AIMMS Modeling Guide - Formulating Optimization Models

This file contains only one chapter of the book. For a free download of the complete book in pdf format, please visit www.aimms.com.

AIMMS B.V.
Diakenhuisweg 29-35
2033 AP Haarlem
The Netherlands
Tel.: +31 23 5511512

AIMMS Inc.
11711 SE 8th Street
Suite 303
Bellevue, WA 98005
USA
Tel.: +1 425 458 4024

AIMMS Pte. Ltd.
55 Market Street #10-00
Singapore 048941
Tel.: +65 6521 2827

AIMMS
SOHO Fuxing Plaza No.388
Building D-71, Level 3
Madang Road, Huangpu District
Shanghai 200025
China
Tel.: ++86 21 5309 8733

Email: info@aimms.com
WWW: www.aimms.com

Chapter 2

Formulating Optimization Models

This chapter

This chapter explains the general structure of optimization models, and some characteristics of their solution. In Chapter 1, an introduction to optimization models was given. In this chapter, optimization models are introduced at a more technical level.

Three classes of constrained optimization models

The three main classes of constrained optimization models are known as *linear*, *integer*, and *nonlinear programming models*. These types have much in common. They share the same general structure of optimization with restrictions. Linear programming is the simplest of the three. As the name indicates, a linear programming model only consists of linear expressions. Initially, linear programming will be explained, followed by integer and nonlinear programming.

The term programming

The term *programming*, as used here, does not denote a particular type of computer programming, but is synonymous with the word *planning*. The three classes of programming models mentioned above all come under the heading of *mathematical programming models*.

2.1 Formulating linear programming models

Wide applicability

Linear programming was developed at the beginning of the mathematical programming era, and is still the most widely used type of constrained optimization model. This is due to the existence of extensive theory, the availability of efficient solution methods, and the applicability of linear programming to many practical problems.

2.1.1 Introduction

Linear equations and inequalities

Basic building blocks of linear programming models are *linear equations* and *linear inequalities* in one or more unknowns. These used in linear programming models to describe a great number of practical applications. An example of a linear equation is:

$$2x + 3y = 8$$

By changing the "=" sign to a "≥" or "≤", this equation becomes a linear inequality, for instance:

$$2x + 3y \geq 8$$

The signs "<" and ">", denoting strict inequalities, are not used in linear programming models. The *linearity* of these equations and inequalities is characterized by the restriction of employing only "+" and "−" operations on the terms (where a term is defined as a coefficient times a variable) and no power terms.

Variables

The unknowns are referred to as *variables*. In the example above the variables are x and y. A solution of a linear programming model consists of a set of values for the variables, consistent with the linear inequalities and/or equations. Possible solutions for the linear equation above are, among others: $(x, y) = (1, 2)$ and $(4, 0)$.

2.1.2 Example of a linear programming model

Production of potato chips

To illustrate a linear programming model, the production of chips by a small company will be studied. The company produces plain and Mexican chips which have different shapes. Both kinds of potato chips must go through three main processes, namely slicing, frying, and packing. These processes have the following time characteristics:

- Mexican chips are sliced with a serrate knife, which takes more time than slicing plain chips.
- Frying Mexican chips also takes more time than frying plain chips because of their shape.
- The packing process is faster for Mexican chips because these are only sold in one kind of bag, while plain chips are sold in both family-bags and smaller ones.

There is a limit on the amount of time available for each process because the necessary equipment is also used for other purposes.The chips also have different contributions to net profit.

Data

The data is specified in Table 2.1. In this simplified example it is assumed that the market can absorb all the chips at the fixed price.
The planner of the company now has to determine a production plan that yields maximum net profit, while not violating the constraints described above.

time [min/kg] required for:	**plain chips**	**Mexican chips**	**availability [min]**
slicing	2	4	345
frying	4	5	480
packing	4	2	330
net profit contribution [$/kg]	2	1.5	

Table 2.1: Data in the potato chips problem

Decision variables

The planner's decision problem can be formulated in terms of a mathematical notation using linear inequalities. The variables in the inequalities must reflect what is unknown to the planner, namely his decisions. In this example, the decision variables concern a production plan. The quantity of plain and Mexican chips to be produced are unknown to the planner. Therefore, the variables are the *amounts of both types of chips to be produced*.

Variable names

In order to obtain a concise mathematical description it is convenient to choose short names for the variables. Let X_p therefore denote the unknown amount of plain chips to be produced, and let X_m denote the unknown quantity of Mexican chips to be produced. X_p and X_m are both measured in kilograms. Inequalities that reflect the availability of the production processes can now be stated.

The constraint on frying

The following inequality, measured in minutes, can be written to describe the limited availability of the fryer:

$$4X_p + 5X_m \leq 480 \quad \text{[min]}$$

In words this inequality states:

The four minutes required to fry a kilogram of plain chips
multiplied by the planned number of kilograms of plain chips
plus
the five minutes required to fry a kilogram of Mexican chips
multiplied by the planned number of kilograms of Mexican chips
must be less than or equal to
the 480 minutes the fryer is available.

Or, a bit shorter:

The time required to fry the plain chips
plus
the time required to fry the Mexican chips
must be less than or equal to
the time the fryer is available.

