8. More Applications

Here we have collected several applications of linear programming, and in particular, of the duality theorem. They are slightly more advanced than those in Chapters 2 and 3, but we have tried to keep everything very concrete and as elementary as possible, and we hope that even a mathematically inexperienced reader will have no problems enjoying these small gems.

8.1 Zero-Sum Games

The Colonel Blotto game. Colonel Blotto and his opponent are preparing for a battle over three mountain passes. Each of them commands five regiments. The one who sends more regiments to a pass occupies it, but when the same number of regiments meet, there will be a draw. Finally, the one who occupies more passes than the other wins the battle, with a draw occurring if both occupy the same number of passes.

Given that all three passes have very similar characteristics, the strategies independently pursued by both Colonel Blotto and his opponent are the following: First they partition their five regiments into three groups. For example, the partition $(0, 1, 4)$ means that one pass will be attacked by 4 regiments, another pass by 1 regiment, and one pass will not be attacked at all. Then, the groups are assigned to the passes randomly; that is, each of the $3! = 6$ possible assignments of groups to passes is equally likely.

The partitions of Colonel Blotto and his opponent determine *winning probabilities* for both of them (in general, these do not add up to one because of possible draws). Both Colonel Blotto and his opponent want to bias the difference of these probabilities in their direction as much as possible. How should they choose their partitions?

This is an instance of a finite two-player **zero-sum game**. In such a game, each of the two players has a finite set of possible strategies (in our case, the partitions), and each pair of opposing strategies leads to a *payoff* known to both players. In our case, we define the payoff as Colonel Blotto's winning probability minus the opponent's winning probability. Whatever one of the players wins, the other player loses, and this explains the term zero-sum game. To some extent, it has become a part of common vocabulary.

When we number the strategies $1, 2, \ldots, m$ for the first player and $1, 2, \ldots, n$ for the second player, the payoffs can be recorded in the form of an $m \times n$ **payoff matrix**. In the Colonel Blotto game, the payoff matrix looks as follows, with the rows corresponding to the strategies of Colonel Blotto and the columns to the strategies of the opponent.

	$(0,0,5)$	$(0,1,4)$	$(0,2,3)$	$(1,1,3)$	$(1,2,2)$
$(0,0,5)$	0	$-\frac{1}{3}$	$-\frac{1}{3}$	-1	-1
$(0,1,4)$	$\frac{1}{3}$	0	0	$-\frac{1}{3}$	$-\frac{2}{3}$
$(0,2,3)$	$\frac{1}{3}$	0	0	0	$\frac{1}{3}$
$(1,1,3)$	1	$\frac{1}{3}$	0	0	$-\frac{1}{3}$
$(1,2,2)$	1	$\frac{2}{3}$	$-\frac{1}{3}$	$\frac{1}{3}$	0

For example, when Colonel Blotto chooses $(0, 1, 4)$ and his opponent chooses $(0, 0, 5)$, then Colonel Blotto wins (actually, without fighting) if and only if his two nonempty groups arrive at the two passes left unattended by his opponent. The probability for this to happen is $\frac{1}{3}$. With probability $\frac{2}{3}$, there will be a draw, so the difference of the winning probabilities is $\frac{1}{3} - 0 = \frac{1}{3}$.

Not knowing what the opponent is going to do, Colonel Blotto might want to choose a strategy that guarantees the highest payoff in the *worst case*. The only candidate for such a strategy is $(0, 2, 3)$: No matter what the opponent does, Colonel Blotto will get a payoff of at least 0 with this strategy, while all other strategies lead to negative payoff in the worst case. (Anticipating that a spy of the opponent might find out about his plans, he must reckon that the worst case will actually happen. The whole game is not a particularly cheerful matter anyway.) In terms of the payoff matrix, Colonel Blotto looks at the minimum in each row, and he chooses a row where this minimum is the largest possible.

Similarly, the opponent wants to choose a strategy that guarantees the lowest payoff (for Colonel Blotto) in the worst case. It turns out that $(0, 2, 3)$ is also the unique such choice for the opponent, because it guarantees that Colonel Blotto will receive payoff at most 0, while all other strategies allow him to achieve a positive payoff if he happens to guess or spy out the opponent's strategy. In terms of the payoff matrix, the opponent looks at the maximum in each column, and he chooses a column where this maximum is the smallest possible.

We note that if both Colonel Blotto and his opponent play the strategies selected as above, they both see their worst expectations come true, exactly those on which they pessimistically based their choice of strategy. Seeing the worst case happen might shatter hopes for a better outcome of the battle, but on the other hand, it is a relief. After the battle has been fought, neither Colonel Blotto nor his opponent will have to regret their choice: Even if both

had known the other's strategy in advance, neither of them would have had an incentive to change his own strategy.

This is an interesting feature of this game: The strategy selected by Colonel Blotto and the strategy selected by his opponent as above are **best responses** against one another. In terms of the payoff matrix, the entry 0 in the row $((0, 2, 3)$ and column $(0, 2, 3))$ is a "saddle point"; it is a *minimum* in its row and a *maximum* in its column. A pair of strategies that are best responses against one another is called a **Nash equilibrium** of the game. As we will see next, not every game has a Nash equilibrium in this sense.

The Rock-Paper-Scissors game. Alice and Bob independently choose a hand gesture indicating either a rock, a piece of paper, or a pair of scissors. If both players choose the same gesture, the game is a draw, and otherwise, there is a cyclic pattern: Scissors beats paper (by cutting it), paper beats rock (by wrapping it up), rock beats scissors (by making it blunt). Assuming that the payoff to Alice is 1 if she wins, 0 if there is a draw, and -1 if she loses, the payoff matrix is

	rock	paper	scissors
rock	0	-1	1
paper	1	0	-1
scissors	-1	1	0

This game has no Nash equilibrium in the sense explained above. No entry of the payoff matrix is a minimum in its row and a maximum in its column at the same time. In more human terms, after every game, the player who lost may regret not to have played the gesture that would have beaten the gesture of the winner (and both may regret in the case of a draw). It is impossible for both players to fix strategies that are best responses each against the other.

But when we generalize the notion of a strategy, there *is* a way for both players to avoid regret. Both should decide *randomly*, selecting each of the gestures with probability 1/3. Even this strategy may lose, of course, but still there is no reason for regret, since with the same probability 1/3, it could have won, and the fact that it didn't is not a fault of the strategy but just bad luck. Indeed, in this way both Alice and Bob can guarantee that their payoff is 0 *in expectation*, and it is easy to see that neither of them can do better by unilaterally switching to a different behavior. We say that we have a *mixed Nash equilibrium* of the game (formally defined below).

A surprising fact is that *every* zero-sum game has a mixed Nash equilibrium. It turns out that such an equilibrium "solves" the game in the sense that it tells us (or rather, both of the two players) how to play the game optimally. As we will see in examples, the random decisions that are involved in a mixed Nash equilibrium need not give each of the possible strategies the same probability, as was the case in the very simple Rock-Paper-Scissors game. However, we will prove that suitable probability distributions always exist and, moreover, that they can be computed using linear programming.

Existence and computation of a mixed Nash equilibrium. Let us repeat the setup of zero-sum games in a more formal manner. We have two players, and we stick to calling them Alice and Bob. Alice has a set of m **pure strategies** at her disposal, while Bob has a set of n pure strategies (we assume that $m, n \geq 1$).

Then there is an $m \times n$ payoff matrix M of real numbers such that m_{ij} is Alice's gain (and Bob's loss) when Alice's ith pure strategy is played against Bob's jth pure strategy. For concreteness, we may think of Bob having to pay $\in m_{ij}$ to Alice. Of course, the situation is symmetric in that m_{ij} might be negative, in which case Alice has to pay $\in -m_{ij}$ to Bob.

A **mixed strategy** of a player is a probability distribution over his or her set of pure strategies. We encode a mixed strategy of Alice by an m-dimensional vector of probabilities

$$\mathbf{x} = (x_1, \ldots, x_m), \quad \sum_{i=1}^{m} x_i = 1, \ \mathbf{x} \geq \mathbf{0},$$

and a mixed strategy of Bob by an n-dimensional vector of probabilities

$$\mathbf{y} = (y_1, \ldots, y_n), \quad \sum_{j=1}^{n} y_j = 1, \ \mathbf{y} \geq \mathbf{0}.$$

So a mixed strategy is not a particular case of a pure strategy; in the Rock-Paper-Scissors game, Alice has three possible pure strategies (rock, paper, and scissors), but infinitely many possible mixed strategies: She can choose any three nonnegative real numbers x_1, x_2, x_3 with $x_1 + x_2 + x_3 = 1$, and play rock with probability x_1, paper with probability x_2, and scissors with probability x_3. Each such triple (x_1, x_2, x_3) specifies a mixed strategy.

Given mixed strategies $\mathbf{x}$ and $\mathbf{y}$ of Alice and Bob, the *expected payoff* (expected gain of Alice) when $\mathbf{x}$ is played against $\mathbf{y}$ is

$$\begin{aligned} &\sum_{i,j} m_{ij} \operatorname*{Prob}_{\mathbf{x},\mathbf{y}}[\text{Alice plays } i, \text{Bob plays } j] \\ &\quad = \sum_{i,j} m_{ij} \operatorname{Prob}_{\mathbf{x}}[\text{Alice plays } i] \cdot \operatorname{Prob}_{\mathbf{y}}[\text{Bob plays } j] \\ &\quad = \sum_{i,j} m_{ij} x_i y_j \\ &\quad = \mathbf{x}^T M \mathbf{y}. \end{aligned}$$

Now we are going to formalize the tenet of Colonel Blotto: "Prepare for the worst." When Alice considers playing some mixed strategy $\mathbf{x}$, she expects Bob to play a **best response** against $\mathbf{x}$: a strategy $\mathbf{y}$ that minimizes her expected payoff $\mathbf{x}^T M \mathbf{y}$. Similarly, for given $\mathbf{y}$, Bob expects Alice to play a strategy $\mathbf{x}$ that maximizes $\mathbf{x}^T M \mathbf{y}$.

For a fixed matrix M, these worst-case payoffs are captured by the following two functions:

$$\beta(\mathbf{x}) = \min_{\mathbf{y}} \mathbf{x}^T M\mathbf{y}, \qquad \alpha(\mathbf{y}) = \max_{\mathbf{x}} \mathbf{x}^T M\mathbf{y}.$$

So $\beta(\mathbf{x})$ is the best (smallest) expected payoff that Bob can achieve against Alice's mixed strategy $\mathbf{x}$, and similarly, $\alpha(\mathbf{y})$ is the best (largest) expected payoff that Alice can achieve against Bob's $\mathbf{y}$. It may also be worth noting that $\mathbf{y}_0$ is Bob's best response against some $\mathbf{x}$ exactly if $\mathbf{x}^T M\mathbf{y}_0 = \beta(\mathbf{x})$ (the symmetric statement for Alice is left to the reader).

Let us note that β and α are well-defined functions, since we are optimizing over compact sets. For β, say, the set of all $\mathbf{x}$ representing probability distributions is an $(m-1)$-dimensional simplex in $\mathbb{R}^m$, and hence indeed compact.

8.1.1 Definition. *A pair* $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ *of mixed strategies is a* **mixed Nash equilibrium** *of the game if* $\tilde{\mathbf{x}}$ *is a best response against* $\tilde{\mathbf{y}}$ *and* $\tilde{\mathbf{y}}$ *is a best response against* $\tilde{\mathbf{x}}$ *(the adjective "mixed" is often omitted); in formulas, this can be expressed as*

$$\beta(\tilde{\mathbf{x}}) = \tilde{\mathbf{x}}^T M\tilde{\mathbf{y}} = \alpha(\tilde{\mathbf{y}}).$$

In the Colonel Blotto game, we have even found a (pure) Nash equilibrium (a pair of pure strategies that are best responses against each other). However, the strategies themselves involved random decisions. We regard these decisions as "hard-wired" into the strategies and the payoff matrix.

