fastest total time? This is a binary problem: you can either *assign* someone to a leg of the relay or not assign them; you can't assign half of each of two people!
- How do you *schedule* professors, rooms, courses and students so that there are no conflicts (or a minimum number of conflicts)? This is again a binary problem: you can't assign a professor to be in two places at once.
- How do you determine the number of rail cars to assign to various cargoes? This is an integer problem: each rail car either goes or stays.

In practical problems it is generally impossible to enumerate every possible solution and just take the best of the lot. Integer programming problems experience "combinatorial explosion" in which the number of solutions grows extremely rapidly as the number of elements in the problem increases because of the many, many possible ways of combining the elements. For example it is very easy to construct relatively small integer programming problems for which the number of possible different solutions is greater than the number of atoms in the known universe. That is *way* too many to enumerate. Adding to the difficulty, it is rarely possible to apply real-valued techniques such as linear programming and then simply round the results to the nearest integer.

We will look at the branch and bound technique for structuring the solution to slow down the combinatorial explosion, and at genetic algorithms for cases in which the combinatorial explosion defeats even branch and bound.

Dynamic programming is used when making a sequence of interrelated decisions. It is used in *equipment replacement* problems for example, where you need to keep a functioning heart monitor in place over a number of years, and which to plan the sequence of repairs and replacements that minimize the total cost of the horizon under consideration.

Nonlinear programming is identical to linear programming, except that the relationships can have a nonlinear form. This makes the problem *much* harder than linear programming, for reasons that we will explore in detail later. However nonlinearity is inescapable in many real-world problems. Nonlinear relationships may, for example, be required to accurately model the losses in radio signal strength with distance as you optimize a microwave communication system. We will explore some of the simpler techniques of nonlinear programming, those that are most often implemented in commercial solvers. Fortunately, while there is a huge and rapidly growing research literature on the mathematics of nonlinear programming, only a relatively small number of techniques are actually available in commercial solvers.

These are the main topics covered in this introductory textbook, but there is a wealth of other techniques available. I hope I've whetted your appetite.