So now there is an inequality that describes the limited availability of the fryer.

Units of measure

An easy check to see whether the meaning of an inequality makes sense is to write it in terms of units of measure. This yields:

$$4[\text{min/kg}]X_p[\text{kg}] + 5[\text{min/kg}]X_m[\text{kg}] \leq 480[\text{min}]$$

The resulting units for each term should be identical, which they are in this case (minutes).

Other constraints

Similar inequalities can also be written for the availabilities of the slicer and the packer:

$$2X_p + 4X_m \leq 345 \quad [\text{min}]$$
$$4X_p + 2X_m \leq 330 \quad [\text{min}]$$

Together these inequalities almost give a complete description of the situation. One set of inequalities is still missing. Obviously, it is not possible to produce a negative amount of chips. So, the following lower bounds on the variables must be added for a complete description of the problem:

$$X_p \geq 0, \quad X_m \geq 0 \quad [\text{kg}]$$

These last inequalities are referred to as *nonnegativity constraints*.

Optimal decisions

The company's planner has to make a choice. From these possible production options, he wants to choose the plan that yields the maximum net profit. By maximizing profit, the number of plans is reduced to those that are preferred. The following linear equation gives the net profit:

$$P = 2X_p + 1.5X_m \quad [\$]$$

The quantity P can be regarded as an additional variable, for which the maximum value is to be found. The value of P depends on the value of the other variables.

Verbal summary

The decision problem has just been posed as a mathematical problem instead of a verbal problem. In order to show similarities and differences between them, both a verbal description of the problem and the mathematical model are given. The *verbal* description of the decision problem is:

Maximize: *Net profit,*
Subject to:

- *a time restriction on slicing,*
- *a time restriction on frying,*
- *a time restriction on packing, and*
- *negative amounts cannot be produced.*

Mathematical summary

The *mathematical* model is formulated as follows:

$$\begin{aligned} &\textbf{Maximize:} && P = 2X_p + 1.5X_m \\ &\textbf{Subject to:} \\ & && 2X_p + 4X_m \leq 345 && \text{(slicing)} \\ & && 4X_p + 5X_m \leq 480 && \text{(frying)} \\ & && 4X_p + 2X_m \leq 330 && \text{(packing)} \\ & && X_p, X_m \geq 0 \end{aligned}$$

2.1.3 Picturing the formulation and the solution

Picturing the decision problem

In this small problem the inequalities can be visualized in a two-dimensional drawing. Where the x-axis and the y-axis represent X_p and X_m respectively, the inequalities and their implications can be plotted. The easiest way to plot the slicer availability inequality is as follows.

- First change the "≤" sign to an "=" and plot the border.

Setting the value of X_p to 0, then the value of X_m can be calculated: $X_m = 345/4 = 86.25$. In the same way the value of X_p is calculated as 172.5 when X_m is set to zero.

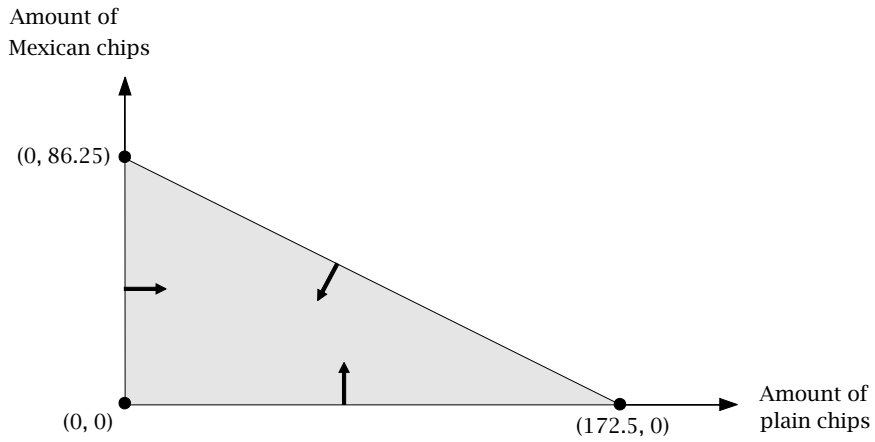

Figure 2.1: The constraint on slicing visualized

So far, two points have been found in the X_p-X_m plane, namely $(X_p, X_m) = (0, 86.25)$ and $(X_p, X_m) = (172.5, 0)$. The line that connects these points is the line

$$2X_p + 4X_m = 345$$

which is plotted.

- Second, determine whether a single point at one side of the line, such as the origin, satisfies the constraint. If it does, shade that side of the line.

The shaded region in Figure 2.1 contains all (X_p, X_m) that satisfy the constraints:

$$2X_p + 4X_m \leq 345, \quad X_p \geq 0 \text{ and } X_m \geq 0$$

In other words, the shaded region contains all combinations of the quantities of the two types of chips that can be sliced in one day.

Picturing the feasible region

Other inequalities can also be represented graphically, as shown in Figure 2.2. From this figure, it is clear that there are many combinations of production levels for plain and Mexican chips that satisfy all these inequalities. The shaded region bounded by the lines corresponding to the inequalities represents all the allowed production levels, and is called the *feasible region*.