Alternatively, we can consider each fixed assignment of regiments to passes as a pure strategy. Then we have a considerably larger payoff matrix, and there is no pure Nash equilibrium. Rather, we have a mixed Nash equilibrium. In practical terms it amounts to the same thing as the strategies described in the previous interpretation of the game, namely, dividing the regiments in groups of 3, 2, and 0, and sending one group to each pass at random.

These are two different views (models) of the same game, and we are free to investigate either one, although one may be more convenient or more realistic than the other.

Let us say that Alice's mixed strategy $\tilde{\mathbf{x}}$ is **worst-case optimal** if $\beta(\tilde{\mathbf{x}}) = \max_{\mathbf{x}} \beta(\mathbf{x})$. That is, Alice expects Bob to play his best response against every mixed strategy of hers, and she chooses a mixed strategy $\tilde{\mathbf{x}}$ that maximizes her expected payoff under this (pessimistic) assumption. Similarly, Bob's mixed strategy $\tilde{\mathbf{y}}$ is worst-case optimal if $\alpha(\tilde{\mathbf{y}}) = \min_{\mathbf{y}} \alpha(\mathbf{y})$.

The next simple lemma shows, among other things, that in order to attain a Nash equilibrium, both players must play worst-case optimal strategies.

8.1.2 Lemma.

(i) *We have* $\max_{\mathbf{x}} \beta(\mathbf{x}) \le \min_{\mathbf{y}} \alpha(\mathbf{y})$. *Actually, for every two mixed strategies* $\mathbf{x}$ *and* $\mathbf{y}$ *we have* $\beta(\mathbf{x}) \le \mathbf{x}^T M\mathbf{y} \le \alpha(\mathbf{y})$.

(ii) *If the pair* $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ *of mixed strategies forms a mixed Nash equilibrium, then both* $\tilde{\mathbf{x}}$ *and* $\tilde{\mathbf{y}}$ *are worst-case optimal.*

(iii) *If mixed strategies* $\tilde{\mathbf{x}}$ *and* $\tilde{\mathbf{y}}$ *satisfy* $\beta(\tilde{\mathbf{x}}) = \alpha(\tilde{\mathbf{y}})$*, then they form a mixed Nash equilibrium.*

Proof. It is an amusing mental exercise to try to "see" the claims of the lemma by thinking informally about players and games. But a formal proof is routine, which is a nice demonstration of the power of mathematical formalism.

The first sentence in (i) follows from the second one, which in turn is an immediate consequence of the definitions of α and β.

In (ii), for any $\mathbf{x}$ we have $\beta(\mathbf{x}) \leq \alpha(\tilde{\mathbf{y}})$ by (i), and since $\beta(\tilde{\mathbf{x}}) = \alpha(\tilde{\mathbf{y}})$, we obtain $\beta(\mathbf{x}) \leq \beta(\tilde{\mathbf{x}})$. Thus $\tilde{\mathbf{x}}$ is worst-case optimal, and a symmetric argument shows the worst-case optimality of $\tilde{\mathbf{y}}$. This proves (ii).

As for (iii), if $\beta(\tilde{\mathbf{x}}) = \alpha(\tilde{\mathbf{y}})$, then by (i) we have $\beta(\tilde{\mathbf{x}}) = \tilde{\mathbf{x}}^T M \tilde{\mathbf{y}} = \alpha(\tilde{\mathbf{y}})$, and hence $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ is a mixed Nash equilibrium. The lemma is proved. □

Here is the main result of this section:

8.1.3 Theorem (Minimax theorem for zero-sum games). *For every zero-sum game, worst-case optimal mixed strategies for both players exist and can be efficiently computed by linear programming. If* $\tilde{\mathbf{x}}$ *is a worst-case optimal mixed strategy of Alice and* $\tilde{\mathbf{y}}$ *is a worst-case optimal mixed strategy of Bob, then* $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ *is a mixed Nash equilibrium, and the number* $\beta(\tilde{\mathbf{x}}) = \tilde{\mathbf{x}}^T M \tilde{\mathbf{y}} = \alpha(\tilde{\mathbf{y}})$ *is the same for all possible worst-case optimal mixed strategies* $\tilde{\mathbf{x}}$ *and* $\tilde{\mathbf{y}}$*.*

The value $\tilde{\mathbf{x}}^T M \tilde{\mathbf{y}}$, the expected payoff in any Nash equilibrium, is called the **value** of the game. Together with Lemma 8.1.2(ii), we get that $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ forms a mixed Nash equilibrium *if and only if* both $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{y}}$ are worst-case optimal.

This theorem, in a sense, tells us everything about playing zero-sum games. In particular, "Prepare for the worst" is indeed the best policy (for nontrivial reasons!). If Alice plays a worst-case optimal mixed strategy, her expected payoff is always *at least* the value of the game, no matter what strategy Bob chooses. Moreover, if Bob is well informed and plays a worst-case optimal mixed strategy, then Alice cannot secure an expected payoff *larger* than the value of the game, no matter what strategy she chooses. So there are no secrets and no psychology involved; both players can as well declare their mixed strategies in advance, and nothing changes.

> Of course, if there are many rounds of the game and Alice suspects that Bob hasn't learned his lesson and doesn't play optimally, she can begin to contemplate how she could exploit this. Then psychology

does come into play. However, by trying strategies that are not worst-case optimal, she is taking a risk, since she also gives Bob a chance to exploit *her*.

It remains to explain the name "minimax theorem." If we consider the equality $\beta(\tilde{\mathbf{x}}) = \alpha(\tilde{\mathbf{y}})$ and use the definitions of β, α, and of worst-case optimality of $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{y}}$, we arrive at

$$\max_{\mathbf{x}} \min_{\mathbf{y}} \mathbf{x}^T M \mathbf{y} = \min_{\mathbf{y}} \max_{\mathbf{x}} \mathbf{x}^T M \mathbf{y},$$

and this is the explanation we offer.

The relation of Theorem 8.1.3 to Lemma 8.1.2(i) is similar to the relation of the duality theorem of linear programming to the weak duality theorem. And indeed, we are going to use the duality theorem in the proof of Theorem 8.1.3 in a substantial way.

Proof of Theorem 8.1.3. We first show how worst-case optimal mixed strategies $\tilde{\mathbf{x}}$ for Alice and $\tilde{\mathbf{y}}$ for Bob can be found by linear programming. Then we prove that the desired equality $\beta(\tilde{\mathbf{x}}) = \alpha(\tilde{\mathbf{y}})$ holds.

We begin by noticing that Bob's best response to a *fixed* mixed strategy $\mathbf{x}$ of Alice can be found by solving a linear program. That is, $\beta(\mathbf{x})$, with $\mathbf{x}$ a concrete vector of m numbers, is the optimal value of the following linear program in the variables $y_1, \ldots, y_n$:

$$\begin{array}{ll} \text{Minimize} & \mathbf{x}^T M \mathbf{y} \\ \text{subject to} & \sum_{j=1}^{n} y_j = 1 \\ & \mathbf{y} \geq \mathbf{0}. \end{array} \tag{8.1}$$

So we can evaluate $\beta(\mathbf{x})$. But for finding a worst-case optimal strategy of Alice we need to maximize β. Unfortunately, $\beta(\mathbf{x})$ is not a linear function, so we cannot directly formulate the maximization of $\beta(\mathbf{x})$ as a linear program. Fortunately, we can circumvent this issue by using linear programming duality.

Using the dualization recipe from Section 6.2, we write down the dual of (8.1):

$$\begin{array}{ll} \text{Maximize} & x_0 \\ \text{subject to} & M^T \mathbf{x} - \mathbf{1} x_0 \geq \mathbf{0} \end{array}$$

(this is a nice exercise in dualization). This dual linear program has only one variable x_0, since $x_1, \ldots, x_m$ are still regarded as fixed numbers. By the duality theorem, the optimal value of the dual linear program is the same as that of the primal, namely, $\beta(\mathbf{x})$.

In order to maximize $\beta(\mathbf{x})$ over all mixed strategies $\mathbf{x}$ of Alice, we set up a new linear program that optically looks exactly like the previous one, but in which $x_1, \ldots, x_m$ are regarded as variables (this works only because the constraints happen to be linear in $x_0, x_1, \ldots, x_m$):

$$\begin{array}{ll} \text{Maximize} & x_0 \\ \text{subject to} & M^T\mathbf{x} - \mathbf{1}x_0 \geq \mathbf{0} \\ & \sum_{i=1}^{m} x_i = 1 \\ & \mathbf{x} \geq \mathbf{0}. \end{array} \tag{8.2}$$

If $(\tilde{x}_0, \tilde{\mathbf{x}})$ denotes an optimal solution of this linear program, we have by construction

$$\tilde{x}_0 = \beta(\tilde{\mathbf{x}}) = \max_{\mathbf{x}} \beta(\mathbf{x}). \tag{8.3}$$

In a symmetric fashion, we can derive a linear program for solving Bob's task of computing a best strategy $\tilde{\mathbf{y}}$. We obtain the problem

$$\begin{array}{ll} \text{minimize} & y_0 \\ \text{subject to} & M\mathbf{y} - \mathbf{1}y_0 \leq \mathbf{0} \\ & \sum_{j=1}^{n} y_j = 1 \\ & \mathbf{y} \geq \mathbf{0} \end{array} \tag{8.4}$$

in the variables $y_0, y_1, \ldots, y_n$. Now an optimal solution $(\tilde{y}_0, \tilde{\mathbf{y}})$ satisfies

$$\tilde{y}_0 = \alpha(\tilde{\mathbf{y}}) = \min_{\mathbf{y}} \alpha(\mathbf{y}). \tag{8.5}$$

So both $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{y}}$ are worst-case optimal strategies (and conversely, worst-case optimal strategies provide optimal solutions of the respective linear programs).

The punchline is that the two linear programs (8.2) and (8.4) are dual to each other! Again, the dualization recipe shows this. It follows that both programs have the same optimum value $\tilde{x}_0 = \tilde{y}_0$. Hence $\beta(\tilde{\mathbf{x}}) = \alpha(\tilde{\mathbf{y}})$ and $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ is a Nash equilibrium by Lemma 8.1.2(iii). □

Rock-Paper-Scissors revisited. To kill the time between their rare public appearances, Santa Claus and the Easter Bunny play the rock-paper-scissors game against each other. The Easter Bunny, however, cannot indicate a pair of scissors with his paw and is therefore limited to two pure strategies. The payoff matrix in this variant is

	rock	paper
rock	0	−1
paper	1	0
scissors	−1	1

We already see that Santa Claus should never play rock: For any possible gesture of the Easter Bunny, paper is a better strategy.

Let us apply the machinery we have just developed to find optimal mixed strategies for Santa Claus and the Easter Bunny. Recall that Santa Claus has to solve the linear program (8.2) to find the probability distribution $\tilde{\mathbf{x}} = (\tilde{x}_1, \tilde{x}_2, \tilde{x}_3)$ that determines his optimal strategy. At the same time, he will compute the game value $\tilde{x}_0$, his expected gain.

The linear program is

$$\begin{array}{lrrrrrrrrl} \text{maximize} & x_0 & & & & & & & & \\ \text{subject to} & & & & x_2 & - & x_3 & - & x_0 & \geq 0 \\ & - & x_1 & & & + & x_3 & - & x_0 & \geq 0 \\ & & x_1 & + & x_2 & + & x_3 & & & = 1 \\ & & & & & & x_1, x_2, x_3 & & & \geq 0. \end{array}$$

A (unique) optimal solution is $(\tilde{x}_0, \tilde{x}_1, \tilde{x}_2, \tilde{x}_3) = (\frac{1}{3}, 0, \frac{2}{3}, \frac{1}{3})$.

The Easter Bunny's problem (8.4) is

$$\begin{array}{lrrrrrrrrl} \text{minimize} & y_0 & & & & & & & & \\ \text{subject to} & & & & - & y_2 & - & y_0 & \leq & 0 \\ & & y_1 & & & & - & y_0 & \leq & 0 \\ & - & y_1 & + & y_2 & - & y_0 & \leq & 0 \\ & & y_1 & + & y_2 & & & = & 1 \\ & & & & & & y_1, y_2 & \geq & 0. \end{array}$$

A (unique) optimal solution is $(\tilde{y}_0, \tilde{y}_1, \tilde{y}_2) = (\frac{1}{3}, \frac{1}{3}, \frac{2}{3})$.

Let us summarize: If both play optimally, Santa Claus wins $\frac{1}{3}$ on average (this is a scientific explanation of why Santa Claus can afford to bring more presents!). Both play paper with probability $\frac{2}{3}$. With the remaining probability $\frac{1}{3}$, Santa Claus plays scissors, while the Easter Bunny plays rock. This result is a simple but still nontrivial application of zero-sum game theory.

In retrospect, the original rock-paper-scissors game might appear rather boring, but this is relative: There is a *World RPS Society* (`http://www.worldrps.com/`) that holds an annual rock-paper-scissors world championship and sells a book on how to play the game.

Choosing numbers. Here is another game, which actually seems fun to play and in which the optimal mixed strategies are not at all obvious. Each of the two players independently writes down an integer between 1 and 6. Then the numbers are compared. If they are equal, the game is a draw. If the numbers differ by one, the player with the smaller number gets €2 from the one with the larger number. If the two numbers differ by two or more, the player with the larger number gets €1 from the one with the smaller number. We want to challenge the reader to compute the optimal mixed strategies for this game (by symmetry, they are the same for both players).

The Colonel Blotto game was apparently first considered by the mathematician Émile Borel in 1921 (he also served as a French minister of the navy, although only for several months). It can be considered for any number of regiments; for 6 regiments, there is still a Nash equilibrium defined by a pair of pure strategies, but for 7 regiments (this is

the case stated in Borel's original paper) it becomes necessary to mix. Borel's paper does not mention the name "Colonel Blotto"; this name appears in Hubert Phillips's *Week-end Problems Book*, a collection of puzzles from 1933.

In his paper, Borel considers *symmetric* games in general. A symmetric game is defined by a payoff matrix M with $M^T = -M$. Borel erroneously states that if the number n of strategies is sufficiently large, one can construct symmetric games in which each player can secure a positive expected payoff, knowing the other player's mixed strategy. He concludes that playing zero-sum games requires psychology, on top of mathematics.

It was only in 1926 that John von Neumann, not knowing about Borel's work and its pessimistic conclusion, formally established Theorem 8.1.3.

Bimatrix games. An important generalization of finite zero-sum games is *bimatrix games*, in which both Alice and Bob want to maximize the payoff with respect to a payoff matrix of their own, A for Alice, and B for Bob (in the zero-sum case, $A = -B$). A bimatrix game also has at least one mixed Nash equilibrium: a pair of strategies $\tilde{\mathbf{x}}, \tilde{\mathbf{y}}$ that are best responses against each other, meaning that $\tilde{\mathbf{x}}^T A\tilde{\mathbf{y}} = \max_{\mathbf{x}} \mathbf{x}^T A\tilde{\mathbf{y}}$ and $\tilde{\mathbf{x}}^T B\tilde{\mathbf{y}} = \max_{\mathbf{y}} \tilde{\mathbf{x}}^T B\mathbf{y}$. We encourage the reader to find a mixed Nash equilibrium in the following variant of the modified rock-paper-scissors game played by Santa Claus and the Easter Bunny: As before, the loser pays €1 to the winner, but in case of a draw, each player donates €0.50 to charity.

The problem of finding a Nash equilibrium in a bimatrix game cannot be formulated as a linear program, and no polynomial-time algorithm is known for it. On the other hand, a Nash equilibrium can be computed by a variant of the simplex method, called the *Lemke–Howson* algorithm (but possibly with exponentially many pivot steps).

In general, Nash equilibria in bimatrix games are not as satisfactory as in zero-sum games, and there is no such thing as "the" game value. We know that in a Nash equilibrium, no player has an incentive to *unilaterally* switch to a different behavior. Yet, it may happen that *both* can increase their payoff by switching simultaneously, a situation that obviously cannot occur in a zero-sum game. This means that the Nash equilibrium was not optimal from the point of view of *social welfare*, and no player has a real desire of being in this particular Nash equilibrium. It may even happen that all Nash equilibria are of this suboptimal nature. Here is an example.

At each of the two department stores in town, *All the Best Deals* and *Buyer's Paradise*, the respective owner needs to decide whether to launch an advertisement campaign for the upcoming Christmas sale. If one store runs a campaign while the competitor doesn't, the expected

extra revenue obtained from customers switching their preferred store (€50,000, say) and new customers (€10,000) easily outweighs the cost of the campaign (€20,000). If, on the other hand, both stores advertise, let us assume that the campaigns more or less neutralize themselves, with extra revenue coming only from new customers in an almost saturated market (€8,000 for each of the stores).

Listing the net revenues a_{ij}, b_{ij} as pairs, in units of €1,000, we obtain the following matrix, with rows corresponding to the strategies of *All the Best Deals*, and columns to *Buyer's Paradise*.

	advertise	don't advertise
advertise	$(-12, -12)$	$(40, -50)$
don't advertise	$(-50, 40)$	$(0, 0)$

If the store owners were friends, they might agree on running no campaign in order to save money (they'd better keep this agreement private in order to avoid a price-fixing charge). But if they do not communicate or mistrust each other, rational behavior will force both of them to waste money on campaigns. To see this, put yourself in the position of one of the store owners. Assuming that your competitor will not advertise, you definitely want to advertise in order to profit from the extra revenue. Assuming that the other store will advertise, you must advertise as well, in order not to lose customers. This means that you will advertise in any case. We say that the strategy "advertise" *strictly dominates* the strategy "don't advertise," so it would be irrational to play the latter.

Because the other store owner reaches the same conclusion, there will be two almost useless campaigns in the end. In fact, the pair of strategies (advertise, advertise) is the unique Nash equilibrium of the game (mixing does not help), but it is suboptimal with respect to social welfare.

In general bimatrix games, the players might not be able to reach the social optimum through rational reasoning, even if this optimum corresponds to an equilibrium of the game. This is probably the most serious deficiency of bimatrix games as models of real-world situations. An example is the *battle of the sexes*. A couple wants to go out at night. He prefers the boxing match, while she prefers the opera, but both prefer going together over going alone. If both are to decide independently where to go, there is no rational way of reaching a social optimum (namely both going out together, no matter where).

When the advertisement game is played repeatedly (after all, there is a Christmas sale every year), the situation changes. In the long run, wasting money every year is such a bad prospect that the following more cooperative behavior makes sense: In the first year, refrain from advertising, and in later years just do what the competitor did the

year before. This strategy is known as TIT FOR TAT. If both stores adopt this policy, they will never waste money on campaigns; but even if one store deviates from it, there is no possibility of exploiting the competitor in the long run. It is easy to see that after a possible first loss, the one playing TIT FOR TAT can "pay" for any further loss, due to a previous loss of the competitor.

The Prisoner's Dilemma. The advertisement game is a variation of the well-known *prisoner's dilemma*, in which two convicts charged with a crime committed together are independently offered (somewhat unethical) plea bargains; if both stay silent, a lack of evidence will lead to only minor punishment. If each testifies against the other, there will be some bearable punishment. But if—and this is the unethical part—exactly one of the two testifies against the other, the betrayer will be rewarded and set free, while the one that remains silent will receive a heavy penalty. As before, rational behavior will force both convicts to testify against each other, even if they had nothing to do with the crime.

A popular introduction to the questions surrounding the prisoner's dilemma is

> W. Poundstone: *Prisoner's Dilemma: John von Neumann, Game Theory, and the Puzzle of the Bomb*, Doubleday, New York 1992

(and the 1964 Stanley Kubrick movie *Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb* might also widen one's horizons in this context).

A general introduction to game theory is

> J. Bewersdorff: *Luck, Logic, and White Lies*, AK Peters, Wellesley 2004.

This book also contains the references to the work by Borel and von Neumann that we have mentioned above.

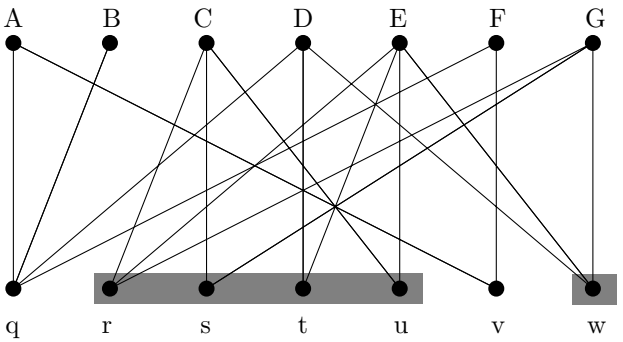
8.2 Matchings and Vertex Covers in Bipartite Graphs

Let us return to the job assignment problem from Section 3.2. There, the human resources manager of a company was confronted with the problem of filling seven positions with seven employees, where every employee has a score (reflecting qualification) for each position he or she is willing to accept. We found that the manager can use linear programming to find an assignment of employees to positions (a perfect matching) that maximizes the sum of

scores. We have also promised to show how the manager can fill the positions optimally if there are *more* employees than positions.

Here we will first disregard the scores and ask under what conditions the task has any solution at all, that is, whether there is any assignment of positions to employees such that every employee gets a position she or he is willing to accept and each position is filled. We know that we can decide this by linear programming (we just invent some arbitrary scores, say all scores 100), but here we ask for a mathematical condition.

For example, let us consider the graph from Section 3.2 but slightly modified (some of the people changed their minds):



There is *no* assignment filling all positions in this situation. This is not immediately obvious, but it becomes obvious once we look at the set $\{r, s, t, u, w\}$ of jobs (marked). Indeed, the set of people willing to take *any* of these 5 jobs is $\{C, D, E, G\}$. This is only 4 people, and so they cannot be assigned to 5 different jobs.

The next theorem, known as *Hall's theorem* or the *marriage theorem*, states that if no assignment exists, we can *always* find such a simple "reason": a subset of k jobs such that the total number of employees willing to take any of them is smaller than k.

Before we formally state and prove this theorem, in the language of bipartite graphs, we need to recall the notions of maximum matching (Section 3.2) and minimum vertex cover (Section 3.3).

A matching in a graph $G = (V, E)$ is a set $E' \subseteq E$ of edges with the property that each vertex is incident to at most one edge in E'. A matching is maximum if it has the largest number of edges among all matchings in G.

A vertex cover of $G = (V, E)$ is a set $V' \subseteq V$ of vertices with the property that each edge is incident to at least one vertex in V'. A vertex cover is minimum if it has the smallest number of vertices among all vertex covers of G.

Hall's theorem gives necessary and sufficient conditions for the existence of a *best possible* matching in a bipartite graph, namely a matching that covers *all* vertices in one class of the vertex bipartition.

8.2.1 Theorem (Hall's theorem). *Let $G = (V, E)$ be a bipartite graph with bipartition $V = X \dot\cup Y$. For a set $T \subseteq X$, we define the neighborhood $N(T) \subseteq Y$ of T as the set*

$$N(T) = \{w \in Y : \{v, w\} \in E \text{ for some } v \in T\}.$$

If for every $T \subseteq X$, $|N(T)| \geq |T|$ holds, then G has a matching that covers all vertices in X.