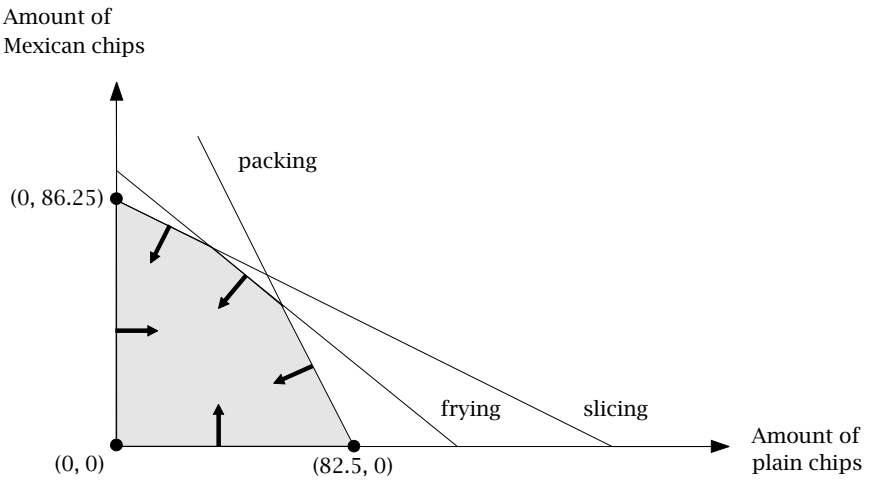

Figure 2.2: The feasible region

Picturing the profit function

When a value, of say \$150, is chosen for the variable P, a line can be drawn that represents all combinations of production levels for Mexican and plain chips that yield a profit of \$150. Such a line is called a *contour* of the profit function, and is drawn in Figure 2.3. The arrow indicates the direction of increasing profit. Since the profit function is linear, the contours are straight lines.

Picturing the optimal decision

But how can one determine which combination yields the *maximum* net profit? Observe that a higher value for P yields a contour parallel to the previous one. Moreover, increasing the value of P causes the line to shift to the right. This is also illustrated in Figure 2.3. However, the profit cannot increase indefinitely, because the profit line will fall outside the feasible region if it shifts too far to the right. In fact, it is clear from the application that the profit cannot increase indefinitely because of the limitations imposed by the availability of the slicer, fryer, and packer.

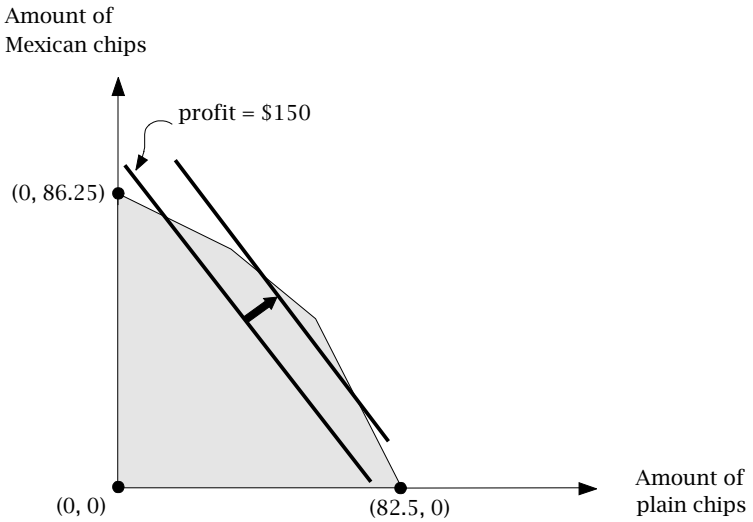

Figure 2.3: Different profit lines

The best solution attainable is when the profit line is shifted *as far to the right as possible* while still touching the feasible region. *Best solution*

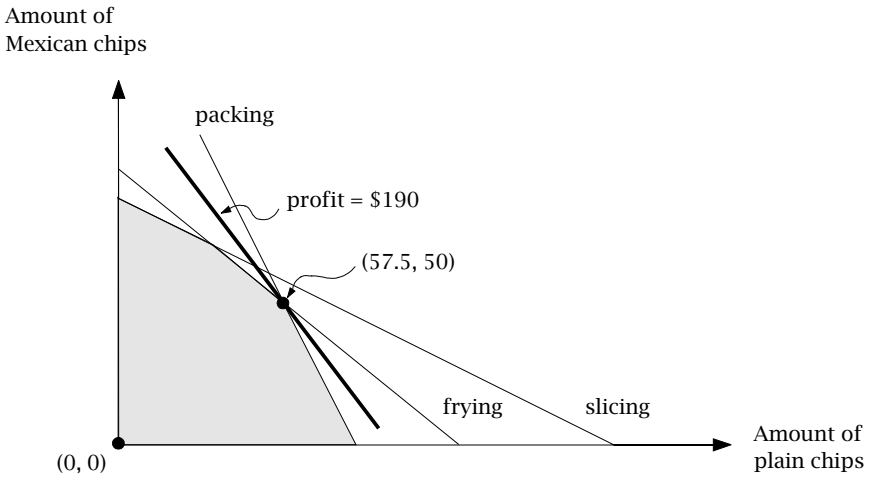

Figure 2.4: The optimal solution

From Figure 2.4, it can be seen that this point is the intersection of the lines corresponding to the frying and packing restrictions. The coordinates of this point can now be calculated by solving a system of two equations in two unknowns—the frying and packing restrictions as equalities:

$$4X_p + 5X_m = 480 \quad \text{(frying)}$$
$$4X_p + 2X_m = 330 \quad \text{(packing)}$$

The above system yields $(X_p, X_m) = (57.5, 50)$ and the corresponding profit is \$190. This combination of values for the decision variables is called the *optimal solution*.

Non-binding constraints

Considering Figure 2.4 once again, it can be observed that only two constraints really restrict the optimal solution. Only the constraints on frying and packing are *binding*. The constraint on slicing can be omitted without changing the optimal solution. Such a constraint is known as a *non-binding* constraint. Although non-binding constraints can be removed from the model without consequences, it is often sensible to include them anyway. A non-binding constraint could become binding as data change, or when experiments are carried out using the model. Moreover, when you build a model, you probably will not know in advance which constraints are non-binding. It is therefore better to include all known constraints.

Corner solutions

Considering Figure 2.4 once again, one can see that the optimal solution is on a corner of the feasible region, namely, the intersection of two lines. This implies that the exact value can be calculated by solving a system of two equations and two unknowns. In general, it can be stated that if a linear programming model has an optimal solution, then there is always an optimal corner solution. This is illustrated in Figure 2.5. Depending on the slope of the objective function, the solution is either at A, B, or C.

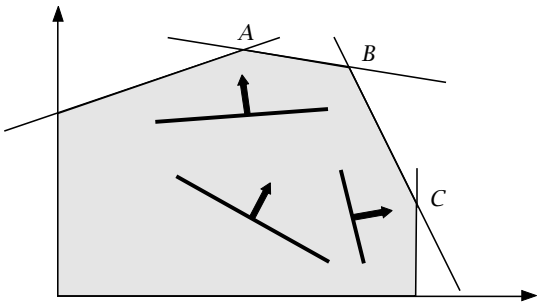

Figure 2.5: There is always an optimal corner solution

Multiple optima

A special case occurs when the slope of the objective is parallel to the slope of one of the binding constraints, as in Figure 2.6. Then there are two optimal corner solutions and an infinite number of optimal solutions along the line segment connecting these two corner solutions. This is a case of so-called *multiple* or *alternative optima*.

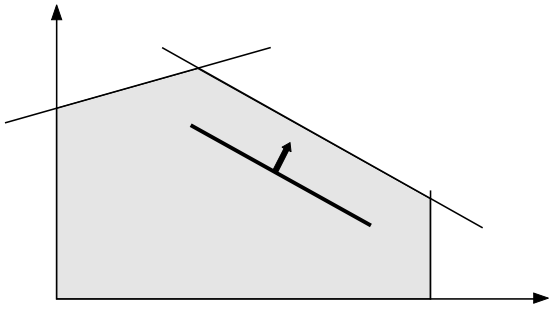

Figure 2.6: Multiple optima

Limitations of pictures

The graphical solution method so far used in this section is only suitable for problems with two decision variables. When there are three decision variables, a similar three-dimensional figure evolves, but this is a lot harder to draw and interpret. Figures with more than three dimensions, corresponding to the number of decision variables, cannot be drawn. The optimal solution must then be expressed algebraically, and solved numerically. The graphical solution method is only used for purposes of illustration.

Computer solutions

The method most often used to calculate an optimal solution is the so-called *simplex method*, developed by George Dantzig ([**?**]). This method examines the corners of the feasible region in a structured sequence, and stops when the optimal solution has been found. For very small models, these calculations could be done manually, but this is both time consuming and prone to errors. In general, these calculations are best done by a computer, using a sophisticated implementation of the simplex method. Almost without exception, such an algorithm finds an optimal solution, or concludes that no such solution exists.

Optimality

When an optimal solution is found by a solver, caution is still needed. Since a model is a simplified picture of the real problem, there may be aspects that are neglected by the model but still influence the *practical* optimality of the solution. Moreover, for most situations there is no single, excellent from all aspects, optimal solution, but a few different ones, each one optimal in its own way. Therefore, the practical interpretation of the model results should always be considered carefully. Experiments can be done using different objectives. In Chapter 4, an introduction is given to *sensitivity analysis*—solution changes due to changes in the data.

Infeasibility

When a problem is declared to be *infeasible* by a solver, it means that the feasible region is empty. In other words, there is no solution that satisfies all the constraints simultaneously. This is illustrated in Figure 2.7. *Infeasibility* can be caused by having too many or conflicting requirements, or by errors in data specification, or by errors in the construction of model equations. Such a result is an incentive to check the correctness of the model. Sometimes an infeasible solution is practically acceptable because the constraints that are violated are not so critical.