Actually, we derive the following statement, from which Hall's theorem easily follows.

8.2.2 Theorem (König's theorem). *Let $G = (V, E)$ be a bipartite graph. Then the size of a maximum matching in G equals the size of a minimum vertex cover of G.*

We prove this theorem using the duality of linear programming. There are also combinatorial proofs, and they are actually simpler than the proof offered here. There are at least two reasons in favor of the proof via duality: First, it is a simple example illustrating a powerful general technique. And second, it gives more than just König's theorem. It shows that a maximum matching, as well as a minimum vertex cover, in a bipartite graph can be computed by linear programming. Moreover, the method can be extended to computing a maximum-weight matching in a bipartite graph with weights on the edges.

Let us first see how Hall's theorem follows from König's theorem.

Proof of Theorem 8.2.1. Let $G = (X \dot\cup Y, E)$ be a bipartite graph with $|N(T)| \geq |T|$ for all $T \subseteq X$. We will show that any minimum vertex cover of G has size $n_1 = |X|$. König's theorem then implies that G has a matching of size n_1, and this matching obviously covers X.

For contradiction, suppose that there is a vertex cover C with k vertices from X and fewer than $n_1 - k$ vertices from Y, for some k. The set $T = X \setminus C$ has size $n_1 - k$ and satisfies $|N(T)| \geq n_1 - k$ by the assumption. But this implies that there is a vertex $w \in N(T)$ that is not in $C \cap Y$. Since this vertex has some neighbor $v \in T$, the edge $\{v, w\}$ is not covered by C, a contradiction. □

Totally unimodular matrices. A matrix A is called **totally unimodular** if every square submatrix of A (obtained from A by deleting some rows and some columns) has determinant 0, 1, or -1. We note that, in particular, the entries of A can be only 0, -1, and $+1$.

Such matrices are interesting since an integer program with a totally unimodular constraint matrix is easy to solve—it suffices to solve its LP relaxation, as we will show in Lemma 8.2.4 below. Let us start with a preparatory step.

8.2.3 Lemma. *Let A be a totally unimodular matrix, and consider the matrix $\bar{A}$ obtained from A by appending a unit vector $\mathbf{e}_i$ as a new last column. Then $\bar{A}$ is totally unimodular as well.*

Proof. Let us fix an $\ell \times \ell$ submatrix Q of $\bar{A}$. If Q is a submatrix of A, then $\det(Q) \in \{-1, 0, 1\}$ by total unimodularity of A. Otherwise, we pick the column ℓ of Q that corresponds to the newly added column in $\bar{A}$, and we compute $\det(Q)$ according to the Laplace expansion on this column. We recall that

$$\det(Q) = \sum_{i=1}^{\ell} (-1)^{i+\ell} q_{i\ell} \det(Q^{i\ell}),$$

where $Q^{i\ell}$ is the matrix resulting from Q by removing row i and column ℓ. By construction, the ℓth column may be $\mathbf{0}$ (and we get $\det(Q) = 0$), or there is exactly one nonzero entry $q_{k\ell} = 1$. In that case,

$$\det(Q) = (-1)^{k+\ell} \det(Q^{k\ell}) \in \{-1, 0, 1\},$$

since $Q^{k\ell}$ is a submatrix of A. □

The following lemma is the basis of using linear programming for solving integer programs with totally unimodular matrices.

8.2.4 Lemma. *Let us consider a linear program with n nonnegative variables and m inequalities of the form*

$$\begin{array}{ll} \textit{maximize} & \mathbf{c}^T \mathbf{x} \\ \textit{subject to} & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}, \end{array}$$

where $\mathbf{b} \in \mathbb{Z}^m$. If A is totally unimodular, and if the linear program has an optimal solution, then it also has an integral *optimal solution $\mathbf{x}^* \in \mathbb{Z}^n$.*

Proof. We first transform the linear program into equational form. The resulting system of equality constraints is $\bar{A}\overline{\mathbf{x}} = \mathbf{b}$, with $\bar{A} = (A \,|\, I_m)$ and $\overline{\mathbf{x}} \in \mathbb{R}^{n+m}$. Then we solve the problem using the simplex method. Let $\overline{\mathbf{x}}^*$ be an optimal basic feasible solution, associated with the feasible basis $B \subseteq \{1, 2, \ldots, n+m\}$. Then we know that the nonzero entries of $\overline{\mathbf{x}}^*$ are given by

$$\overline{\mathbf{x}}^*_B = \bar{A}_B^{-1} \mathbf{b};$$

see Section 5.5.

By Cramer's rule, the entries of $\bar{A}_B^{-1}$ can be written as rational numbers with common denominator $\det(\bar{A}_B)$. The matrix $\bar{A}_B$ is a square submatrix of $\bar{A}$, where $\bar{A}$ is totally unimodular (by repeated application of Lemma 8.2.3). Since $\bar{A}_B$ is nonsingular, we get $\det(\bar{A}_B) \in \{-1, 1\}$, and the integrality of $\overline{\mathbf{x}}^*$ follows. □

Incidence matrices of bipartite graphs. Here is the link between total unimodularity and König's theorem. Let $G = (X \dot\cup Y, E)$ be a bipartite graph with n vertices $v_1, \ldots, v_n$ and m edges $e_1, \ldots, e_m$. The **incidence matrix** of G is the matrix $A \in \mathbb{R}^{n \times m}$ defined by

$$a_{ij} = \begin{cases} 1 & \text{if } v_i \in e_j \\ 0 & \text{otherwise.} \end{cases}$$

8.2.5 Lemma. *Let $G = (X \dot\cup Y, E)$ be a bipartite graph. The incidence matrix A of G is totally unimodular.*

Proof. We need to prove that every $\ell \times \ell$ submatrix Q of A has determinant 0 or ± 1, and we proceed by induction on ℓ. The case $\ell = 1$ is immediate, since the entries of an incidence matrix are only 0's and 1's.

Now we consider $\ell > 1$ and an $\ell \times \ell$ submatrix Q. Since the columns of Q correspond to edges, each column of Q has at most two nonzero entries (which are 1). If there is a column with only zero entries, we get $\det(Q) = 0$, and if there is a column with only one nonzero entry, we can expand the determinant on this column (as in the proof of Lemma 8.2.3) and get that up to sign, $\det(Q)$ equals the determinant of an $(\ell-1) \times (\ell-1)$ submatrix Q'. By induction, $\det(Q') \in \{-1, 0, 1\}$, so the same holds for Q.

Finally, if every column of Q contains precisely two 1's, we claim that $\det(Q) = 0$. To see this, we observe that the sum of all rows of Q corresponding to vertices in X is the row vector $(1, \ldots, 1)$, since for each column of Q, exactly one of its two 1's comes from a vertex in X. For the same reason, we get $(1, \ldots, 1)$ by summing up the rows for vertices in Y, and it follows that the rows of Q are linearly dependent. □

Now we are ready to prove König's theorem.

Proof of Theorem 8.2.2. We first consider the integer program

$$\begin{array}{ll} \text{maximize} & \sum_{j=1}^{m} x_j \\ \text{subject to} & A\mathbf{x} \leq \mathbf{1} \\ & \mathbf{x} \geq 0 \\ & \mathbf{x} \in \mathbb{Z}^m, \end{array}$$

where A is the incidence matrix of G. In this integer program, the row of A corresponding to vertex v_i induces the constraint

$$\sum_{j : e_j \ni v_i} x_j \leq 1.$$

This implies that $x_j \in \{0, 1\}$ for all j, and that the edges e_j with $\tilde{x}_j = 1$ in an optimal solution $\tilde{\mathbf{x}}$ form a maximum matching in G.

Next we consider the integer program

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{n} y_i \\ \text{subject to} & A^T\mathbf{y} \geq \mathbf{1} \\ & \mathbf{y} \geq 0 \\ & \mathbf{y} \in \mathbb{Z}^n, \end{array}$$

where A is as before the incidence matrix of G. In this integer program, the row of A^T corresponding to edge e_j induces the constraint

$$\sum_{i:v_i \in e_j} y_i \geq 1.$$

This implies that in any *optimal* solution $\tilde{\mathbf{y}}$ we have $\tilde{y}_i \in \{0, 1\}$ for all i, since any larger value could be decreased to 1. But then the vertices v_i with $\tilde{y}_i = 1$ in an optimal solution $\tilde{\mathbf{y}}$ form a minimum vertex cover of G.

To summarize, the optimum value of the first integer program is the size of a maximum matching in G, and the optimum value of the second integer program is the size of a minimum vertex cover in G.

In both integer programs, we may now drop the integrality constraints without affecting the optimum values: A, and therefore also A^T, are totally unimodular by Lemma 8.2.5, and so Lemma 8.2.4 applies. But the resulting *linear* programs are dual to each other; the duality theorem thus shows that their optimal values are equal, and this proves Theorem 8.2.2. □

It remains to explain the algorithmic implications of the proof (namely, how a maximum matching and a minimum vertex cover can actually be computed). To get a maximum matching, we simply need to find an *integral* optimal solution of the first linear program. When we use the simplex method to solve the linear program, we get this for free; see the proof of Lemma 8.2.4 and, in particular, the claim toward the end of its proof. Otherwise, we can apply Theorem 4.2.3(ii) to construct a basic feasible solution from any given optimal solution, and this basic feasible solution will be integral. A minimum vertex cover is obtained from the second (dual) linear program in the same fashion.

The previous arguments show more: Given edge weights $w_1, \ldots, w_m$, any optimal solution of the integer program

$$\begin{array}{ll} \text{maximize} & \sum_{j=1}^{m} w_j x_j \\ \text{subject to} & A\mathbf{x} \leq \mathbf{1} \\ & \mathbf{x} \geq 0 \\ & \mathbf{x} \in \mathbb{Z}^m \end{array}$$

corresponds to a *maximum-weight matching* in G. Since we can, as before, relax the integrality constraints without affecting the optimum value, an integral optimal solution of the relaxation can be found, and it yields a maximum-weight matching in G. This solves the optimal job assignment problem if there are more employees than jobs.

The fact that a linear program with totally unimodular constraint matrix and integral right-hand side has an integral optimal solution implies something much stronger: Since every vertex of the feasible region is optimal for some objective function (see Section 4.4), we know that *all* vertices of the feasible region are integral. We say that the feasible region forms an **integral polyhedron**.

Such integrality results together with linear programming duality can yield interesting and powerful *minimax theorems*. König's theorem is one such example. Another classical minimax theorem that can be proved along these lines is the *max-flow-min-cut theorem*.

To state this theorem, we consider a network modeled by a directed graph $G = (V, E)$ with edge capacities w_e. In Section 2.2, we have interpreted them as maximum transfer rates of data links. Given two designated vertices, the source s and the sink t, the *maximum flow* value is the maximum rate at which data can flow from s to t through the network.

The *minimum cut* value, on the other hand, is the minimum total capacity of any set of data links whose breakdown disconnects t from s.

The max-flow-min-cut theorem states that the maximum flow value is *equal* to the minimum cut value. One of several proofs writes both values as optimal values of linear programs that are dual to each other.

When we consider matchings and vertex covers in general (not necessarily bipartite) graphs, the situation changes: Total unimodularity no longer applies, and the "duality" between the two concepts disappears.

In fact, the problem of finding a minimum vertex cover in a general graph is computationally difficult (NP-hard); see Section 3.3. A maximum-weight matching, on the other hand, can still be computed in polynomial time for general graphs, although this result is by no means trivial. Behind the scenes, there is again an integrality result, based on the notion of *total dual integrality*; see the glossary.