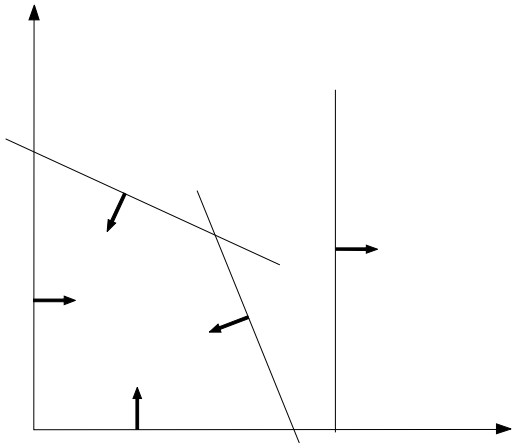

Figure 2.7: Infeasibility illustrated

Unboundedness

Unboundedness is just what the word implies; the constraints fail to bound the feasible region in the direction in which the objective is optimized. As a result, the value of the objective function increases or decreases indefinitely, which might look attractive, but is certainly not realistic. In Figure 2.8, a graphical illustration is given. Unboundedness is not a problem occurring in reality but a formulation error. Common errors causing unboundedness include the following. It could be that a free variable should be nonnegative, or that the direction of optimization has been inadvertently reversed. Alternatively, a constraint may have been omitted. Note that whether an unbounded constraint set causes difficulties depends on the objective function. In Figure 2.9 an example is given in which an optimal solution does exist, although the feasible region is unbounded.

References

Readers who want to know more about the theoretical aspects of linear programming, or the simplex method, can refer to [?] or [?], and there are many other excellent works on the subject.

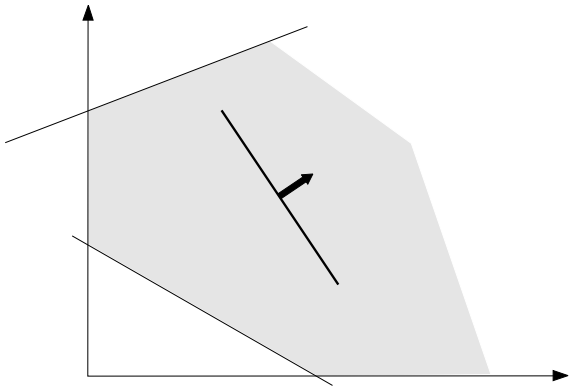

Figure 2.8: Unbounded objective, unbounded feasible region

2.2 Formulating mixed integer programming models

This section

In this section some basic characteristics of mixed integer programming models and their solution will be discussed.

Need for integer solutions ...

Recall the example of the chips-producing company in the previous section where the optimal solution was $(X_p, X_m) = (57.5, 50)$. If chips are packed in bags of 1 kilogram, this optimal solution cannot be translated into a practical solution. One approach is to round down the solution. In this example, rounding yields the feasible solution $(X_p, X_m) = (57, 50)$, but the profit is reduced from \$190 to \$189.

... illustrated

There are many real-world problems that require their solutions to have integer values. Examples are problems that involve equipment utilization, setup costs, batch sizes, and "yes-no" decisions. Fractional solutions of these problems do not make real-world sense; just imagine constructing half a building.

Pros and cons of rounding ...

Rounding is a straightforward way to obtain *integer values*. There might be doubt, however, whether such a solution is optimal or even feasible. In practice, rounding is a satisfactory approach when:

- the figures to be rounded are so large that the error is negligible,
- the input data are not known with certainty, so that a fractional solution is not that accurate anyway,
- the rounding procedure is certain to give a feasible solution, and
- algorithms developed especially to search for an integer solution are too expensive in terms of computer resources and time.

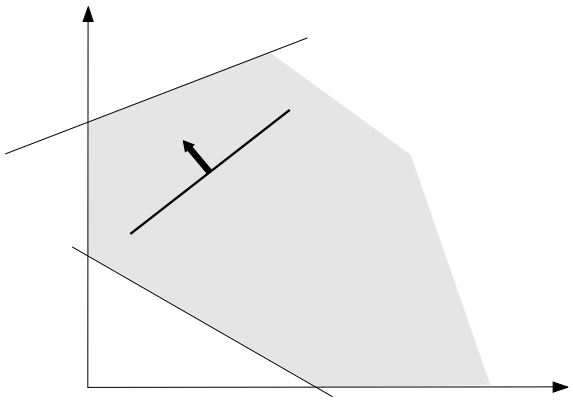

Figure 2.9: Bounded objective, unbounded feasible region

... an alternative

The alternative to rounding has already been mentioned—making use of an *integer programming algorithm*. Next, the potato chips example will be used to present some aspects of integer programming. When an integer programming algorithm is used in the example above, the solution $(X_p, X_m) = (58, 49)$ yields a higher profit, \$189.5.

Extending the potato chips example

Consider again the chips-producing company. Their special chips are becoming so popular that a large supermarket chain wants to sell these potato chips in their stores. The additional restriction is that the chips must be delivered in batches of 30 kg, for efficiency reasons. As a result, the optimal production plan, determined above, is no longer adequate. The optimal amounts of 57.5 kg of plain and 50 kg of Mexican chips would make up, respectively, 1.92 and 1.67 batches of potato chips. This solution does not satisfy the additional restriction imposed by the supermarket chain. An integer programming model could now be used to determine a new (and acceptable) production plan.

Description

The linear programming model of Section 2 can be reformulated as an integer programming model by measuring the amounts of chips in batches, and by adding the requirement that the solution takes on integer values. When measuring the amounts of chips in batches of 30 kg, the production processes require the following amounts of time. Each batch of plain chips takes 60 minutes to slice, 120 minutes to fry, and 120 minutes to pack. Each batch of Mexican chips takes 120 minutes to slice, 150 minutes to fry, and 60 minutes to pack. Furthermore, the net profit on a batch of plain chips is \$60, while the net profit on a batch of Mexican chips is \$45.

Formulation

Let X_p^B denote the number of batches of plain chips produced, and let X_m^B denote the number of batches of Mexican chips produced. Note that the restriction is added that X_p^B and X_m^B must have integer values (fractions of batches are not allowed). Then the integer programming model becomes:

$$\begin{aligned}
&\textbf{Maximize:} && 60X_p^B + 45X_m^B = P \\
&\textbf{Subject to:} \\
&&& 60X_p^B + 120X_m^B \le 345 && \text{(slicing)} \\
&&& 120X_p^B + 150X_m^B \le 480 && \text{(frying)} \\
&&& 120X_p^B + 60X_m^B \le 330 && \text{(packing)} \\
&&& X_p^B, X_m^B \ge 0 \\
&&& X_p^B, X_m^B \textit{ integers}
\end{aligned}$$

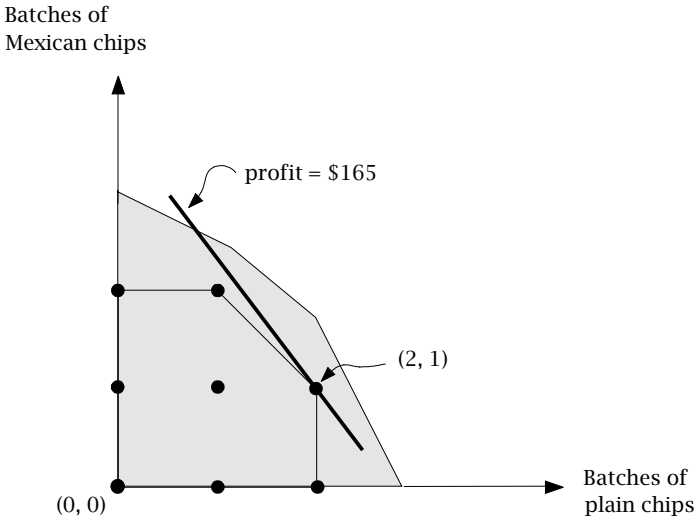

Figure 2.10: The feasible region in the integer program

Picturing the solution

Figure 2.10 is almost the same as Figure 2.4, except that the feasible region is now limited to the grid points within the region bounded by the constraints. It is clear that the grid point that maximizes the profit, subject to the constraints, is $(X_p^B, X_m^B) = (2, 1)$. This implies a production plan with 60 kilograms of plain chips, 30 kilograms of Mexican chips, and a profit of $165. Note that this solution is not a corner or on the boundary of the region determined by the frying, slicing, and packing constraints. These constraints do limit the feasible region, however, because otherwise a higher profit would be possible. Notice also that the profit has dropped from $190 to $165. The company might consider making a weekly production plan, and delivering the batches once a week. In this way the production of a part of a batch in a day would be allowed, and this would increase the average daily profit. From this last comment, it is

clear that formulating a model involves more than merely formulating a set of constraints and an objective function.

Pure and mixed integer programming

Constrained optimization problems in which all variables must take on integer values are referred to as *pure integer programming* problems. Constrained optimization problems, in which only some of the variables must take on integer values, are referred to as *mixed integer programming* problems. If both the objective and all constraints are linear, then the term *(mixed) integer linear programming* applies.

Zero-one programming

A *zero-one programming* problem is a pure integer programming problem with the additional restraint that all variables are either zero or one. Such zero-one variables are referred to as *binary variables*. Binary variables provide a variety of new possibilities for formulating logical conditions and yes/no decisions in integer programming models. The use of binary variables is illustrated in Chapters 7 and 9.

Solvability

For a model builder it is a straightforward matter to add the requirement of integrality to a mathematical problem description. However, solving an integer programming problem is almost always harder than solving the underlying linear program. Most of the solution algorithms available do, in fact, *test* all promising integer solutions until the best (or a good) one is found. A sequence of linear programs is solved in order to obtain a solution of an integer program. The most widely used solution algorithms work on the so-called *branch and bound method*, in which a tree structure is used to conduct the search systematically. The branch and bound method, as well as some alternative algorithms, is explained in many textbooks on integer programming, for example [**?**].

Relaxed LP models

In integer programming, the way a model is formulated can influence its solvability. If you have trouble finding an integer solution, a good approach is to first drop the integrality requirement and solve the model as an LP (this LP model is referred to as the *relaxed (LP) model* of the integer programming model). Then reformulate the model until the optimal LP solution value is close to the integer one. You can then apply an integer programming solution algorithm (see [**?**] for examples). One method of speeding up the solution process is to bound all variables (upper and lower bounds) as tight as possible, so that the number of possible integer solutions is reduced.