8.3 Machine Scheduling

In the back office of the copy shop *Copy & Paste*, the operator is confronted with n copy jobs submitted by customers the night before. For processing them, she has m photocopying machines with different features at her disposal. For all i, j, the operator quickly estimates how long it would take the ith machine to process the jth job, and she makes a table of the resulting running times, like this:

	Single B&W	Duplex B&W	Duplex Color
Master's thesis, 90 pages two-sided, 10 B&W copies	—	45 min	60 min
All the Best Deals flyer, 1 page one-sided, 10,000 B&W copies	2h 45 min	4h 10 min	5h 30 min
Buyer's Paradise flyer, 1 page one-sided, 10,000 B&W copies	2h 45 min	4h 10 min	5h 30 min
Obituary, 2 pages two-sided, 100 B&W copies	—	2 min	3 min
Party platform, 10 pages two-sided, 5,000 color copies	—	—	3h 30 min

Since the operator can go home as soon as all jobs have been processed, her goal is to find an assignment of jobs to machines (a *schedule*) such that the **makespan**—the time needed to finish all jobs—is minimized. In our example, this is not hard to figure out: For the party platform, there is no choice between machines. We can also observe that it is advantageous to use both B&W machines for processing the two flyers, no matter where the thesis and the obituary go. Given this, the makespan is at least 4h 55 min if the thesis is processed on the Duplex B&W machine, so it is better put on the color machine to achieve the optimum makespan of 4h 30 min (with the obituary running on the B&W machine).

In general, finding the optimum makespan is computationally difficult (NP-hard). The obvious approach of trying all possible schedules is of course not a solution for a larger number n of jobs. What we show in this section is that the operator can quickly compute a schedule whose makespan is at most *twice* as long as the optimum makespan. All she needs for that are some linear programming skills. (To really appreciate this result, one shouldn't think of a problem with 5 jobs but with thousands of jobs.)

We should emphasize that in this scheduling problem the jobs are considered *indivisible*, and so each job must be processed on a single machine. This, in a sense, is what makes the problem difficult. As we will soon see, an optimal "fractional schedule," where a single job could be divided among several machines in arbitrary ratios, can be found efficiently by linear programming.

Two integer programs for the scheduling problem. Let us identify the m machines with the set $M := \{1, \ldots, m\}$ and the n jobs with the set $J := \{m+1, \ldots, m+n\}$.

Let d_{ij} denote the running time of job $j \in J$ on machine $i \in M$. We assume $d_{ij} > 0$. To simplify notation, we also assume that any machine can process any job: An infeasible assignment of job j to machine i can be modeled by a large running time $d_{ij} = K$. If K is larger than the sum of all "real" running times, the optimal schedule will avoid infeasible assignments, given that there is a feasible schedule at all.

With these notions, the following integer program in the variables t and x_{ij}, $i \in M$, $j \in J$ computes an assignment of jobs to machines that minimizes the makespan:

$$\begin{array}{lrcll} \text{Minimize} & t & & & \\ \text{subject to} & \sum_{i \in M} x_{ij} & = & 1 & \text{for all } j \in J \\ & \sum_{j \in J} d_{ij} x_{ij} & \leq & t & \text{for all } i \in M \\ & x_{ij} & \geq & 0 & \text{for all } i \in M, j \in J \\ & x_{ij} & \in & \mathbb{Z} & \text{for all } i \in M, j \in J. \end{array}$$

Under the integrality constraints, the conditions $\sum_{i \in M} x_{ij} = 1$ and $x_{ij} \geq 0$ imply that $x_{ij} \in \{0, 1\}$ for all i, j. With the interpretation that $x_{ij} = 1$ if job j is assigned to machine i, and $x_{ij} = 0$ otherwise, the first n equations stipulate that each job is assigned to exactly one machine. The next m inequalities make sure that no machine needs more time than t to finish all jobs assigned to it. Minimizing t leads to equality for at least one of the machines, so the best t is indeed the makespan of an optimal schedule.

As we have already seen in Section 3.3 (the minimum vertex cover problem), solving the *LP relaxation* obtained from an integer program by deleting the integrality constraints can be a very useful step toward an approximate solution of the original problem. In our case, this approach needs an additional twist: We will relax another integer program, obtained by adding *redundant constraints* to the program above. After dropping integrality, the added constraints are no longer redundant and lead to a better LP relaxation.

Let t_{opt} be the makespan of an optimal schedule. If $d_{ij} > t_{\text{opt}}$, then we know that job j cannot run on machine i in any optimal schedule, so we may add the constraint $x_{ij} = 0$ to the integer program without affecting its validity. More generally, if T is any upper bound on t_{opt}, we can write down the following integer program, which has the same optimal solutions as the original one:

$$\begin{array}{lrcll} \text{Minimize} & t & & & \\ \text{subject to} & \sum_{i \in M} x_{ij} & = & 1 & \text{for all } j \in J \\ & \sum_{j \in J} d_{ij} x_{ij} & \leq & t & \text{for all } i \in M \\ & x_{ij} & \geq & 0 & \text{for all } i \in M, j \in J \\ & x_{ij} & = & 0 & \text{for all } i \in M, j \in J \text{ with } d_{ij} > T \\ & x_{ij} & \in & \mathbb{Z} & \text{for all } i \in M, j \in J. \end{array}$$

But we do not know t_{opt}, so what is the value of T we use for the relaxation? There is no need to specify this right here; for the time being, you can imagine that we set $T = \max_{ij} d_{ij}$, a value that will definitely work, because it makes our second integer program coincide with the first one.

A good schedule from the LP relaxation. As already indicated, the first step is to solve the LP relaxation, denoted by $\text{LPR}(T)$, of our second integer program:

$$
\begin{array}{lrcl}
\text{Minimize} & t & & \\
\text{subject to} & \sum_{i \in M} x_{ij} & = & 1 \quad \text{for all } j \in J \\
& \sum_{j \in J} d_{ij} x_{ij} & \leq & t \quad \text{for all } i \in M \\
& x_{ij} & \geq & 0 \quad \text{for all } i \in M, j \in J \\
& x_{ij} & = & 0 \quad \text{for all } i \in M, j \in J \text{ with } d_{ij} > T.
\end{array}
$$

In contrast to the vertex cover application, we cannot work with any optimal solution of the relaxation, though: We need a *basic feasible* optimal solution; see Section 4.2. To be more precise, we rely on the following property of a basic feasible solution; see Theorem 4.2.3.

8.3.1 Assumption. *The columns of the constraint matrix A corresponding to the nonzero variables x^*_{ij} in the optimal solution of* LPR(T) *are linearly independent.*

As usual, the nonnegativity constraints $x_{ij} \geq 0$ do not show up in A. In case the simplex method is used to solve the relaxation, such a solution comes for free (the column of A corresponding to t could then actually be added to the set of columns in the assumption, but this is not needed). Otherwise, we can easily construct a solution satisfying the assumption from any given optimal solution, according to the recipe in the proof of Theorem 4.2.3.

At this point the reader may wonder why we would want to use an algorithm different from the simplex method here, in particular when we are searching for a basic feasible optimal solution. The reason is of theoretical nature: We want to prove that a schedule whose makespan is at most twice the optimum can be found *in polynomial time*. As we have pointed out in the introductory part of Chapter 7, the simplex method is not known to run in polynomial time for any pivot rule, and for most pivot rules it simply does not run in polynomial time. For the theoretical result we want, we had therefore better use one of the polynomial-time methods for solving linear programs, sketched in Chapter 7. For practical purposes, the simplex method will do, of course.

In general terms, what we are trying to develop here is a polynomial-time *approximation algorithm* for an NP-hard problem. Since complexity theory indicates that we will not be able to solve the problem exactly within reasonable time bounds, it is quite natural to ask for an approximate solution that can be obtained in polynomial time. The quality of an approximate solution is typically measured by the *approximation factor*, the ratio between the value of the approximate solution and the value of an optimal solution. In our approximation algorithm for the scheduling problem, this factor will be at most 2.

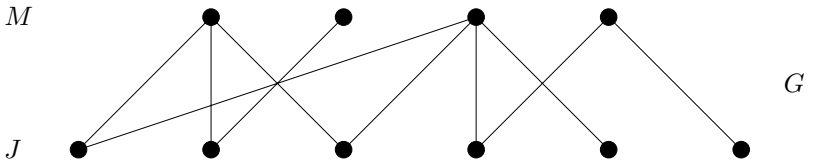
Let us fix the values t^* and x^*_{ij} of the variables in some optimal solution of the LP relaxation. We now consider the bipartite graph $G = (M \cup J, E)$, with

$$E = \{\{i,j\} \subseteq M \cup J \mid x^*_{ij} > 0\}.$$

For an arbitrary optimal solution, this graph could easily be a (boring) complete bipartite graph, but under Assumption 8.3.1 it becomes more interesting.

8.3.2 Lemma. *In any subgraph of G (obtained by deleting edges, or vertices with their incident edges), the number of edges is at most the number of vertices.*

The graph G for $m = 4$ machines and $n = 6$ jobs might look like this, for example:



Proof. Let A be the constraint matrix of the LP relaxation. It has one row for each machine, one for each job, and one for each runtime d_{ij} exceeding T. The columns of A corresponding to the nonzero variables x^*_{ij} (equivalently, to the edges of G) are linearly independent by our assumption.

Now we consider any subgraph of G, with vertex set $M' \cup J' \subseteq M \cup J$ and edge set $E' \subseteq E$. Let A' be the submatrix obtained from A by restricting to rows corresponding to $M' \cup J'$, and to columns corresponding to E'.

Claim. *The columns of A' are linearly independent.*

> To see this, we first observe that the columns of A corresponding to E' are linearly independent, simply because $E' \subseteq E$. Any variable $x_{ij}, \{i,j\} \in E'$, occurs in the inequality for machine $i \in M'$ and in the equation for job $j \in J'$, but in no other equation, since $x^*_{ij} > 0$ implies $d_{ij} \leq T$. This means that the columns of A corresponding to E' have zero entries in all rows except for those corresponding to machines or jobs in $M' \cup J'$. Hence, these columns remain linearly independent even when we restrict them to the rows corresponding to $M' \cup J'$.

By the claim, we have $|E'| \leq |M' \cup J'|$, and this is the statement of the lemma. □

This structural result about G allows us to find a good schedule.

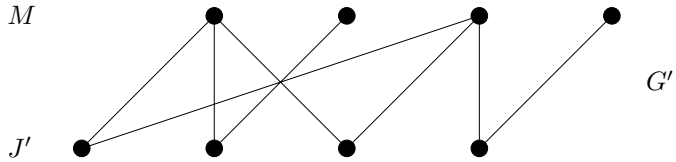
8.3.3 Lemma. *Let $T \geq 0$ be such that the LP relaxation* $\mathrm{LPR}(T)$ *is feasible, and suppose that a feasible solution is given that satisfies Assumption 8.3.1 and has value $t = t^*$. Then we can efficiently construct a schedule of makespan at most $t^* + T$.*

Proof. We need to assign each job to some machine. We begin with the jobs j that have degree one in the graph G, and we assign each such j to its unique neighbor i. By the construction of G and the equation $\sum_{i \in M} x_{ij} = 1$, we have $x^*_{ij} = 1$ in this case. If machine i has been assigned a set S_i of jobs in this way, it can process these jobs in time

$$\sum_{j \in S_i} d_{ij} = \sum_{j \in S_i} d_{ij} x^*_{ij} \leq \sum_{j \in J} d_{ij} x^*_{ij} \leq t^*.$$

So each machine can handle the jobs assigned by this partial schedule in time t^*.

Next we remove all assigned jobs and their incident edges from G. This leaves us with a subgraph $G' = (M \cup J', E')$. In G', all vertices of degree one are machines. In the example depicted above, two jobs have degree one, and their deletion results in the following subgraph G':



We will show that we can find a matching in G' that covers all the remaining jobs. If we assign the jobs according to this matching, every machine gets at most one additional job, and this job can be processed in time at most T by the construction of our second integer program. Therefore, the resulting full schedule has makespan at most $t^* + T$.

It remains to construct the matching. To this end, we use *Hall's theorem* from Section 8.2. According to this theorem, a matching exists if for every subset $J'' \subseteq J'$ of jobs, its neighborhood (the set of all machines connected to at least one job in J'') has size at least $|J''|$.

To check this condition, we let $J'' \subseteq J'$ be such a subset of jobs and $N(J'')$ its neighborhood. If e is the number of edges in the subgraph of G' induced by $J'' \cup N(J'')$, then Lemma 8.3.2 guarantees that $e \leq |J'' \cup N(J'')|$. On the other hand, since every job has at least two neighbors, we have $e \geq 2|J''|$, and this shows that $|N(J'')| \geq |J''|$.

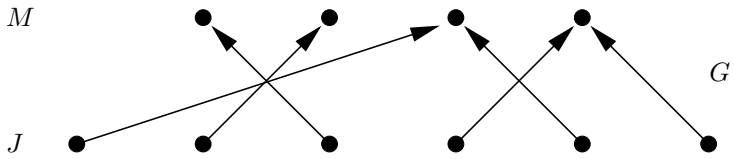
Although this proof is nonconstructive, we can easily find the matching (once we know that it exists) by linear programming as in Section 8.2, or by other known polynomial-time methods. □

There is a direct way of constructing the matching in the proof of Lemma 8.3.3 that relies neither on Halls's theorem nor on general (bipartite) matching algorithms. It goes as follows: Lemma 8.3.2 implies that each connected component of G' is either a tree, or a tree with one extra edge connecting two of its vertices. In the latter case, the

component has exactly one cycle of even length, because G' is bipartite. Therefore, we can match all jobs occurring on cycles, and after removing the vertices of all cycles, we are left with a subgraph G'', all of whose connected components are trees, *with at most one vertex per tree being a former neighbor of a cycle vertex*. It follows that in every tree of G'', at most one vertex can be a job of degree one, since all other degree-one vertices already had degree one in G' and are therefore machines.

The matching of the remaining jobs is easy. We root any tree in G'' at its unique job of degree one (or at any vertex if there is no such job), and we match every job to one of its children in the rooted tree. For this, we observe that there cannot be an isolated job in G'': Since a job in G'' was on no cycle in G', the removal of cycles can affect only one of the at least two neighbors of the job in G'.

For our running example, here is a complete assignment of jobs to machines obtained from the described procedure:



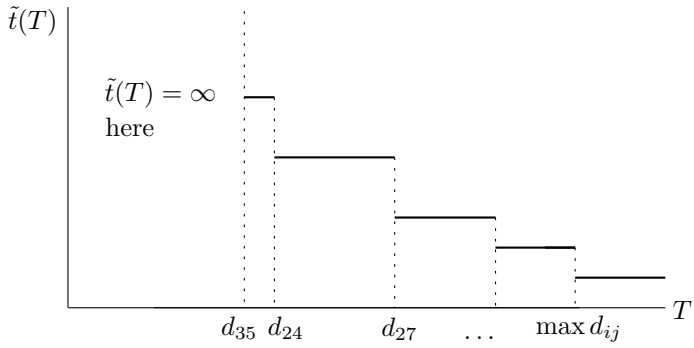
Choosing the parameter T. How good is the bound we get from the previous lemma? We will assume that t^* is the value of an optimal basic feasible solution of the LP relaxation with parameter T. Then t^* is a lower bound for the optimum makespan t_{opt}, so this part of the bound looks promising. But when we recall that T must be an *upper* bound for t_{opt} in order for our second integer program to be valid, it seems that we would have to choose $T = t_{\text{opt}}$ to get makespan at most $2t_{\text{opt}}$. But this cannot be done, since we have argued above that it is hard to compute t_{opt} (and if it were easy, there would be no need for an approximate solution anyway).

Luckily, there is a way out. Reading Lemma 8.3.3 carefully, we see that T only needs to be chosen so that the *LP relaxation* $\text{LPR}(T)$ is feasible, and there is no need for the second *integer* program to be feasible.

If $\text{LPR}(T)$ is feasible, Lemma 8.3.3 allows us to construct a schedule with makespan at most t^*+T, so the best T is the one that minimizes t^*+T subject to $\text{LPR}(T)$ being feasible. Since t^* depends on T, we make this explicit now by writing $t^* = t^*(T)$. If $\text{LPR}(T)$ is infeasible, we set $t^*(T) = \infty$.

How to find the best T. We seek a point T^* in which the function $f(T) = t^*(T) + T$ attains a minimum.

First we observe that $t^*(T)$ is a step function as in the following picture:



Indeed, let us start with the value $T = \max_{ij} d_{ij}$, and let us decrease T continuously. The value of $t^*(T)$ may change (jump up) only immediately after moments when a new constraint of the form $x_{ij} = 0$ appears in $\mathrm{LPR}(T)$, and this happens only for $T = d_{ij}$. Between these values the function $t^*(T)$ stays constant.

Consequently, the function $f(T) = t^*(T) + T$ is linearly increasing on each interval between two consecutive d_{ij}'s, and the minimum is attained at some d_{ij}. So we can compute the minimum, and the desired best value T^*, by solving at most mn linear programs of the form $\mathrm{LPR}(T)$, with T ranging over all d_{ij}. Under our convention that $t^*(T) = \infty$ if $\mathrm{LPR}(T)$ is infeasible, the minimum will be attained at a value T^* with $\mathrm{LPR}(T^*)$ feasible.

8.3.4 Theorem. *Let T^* be the value of T that minimizes $t^*(T) + T$. With $T = T^*$, the algorithm in the proof of Lemma 8.3.3 computes a schedule of makespan at most $2t_{\mathrm{opt}}$.*

Proof. We know that for $T = t_{\mathrm{opt}}$, the second integer program is feasible and has optimum value t_{opt}. Hence $\mathrm{LPR}(t_{\mathrm{opt}})$ is feasible as well and its optimum value can be only smaller: $t^*(t_{\mathrm{opt}}) \le t_{\mathrm{opt}}$. We thus have

$$\begin{aligned} t^*(T^*) + T^* &= \min_T \Big(t^*(T) + T \Big) \\ &\le t^*(t_{\mathrm{opt}}) + t_{\mathrm{opt}} \\ &\le 2t_{\mathrm{opt}}. \end{aligned}$$

□

The 2-approximation algorithm for the scheduling problem is adapted from the paper

J. K. Lenstra, D. B. Shmoys, É. Tardos: Approximation algorithms for scheduling unrelated parallel machines, *Mathematical Programming* 46(1990), pages 259–271.

The paper also proves that it is NP-hard to approximate the optimum makespan with a factor less than $\frac{3}{2}$.

Aiming at simplicity, we have presented a somewhat inefficient version of the algorithm. A considerably faster version, with the same approximation guarantee, can be obtained if we do not minimize the function $T \mapsto t^*(T) + T$, but instead we only look for the smallest T with $t^*(T) \leq T$. Such a value of T still guarantees $t^*(T) + T \leq 2t_{\text{opt}}$, but it can be found by binary search over the sorted sequence of the d_{ij}. In this way, it suffices to solve LPR(T) for $O(\log mn)$ values of T, instead of mn values as in our presentation. See the paper quoted above for details.

8.4 Upper Bounds for Codes

Error-correcting codes. Let us consider a DVD player that has a remote control unit with 16 keys. Whenever one of the keys is pressed, the unit needs to communicate this event to the player.

A natural option would be to send a 4-bit sequence: Since there are $2^4 = 16$ different 4-bit sequences (referred to as "words" in this context), a 4-bit sequence is an economical way of communicating one of 16 possibilities.

However, let us suppose that the transmission of bits from the remote control to the player is not quite reliable, and that each of the transmitted bits can be received incorrectly with some small probability, say 0.005. Then we expect that about 2% of the commands are received incorrectly, which can be regarded as a rather serious flaw of the device.

One possibility of improvement is to triple each of the four transmitted bits. That is, instead of a 4-bit word *abcd* the unit sends the 12-bit word *aaabbbcccddd*. Now a transmission error in a *single* bit can be recognized and corrected. For example, if the sequence 111001000111 is received, and if we assume that at most one bit was received erroneously, it is clear that 111000000111 must have been sent. Thus the original 4-bit sequence was 1001. Of course, it might be that actually two or more bits are wrong, and then the original sequence is not reconstructed correctly, but this has much lower probability. Namely, if we assume that the errors in different bits are independent and occur with probability 0.005 (which may or may not be realistic, depending on the technical specifications), then the probability of two or more errors in a 12-bit sequence is approximately 0.16%. This is a significant improvement in reliability. However, the price to pay is transmitting three times as many bits, which presumably exhausts the battery of the remote control much faster.

A significantly better solution to this problem was discovered by Richard Hamming in the 1950s (obviously not in the context of DVD players). In

order to distinguish 16 possibilities, we send one of the following 7-bit words: 0000000, 0001011, 0010101, 0011110, 0100110, 0101101, 0110011, 0111000, 1000111, 1001100, 1010010, 1011001, 1100001, 1101010, 1110100, 1111111. It can be checked that every two of these words differ in at least 3 bits. (This fact can be checked by brute force, but it is also a consequence of an elegant general theory, which we do not treat here.) Therefore, if one error occurs in the transmission, the sequence that was sent can be reconstructed uniquely. Hence the capability of correcting any single-bit error is retained, but the number of transmitted bits is reduced to slightly more than half compared to the previous approach.

A similar problem can be investigated for other settings of the parameters as well. In general, we want to communicate one of N possibilities, we do it by transmitting an n-bit word, and we want that any at most r errors can be corrected.

This problem has an enormous theoretical and practical significance. Our example with a DVD player was simple-minded, but error-correcting codes play a role in any technology involving transmission or storage of information, from computer disks and cell phones to deep-space probes. We now introduce some common terminology related to error-correcting codes.

Terminology. The **Hamming distance** of two words $\mathbf{w}, \mathbf{w}' \in \{0,1\}^n$ is the number of bits in which $\mathbf{w}$ differs from $\mathbf{w}'$:

$$d_H(\mathbf{w}, \mathbf{w}') := |\{j \in \{1, \ldots, n\} : w_j \neq w'_j\}|.$$

The Hamming distance can be interpreted as the number of errors "necessary" to transform $\mathbf{w}$ into $\mathbf{w}'$. The **weight** of $\mathbf{w} \in \{0,1\}^n$ is the number of 1's in $\mathbf{w}$:

$$|\mathbf{w}| := |\{j \in \{1, \ldots, n\} : w_j = 1\}|.$$

Finally, for $\mathbf{w}, \mathbf{w}' \in \{0,1\}^n$, we define their sum modulo 2 as the word

$$\mathbf{w} \oplus \mathbf{w}' = ((w_1 + w'_1) \bmod 2, \ldots, (w_n + w'_n) \bmod 2) \in \{0,1\}^n.$$

These three notions are interrelated by the formula

$$d_H(\mathbf{w}, \mathbf{w}') = |\mathbf{w} \oplus \mathbf{w}'|. \tag{8.6}$$

In the last of the solutions to the DVD-player problem discussed above, the crucial object was the set $\mathcal{C} = \{0000000, 0001011, 0010101, 0011110, 0100110, 0101101, 0110011, 0111000, 1000111, 1001100, 1010010, 1011001, 1100001, 1101010, 1110100, 1111111\}$ of 7-bit words in which every two distinct words had Hamming distance at least 3. In coding theory, any subset $\mathcal{C} \subseteq \{0,1\}^n$ is called a *code*. (This may sound strange, since under a code one usually imagines some kind of method or procedure for coding, but in the theory of error-correcting codes one has to get used to this terminology.) For correcting errors, the crucial parameter is the distance of the code:

8.4.1 Definition. *A code* $\mathcal{C} \subseteq \{0,1\}^n$ *has* **distance** d *if* $d_H(\mathbf{w}, \mathbf{w}') \geq d$ *for any two distinct words* $\mathbf{w}, \mathbf{w}'$ *in* $\mathcal{C}$. *For* $n, d \geq 0$, *let* $A(n,d)$ *denote the maximum cardinality of a code* $\mathcal{C} \subseteq \{0,1\}^n$ *with distance* d.