Optimality tolerance

In the previous paragraph, the remark was made that in *integer programming*, searching for an optimal solution might take too much time, because a sequence of linear programs has to be solved in order to obtain an integer solution. A widespread strategy is to specify a measure of the maximum (absolute or relative) deviation between the objective value of the integer solution and

the optimal objective value of the relaxed (LP) problem. As soon as a solution is within this measure, the algorithm terminates. The solution thus found can be regarded as a *compromise* between optimality and practical feasibility.

Infeasibility

Notice that a feasible linear program may become infeasible as soon as the requirement of integrality is added. For example, picture a feasible region which is not empty, but excludes all grid points.

Unboundedness

To be able to solve a mixed integer models, integer variables should only be able to take a finite number of values. For this reason, pure integer models (i.e. models without any continuous variable) are never unbounded.

Integer solutions from linear programs

Besides difficult integer programming problems, there exist also easy ones. These are problems that can be formulated as *network flow problems*. These problems can be solved as linear programs, and the optimal solution is guaranteed to be integer. This is due to a special mathematical property of network flow problems (i.e. *totally unimodularity* of the constraint matrix). It is unfortunate that this property can easily disappear as a result of small model extensions, so the number of easy-to-solve integer programming problems encountered in practice is relatively small. Network flow problems are discussed in more detail in Chapter 5.

2.3 Formulating nonlinear programming models

This section

In this section some basic characteristics of the third type of constrained models, namely nonlinear programming models, will be discussed.

Nonlinear expressions

Besides adding integrality constraints to a linear programming model, another major extension can be made by relaxing the convention that all expressions must be linear. Since many physical and economic phenomena in the world surrounding us are highly nonlinear, *nonlinear programming* models are sometimes required rather than linear ones.

Extending the potato chips example

Competition is growing. The chips-producing company decides to change its selling prices, which seem to be too high. The company decides to maximize the production of potato chips (which forms the supply), by setting the prices so that supply and demand are equal. Models in which supply and demand are equated are widely used in economics, and are known as *equilibrium models*. [**?**] gives an introduction. The company's marketing manager has determined the relationship between demand (D) and price (P) to be the following identities.

- *Demand for plain chips:* $D_p = -66.7P_p + 300$
- *Demand for Mexican chips:* $D_m = -83.4P_m + 416.6$

Furthermore, the fixed production costs are \$2 per kg for both chip types.

Formulation

The constraints on the limited availability of the slicer, fryer, and packer have not changed. The objective, however, has. The net profit now depends on the prices that the company sets. By definition, the profit equals revenue minus costs, which can be written as

$$P = (P_p - 2)X_p + (P_m - 2)X_m$$

Since the company wants to set the price in such a way that supply equals demand, the demand curves can be used to determine that price.

- *Supply equals demand for plain chips:*

$$X_p = D_p = -66.7P_p + 300$$

- *Supply equals demand for Mexican chips:*

$$X_m = D_m = -83.4P_m + 416.6$$

Which can also be written as

$$P_p = -0.015X_p + 4.5$$
$$P_m = -0.012X_m + 5.0$$

where D_p and D_m have been eliminated , and P_p and P_m have been solved for. These prices can now be used in the objective function, which becomes a nonlinear function depending only on X_p and X_m:

$$P = (-0.015X_p + 4.5 - 2)X_p + (-0.012X_m + 5 - 2)X_m$$

or

$$P = -0.015X_p^2 + 2.5X_p - 0.012X_m^2 + 3X_m$$

Algebraic description

A new (nonlinear) objective function has been derived, and the model can now be stated completely.

Maximize: $-0.015X_p^2 + 2.5X_p - 0.012X_m^2 + 3X_m = P$

Subject to:

$$2X_p + 4X_m \leq 345 \quad \text{(slicing)}$$
$$4X_p + 5X_m \leq 480 \quad \text{(frying)}$$
$$4X_p + 2X_m \leq 330 \quad \text{(packing)}$$
$$X_p, X_m \geq 0$$