We claim that a code $\mathcal{C}$ can correct any at most r errors if and only if it has distance at least $2r+1$. Indeed, on the one hand, if $\mathcal{C}$ contained two distinct words $\mathbf{w}', \mathbf{w}''$ that differ in at most $2r$ bits, we consider any word $\mathbf{w}$ resulting from $\mathbf{w}'$ by flipping exactly half of the bits (rounded down) that distinguish $\mathbf{w}'$ from $\mathbf{w}''$. When the word $\mathbf{w}$ is received, there is no way to tell which of the words $\mathbf{w}'$ and $\mathbf{w}''$ was intended. On the other hand, if any two distinct code words differ by at least $2r+1$ bits, then for any word $\mathbf{w} \in \{0,1\}^n$ there is at most one code word from which $\mathbf{w}$ can be obtained through r or fewer errors, and this must be the word that was sent when $\mathbf{w}$ is received.

Given the number n of bits we can afford to transmit and the number r of errors we need to be able to correct, we want a code $\mathcal{C} \subseteq \{0,1\}^n$ with distance at least $2r+1$ and with $|\mathcal{C}|$ as large as possible, since the number of words in the code corresponds to the amount of information that we can transmit. Thus determining or estimating $A(n,d)$, the maximum possible size of a code $\mathcal{C} \subseteq \{0,1\}^n$ with distance d, is one of the main problems of coding theory.

The problem of finding the largest codes for given n and r can *in principle* be solved by complete enumeration: We can go through all possible subsets of $\{0,1\}^n$ and output the largest one that gives a code with distance d. However, this method becomes practically infeasible already for very small n. It turns out that the problem, for arbitrary n and d, is computationally difficult (NP-hard). Starting from very moderate values of n and d, the maximum code sizes are not exactly known, except for few lucky cases. Tightening the known upper and lower bounds on maximum sizes of error-correcting codes is the topic of ongoing research in coding theory.

In this section we present a technique for proving upper bounds based on linear programming. When this technique was introduced by Philippe Delsarte in 1973, it provided upper bounds of unprecedented quality.

Special cases. For all n, we have $A(n,1) = 2^n$, because any code has distance 1. The case $d = 2$ is slightly more interesting. By choosing $\mathcal{C}$ as the set of all words of even weight, we see that $A(n,2) \geq 2^{n-1}$. But we actually have $A(n,2) = 2^{n-1}$, since it is easy to show by induction that every code with more than 2^{n-1} words contains two words of Hamming distance 1.

Given the simplicity of the cases $d = 1, 2$, it may come as a surprise that already for $d = 3$, little is known. This is the setup for error-correcting codes with one error allowed. Exact values of $A(n,3)$ have been determined only up to $n \leq 16$; for $n = 17$, for example, the known bounds are

$$5312 \leq A(17,3) \leq 6552.$$

The sphere-packing bound. For any n and d, a simple upper bound on $A(n,d)$ can be obtained by a *volume argument*. Let us motivate this with a real-life analogy. The local grocery is exhibiting a large glass box filled with peas, and the person to make the most accurate estimate of the number of peas in the box wins a prize. Without any counting, you can conclude that the number of peas is bounded above by the volume of the box divided by the volume of a single pea (assuming that all the peas have the same volume).

The same kind of argument can be used for the number $A(n,d)$, where we may assume in our application that $d = 2r+1$ is odd. Let us fix any code $\mathcal{C}$ of distance d. Now we think of the set $\{0,1\}^n$ as the glass box, and of the $|\mathcal{C}|$ *Hamming balls*

$$B(\mathbf{w},r) := \{\mathbf{w}' \in \{0,1\}^n : d_H(\mathbf{w},\mathbf{w}') \le r\}, \quad \mathbf{w} \in \mathcal{C},$$

as the peas. Since the code has distance $2r+1$, all these Hamming balls are disjoint and correspond in our analogy to peas. Consequently, their number cannot be larger than the total number of words (the volume of the box) divided by the number of words in a single Hamming ball (the volume of a pea). The number of words at Hamming distance exactly i from $\mathbf{w}$ is $\binom{n}{i}$. This implies

$$|B(\mathbf{w},r)| = \sum_{i=0}^{r} \binom{n}{i},$$

and the following upper bound on $A(n,2r+1)$ is obtained.

8.4.2 Lemma (Sphere-packing bound). *For all n and r,*

$$A(n,2r+1) \le \left\lfloor \frac{2^n}{\sum_{i=0}^{r}\binom{n}{i}} \right\rfloor.$$

For example, the sphere-packing bound gives $A(7,3) \le 16$ (and so the Hamming code in our initial example is optimal), and

$$A(17,3) \le \lfloor 131072/18 \rfloor = 7281.$$

In the following theorem, which is the main result of this section, an upper bound on $A(n,d)$ is expressed as an optimum of a certain linear program.

8.4.3 Theorem (The Delsarte bound). *For integers n, i, t with $0 \le i$, $t \le n$, let us put*

$$K_t(n,i) = \sum_{j=0}^{\min(i,t)} (-1)^j \binom{i}{j}\binom{n-i}{t-j}.$$

Then for every n and d, $A(n,d)$ is bounded above by the optimum value of the following linear program in variables $x_0, x_1, \ldots, x_n$:

$$\begin{array}{lll} \textit{Maximize} & x_0 + x_1 + \cdots + x_n & \\ \textit{subject to} & x_0 = 1 & \\ & x_i = 0, & i = 1, 2, \ldots, d-1 \\ & \sum_{i=0}^{n} K_t(n,i) \cdot x_i \geq 0, & t = 1, 2, \ldots, n \\ & x_0, x_1, \ldots, x_n \geq 0. & \end{array}$$

Example. Using the sphere packing bound, we have previously found $A(17,3) \leq 7281$. To compute the Delsarte bound, we solve the linear program in the theorem (after eliminating x_0, x_1, x_2, which are actually constants, we have 15 nonnegative variables and 17 constraints). The optimum value is $6553\frac{3}{5}$, which implies $A(17,3) \leq 6553$. The current best upper bound is 6552, an improvement by only 1!

Toward an explanation. The proof of the Delsarte bound will proceed as follows. With every subset $\mathcal{C} \subseteq \{0,1\}^n$ we associate nonnegative real quantities $\tilde{x}_0, \tilde{x}_1, \ldots, \tilde{x}_n$ such that $|\mathcal{C}| = \tilde{x}_0 + \cdots + \tilde{x}_n$. Then we will show that whenever $\mathcal{C}$ is a code with distance d, the $\tilde{x}_i$ constitute a feasible solution of the linear program in the theorem. It follows that the maximum of the linear program is at least as large as the size of any existing code $\mathcal{C}$ with distance d (but of course, it may be larger, since a feasible solution does not necessarily corresponds to a code).

Given $\mathcal{C} \subseteq \{0,1\}^n$, the $\tilde{x}_i$ are defined by

$$\tilde{x}_i = \frac{1}{|\mathcal{C}|} \cdot \Big| \{(\mathbf{w}, \mathbf{w}') \in \mathcal{C}^2 : d_H(\mathbf{w}, \mathbf{w}') = i\} \Big|, \quad i = 0, \ldots, n.$$

Thus, $\tilde{x}_i$ is the number of ordered pairs of code words with Hamming distance i, divided by the total number of code words. Since any of the $|\mathcal{C}|^2$ ordered pairs contributes to exactly one of the $\tilde{x}_i$, we have

$$\tilde{x}_0 + \tilde{x}_1 + \cdots + \tilde{x}_n = |\mathcal{C}|.$$

Some of the constraints in the linear program in Theorem 8.4.3 are now easy to understand. We clearly have $\tilde{x}_0 = 1$, since every $\mathbf{w} \in \mathcal{C}$ has distance 0 only to itself. The equations $\tilde{x}_1 = 0$ through $\tilde{x}_{d-1} = 0$ hold by the assumption that $\mathcal{C}$ has distance d; that is, there are no pairs of code words with Hamming distance between 1 and $d-1$. Interestingly, this is the *only* place in the proof of the Delsarte bound where the assumption of $\mathcal{C}$ being a code with distance d is used.

The remaining set of constraints is considerably harder to derive, and it lacks a really intuitive explanation. Thus, to prove Theorem 8.4.3, we have to establish the following.

8.4.4 Proposition. *Let $\mathcal{C} \subseteq \{0,1\}^n$ be an arbitrary set, let $\tilde{x}_i = \tilde{x}_i(\mathcal{C})$ be defined as above, and let $t \in \{1, 2, \ldots, n\}$. Then we have the inequality*

$$\sum_{i=0}^{n} K_t(n,i) \cdot \tilde{x}_i \geq 0.$$

In the next lemma, $I \subseteq \{1,2,\ldots,n\}$ is a set of indices, and we write $d_H^I(\mathbf{w},\mathbf{w}')$ for the number of indices $i \in I$ with $w_i \neq w'_i$ (thus, the components outside I are ignored).

8.4.5 Lemma. *Let $I \subseteq \{1,2,\ldots,n\}$ be a set of indices, and let $\mathcal{C} \subseteq \{0,1\}^n$. Then the number of pairs $(\mathbf{w},\mathbf{w}') \in \mathcal{C}^2$ with $d_H^I(\mathbf{w},\mathbf{w}')$ even is at least as large as the number of pairs $(\mathbf{w},\mathbf{w}') \in \mathcal{C}^2$ with $d_H^I(\mathbf{w},\mathbf{w}')$ odd. (In probabilistic terms, if we choose $\mathbf{w},\mathbf{w}' \in \mathcal{C}$ independently at random, then the probability that they differ in an even number of positions from I is at least as large as the probability that they differ in an odd number of positions from I.)*

Proof. Let us write $|\mathbf{w}|_I = |\{i \in I : w_i = 1\}|$, and let us set

$$\mathcal{E} = \{\mathbf{w} \in \mathcal{C} : |\mathbf{w}|_I \text{ is even}\}, \quad \mathcal{O} = \{\mathbf{w} \in \mathcal{C} : |\mathbf{w}|_I \text{ is odd}\}.$$

From the equation $d_H^I(\mathbf{w},\mathbf{w}') = |\mathbf{w} \oplus \mathbf{w}'|_I$, we see that if $d_H^I(\mathbf{w},\mathbf{w}')$ is even, then $|\mathbf{w}|_I$ and $|\mathbf{w}'|_I$ have the same parity, and so $\mathbf{w}$ and $\mathbf{w}'$ are both in $\mathcal{E}$ or both in $\mathcal{O}$. On the other hand, for $d_H^I(\mathbf{w},\mathbf{w}')$ odd, one of $\mathbf{w}$, $\mathbf{w}'$ lies in $\mathcal{E}$ and the other one in $\mathcal{O}$. So the assertion of the lemma is equivalent to $|\mathcal{E}|^2 + |\mathcal{O}|^2 \geq 2 \cdot |\mathcal{E}| \cdot |\mathcal{O}|$, which follows by expanding $(|\mathcal{E}| - |\mathcal{O}|)^2 \geq 0$. □

8.4.6 Corollary. *For every $\mathcal{C} \subseteq \{0,1\}^n$ and every $\mathbf{v} \in \{0,1\}^n$ we have*

$$\sum_{(\mathbf{w},\mathbf{w}') \in \mathcal{C}^2} (-1)^{(\mathbf{w} \oplus \mathbf{w}')^T \mathbf{v}} \geq 0.$$

Proof. This is just another way of writing the statement of Lemma 8.4.5. Indeed, if we set $I = \{i : v_i = 1\}$, then $(\mathbf{w} \oplus \mathbf{w}')^T \mathbf{v} = d_H^I(\mathbf{w},\mathbf{w}')$, and hence the sum in the corollary is exactly the number of pairs $(\mathbf{w},\mathbf{w}')$ with $d_H^I(\mathbf{w},\mathbf{w}')$ even minus the number of pairs with $d_H^I(\mathbf{w},\mathbf{w}')$ odd.