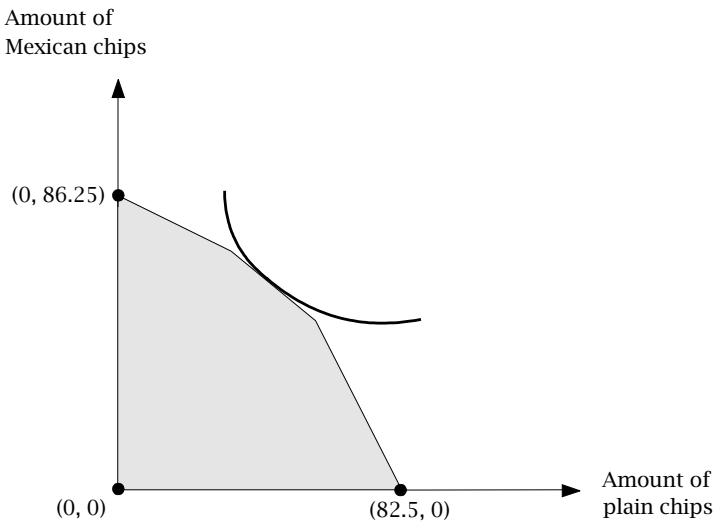

Figure 2.11: The feasible region in the nonlinear programming model

Picturing the solution

When the contour of the objective and the area of the feasible region are plotted, the same pictorial solution procedure can be followed as in the previous examples. Figure 2.11 shows this plot. One difference from previous examples is that the contour of the profit function is no longer linear. This contour can still be shifted to the right as long as it touches the feasible region. From the figure the optimal point can roughly be determined: about 40 kg of plain chips, and about 60 kg of Mexican chips. Note that this optimal point is not on a corner. Using AIMMS to determine the exact solution gives 42.84 kg of plain chips, and 61.73 kg of Mexican chips, which yields a profit of $219.03.

Terminology

A nonlinear programming problem consists of an algebraic objective function subject to a set of algebraic constraints and simple bounds. The term *algebraic* is used here to indicate that algebraic operators for addition, subtraction, division, multiplication, exponentiation, etc. are applied to the variables. Differential and integral operators are not considered. The algebraic constraints consist of equations and/or inequalities. The *simple bounds* are lower and/or upper bounds on the variables. The variables themselves can take on any real value between these bounds. If no simple bounds are stated for a particular variable, then its value may vary between minus infinity and plus infinity. Nonlinear programming models are often referred to as NLP models, and their objective contours and constraint boundaries need no longer be straight lines. The combinations of curved contours and boundaries can make them very difficult to solve.

Local and global optima

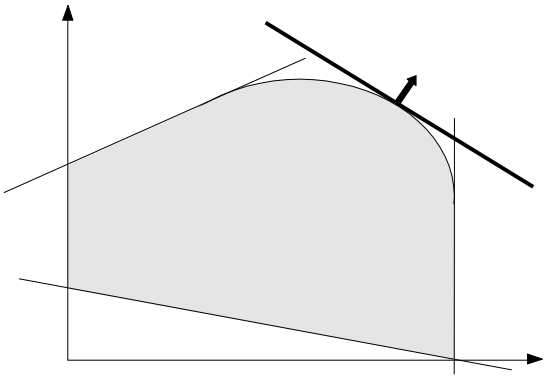

Figure 2.12: A non-corner solution of a nonlinear program

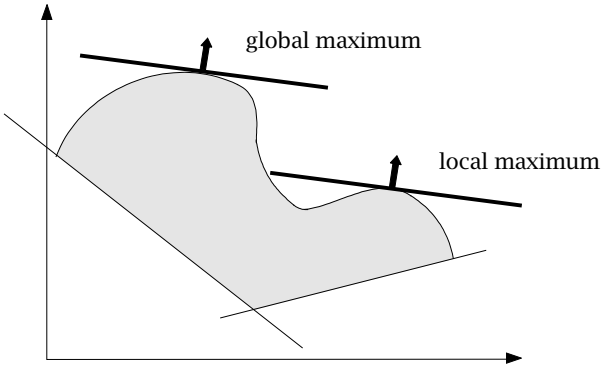

Figure 2.13: Local and global optima illustrated

Figures 2.11 and 2.12 illustrate that the optimal solution of a nonlinear programming model need not be on a corner. Furthermore if a solution is found, it may only be optimal with respect to the points in a small neighborhood, while a better optimum exists further away. An example of this is given in Figure 2.13. The terms *globally* and *locally optimal* are used to make the distinction. Only in the specific situation where the nonlinear programming problem has only one optimal solution can one ignore this distinction. The theoretical conditions under which a problem only has one optimal solution are not easy to verify for most real problems. As a result caution on the part of a model builder is needed. For a theoretical treatment of nonlinear programming, see [**?**].

Solvability

Even though it is not too difficult to construct a large variety of nonlinear programming problems, solving them is a different matter. The reasons for this are that the theory of nonlinear programming is much less developed than the theory of linear programming, and because the solution algorithms are not always capable of solving the nonlinear programming problems presented to them. It should be noted, however, that problems in which the nonlinear terms appear only in the objective function are generally easier to solve than those in which nonlinear terms occur in the constraints. Nonlinear programming models with integer variables are even more difficult to solve and will not be addressed in this book.

2.4 Summary

In this chapter, the three most common types of constrained have been introduced: linear programming models, integer linear programming models and nonlinear programming models. *Linear programming (LP) models* must satisfy the restriction that both the objective function and the constraints are linear. Despite this restriction, linear programming models are still the most common due to the availability of powerful solution algorithms. *Integer linear programming (IP) models* are like linear ones, except that one or more of the variables must take on integer values. Except for a small group, namely network flow models, integer programming models are harder to solve than linear programs. *Nonlinear programming (NLP) models* can have a nonlinear objective and/or nonlinear constraints. An optimal solution is not guaranteed to be globally optimal. When an optimization model is solved, one of the following situations will occur: an optimal solution is found, which is the desired situation; the problem is infeasible, which is a result of inconsistencies in the constraint set; the problem is unbounded, which is often caused by a modeling error.