The corollary also has a quick algebraic proof, which some readers may prefer. It suffices to note that $(\mathbf{w} \oplus \mathbf{w}')^T \mathbf{v}$ has the same parity as $(\mathbf{w} + \mathbf{w}')^T \mathbf{v}$ (addition modulo 2 was replaced by ordinary addition of integers), and so

$$\begin{aligned}
\sum_{(\mathbf{w},\mathbf{w}') \in \mathcal{C}^2} (-1)^{(\mathbf{w} \oplus \mathbf{w}')^T \mathbf{v}} &= \sum_{(\mathbf{w},\mathbf{w}') \in \mathcal{C}^2} (-1)^{(\mathbf{w} + \mathbf{w}')^T \mathbf{v}} \\
&= \sum_{(\mathbf{w},\mathbf{w}') \in \mathcal{C}^2} (-1)^{\mathbf{w}^T \mathbf{v}} \cdot (-1)^{\mathbf{w}'^T \mathbf{v}} \\
&= \left(\sum_{\mathbf{w} \in \mathcal{C}} (-1)^{\mathbf{w}^T \mathbf{v}} \right)^2 \geq 0.
\end{aligned}$$

□

Proof of Proposition 8.4.4. To prove the tth inequality in the proposition, i.e., $\sum_{i=0}^{n} K_t(n,i) \cdot \tilde{x}_i \geq 0$, we sum the inequality in Corollary 8.4.6 over all $\mathbf{v} \in \{0,1\}^n$ of weight t. Interchanging the summation order, we obtain

$$0 \leq \sum_{(\mathbf{w},\mathbf{w}') \in \mathcal{C}^2} \quad \sum_{\mathbf{v} \in \{0,1\}^n : |\mathbf{v}| = t} (-1)^{(\mathbf{w} \oplus \mathbf{w}')^T \mathbf{v}}.$$

To understand this last expression, let us fix $\mathbf{u} = \mathbf{w} \oplus \mathbf{w}'$ and write $S(\mathbf{u}) = \sum_{\mathbf{v} \in \{0,1\}^n : |\mathbf{v}| = t} (-1)^{\mathbf{u}^T \mathbf{v}}$. In this sum, the $\mathbf{v}$ with $\mathbf{u}^T \mathbf{v} = j$ are counted with sign $(-1)^j$. How many $\mathbf{v}$ of weight t and with $\mathbf{u}^T \mathbf{v} = j$ are there? Let $i = |\mathbf{u}| = d_H(\mathbf{w}, \mathbf{w}')$ be the number of 1's in $\mathbf{u}$. In order to form such a $\mathbf{v}$, we need to put j ones in positions where $\mathbf{u}$ has 1's and $t - j$ ones in positions where $\mathbf{u}$ has 0's. Hence the number of these $\mathbf{v}$ is $\binom{i}{j}\binom{n-i}{t-j}$, and

$$S(\mathbf{u}) = \sum_{j=0}^{\min(i,t)} (-1)^j \binom{i}{j} \binom{n-i}{t-j},$$

which we recognize as $K_t(n,i)$. So we have

$$0 \leq \sum_{(\mathbf{w},\mathbf{w}') \in \mathcal{C}^2} K_t(n, d_H(\mathbf{w}, \mathbf{w}')),$$

and it remains to note that the number of times $K_t(n,i)$ appears in this sum is $|\mathcal{C}| \cdot \tilde{x}_i$. This finishes the proof of Proposition 8.4.4 and thus also of Theorem 8.4.3. □

A small strengthening of the Delsarte bound. We have seen that Theorem 8.4.3 yields $A(17,3) \leq 6553$. We show how the inequalities in the theorem can be slightly strengthened using a parity argument, which leads to the best known upper bound $A(17,3) \leq 6552$. Similar tricks can improve the Delsarte bound in some other cases as well, but the improvements are usually minor.

For contradiction let us suppose that $n = 17$ and there is a code $\mathcal{C} \subseteq \{0,1\}^n$ of distance 3 with $|\mathcal{C}| = 6553$. The size of $\mathcal{C}$ is odd, and we note that for every code of odd size and every t, the last inequality in the proof of Corollary 8.4.6 can be strengthened to

$$\left(\sum_{\mathbf{w} \in \mathcal{C}} (-1)^{\mathbf{w}^T \mathbf{v}} \right)^2 \geq 1,$$

since an odd number of values from $\{-1, 1\}$ cannot sum to zero. If we propagate this improvement through the proof of Proposition 8.4.4, we arrive at the following inequality for the $\tilde{x}_i$:

$$\sum_{i=0}^{n} K_t(n,i) \cdot \tilde{x}_i \geq \frac{\binom{n}{t}}{|\mathcal{C}|}\,.$$

Since in our particular case we suppose $|\mathcal{C}| = 6553$, we can replace the constraints $\sum_{i=0}^{n} K_t(n,i) \cdot x_i \geq 0$, $t = 1, 2, \ldots, n$, in the linear program in Theorem 8.4.3 by $\sum_{i=0}^{n} K_t(n,i) \cdot x_i \geq \binom{n}{t}/6553$, and the $\tilde{x}_i$ defined by our $\mathcal{C}$ remain a feasible solution. However, the optimum of this modified linear program is only $6552\frac{3}{5}$, which contradicts the assumption $|\mathcal{C}| = 6553$. This proves $A(17,3) \leq 6552$.

The paper

> M. R. Best, A. E. Brouwer, F. J. MacWilliams, A. M. Odlyzko, and N. J. A. Sloane: Bounds for binary codes of length less than 25, *IEEE Trans. Inform. Theory* 24 (1978), pages 81–93.

describes this particular strengthening of the Delsarte bound and some similar approaches. A continually updated table of the best known bounds for $A(n,d)$ for small n and d is maintained by Andries Brouwer at

> `http://www.win.tue.nl/~aeb/codes/binary-1.html`.

The Delsarte bound explained. The result in Theorem 8.4.3 goes back to the thesis of Philippe Delsarte:

> P. Delsarte: An algebraic approach to the association schemes of coding theory, *Philips Res. Repts. Suppl.* 10 (1973).

Here we sketch Delsarte's original proof. At a comparable level of detail, his proof is more involved than the ad hoc proof above (from the paper by Best et al.). On the other hand, Delsarte's proof is more systematic, and even more importantly, it can be extended to prove a stronger result, which we mention below.

For $i \in \{0, \ldots, n\}$, let M_i be the $2^n \times 2^n$ matrix defined by[1]

$$(M_i)_{\mathbf{v},\mathbf{w}} = \begin{cases} 1 & \text{if } d_H(\mathbf{v},\mathbf{w}) = i \\ 0 & \text{otherwise.} \end{cases}$$

The set of matrices of the form

$$\sum_{i=0}^{n} y_i M_i, \quad y_0, \ldots, y_n \in \mathbb{R},$$

is known to be closed under addition and scalar multiplication (this is clear), and under matrix multiplication (this has to be shown). A set

[1] We assume that rows and columns are indexed by the words from $\{0,1\}^n$.

of matrices closed under these operations is called a **matrix algebra**. In our case, one speaks of the **Bose–Mesner algebra of the Hamming association scheme**. The matrix multiplication turns out to be commutative on this algebra, and this is known to imply a strong condition: The M_i have a *common* diagonalization, meaning that there is an orthogonal matrix U with $U^T M_i U$ diagonal for all i.

Once we know this, it is a matter of patience to find such a matrix U. For example, the matrix U defined by

$$U_{\mathbf{v},\mathbf{w}} = \frac{1}{2^{n/2}}(-1)^{\mathbf{v}^T\mathbf{w}}$$

will do. First we have to check (this is easy) that this matrix is indeed orthogonal, meaning that $U^T U = I_n$.

For the entries of $U^T M_i U$, we can derive the formula

$$(U^T M_i U)_{\mathbf{v},\mathbf{w}} = \frac{1}{2^n} \sum_{\substack{(\mathbf{u},\mathbf{u}')\in(\{0,1\}^n)^2 \\ d_H(\mathbf{u},\mathbf{u}')=i}} (-1)^{\mathbf{u}^T\mathbf{v}+\mathbf{u}'^T\mathbf{w}}.$$

We claim that this sum evaluates to 0 whenever $\mathbf{v} \neq \mathbf{w}$; this will imply that $U^T M_i U$ is indeed a diagonal matrix. To prove the claim, we let j be any index for which $v_j \neq w_j$. In the sum, we can then pair up the terms for $(\mathbf{u}, \mathbf{u}')$ and $(\mathbf{u} \oplus \mathbf{e}_j, \mathbf{u}' \oplus \mathbf{e}_j)$, with $\mathbf{e}_j$ being the word with a 1 exactly at position j. This pairing covers all terms of the sum, and paired-up terms are easily seen to cancel each other. (If you didn't believe $U^T U = I_n$, this can be shown with an even simpler pairing argument along these lines.)

On the diagonal, we get

$$\begin{aligned}(U^T M_i U)_{\mathbf{w},\mathbf{w}} &= \frac{1}{2^n} \sum_{\substack{(\mathbf{u},\mathbf{u}')\in(\{0,1\}^n)^2 \\ d_H(\mathbf{u},\mathbf{u}')=i}} (-1)^{(\mathbf{u}\oplus\mathbf{u}')^T\mathbf{w}} \\ &= \sum_{\substack{\mathbf{v}\in\{0,1\}^n \\ |\mathbf{v}|=i}} (-1)^{\mathbf{v}^T\mathbf{w}} = K_i(n, |\mathbf{w}|)\end{aligned}$$

(for the last equality see the proof of Proposition 8.4.4) since any $\mathbf{v}$ of weight i can be written in the form $\mathbf{v} = \mathbf{u} \oplus \mathbf{u}'$ in 2^n different ways, one for each $\mathbf{u} \in \{0,1\}^n$.

Next, let us fix a code $\mathcal{C}$ and look at a specific matrix in the Bose–Mesner algebra. For this, we define the values

$$\tilde{y}_i = \frac{|\{(\mathbf{w},\mathbf{w}') \in \mathcal{C}^2 : d_H(\mathbf{w},\mathbf{w}') = i\}|}{2^n \binom{n}{i}}, \quad i = 0, \ldots, n.$$

We note that $\tilde{y}_i$ is the probability that a randomly chosen pair of words with Hamming distance i is a pair of code words. Moreover, $\tilde{y}_i$ is related to our earlier quantity $\tilde{x}_i$ via

$$\tilde{y}_i = \frac{|\mathcal{C}|}{2^n \binom{n}{i}} \tilde{x}_i. \tag{8.7}$$

Here comes Delsarte's main insight.

8.4.7 Lemma. *The matrix* $\tilde{M} = \sum_{i=0}^{n} \tilde{y}_i M_i$ *is positive semidefinite.*

Proof. We first observe that

$$\tilde{M}_{\mathbf{v},\mathbf{w}} = \tilde{y}_i, \tag{8.8}$$

where $i = d_H(\mathbf{v}, \mathbf{w})$. We will express $\tilde{M}$ as a positive linear combination of matrices that are obviously positive semidefinite. We start with the matrix $X^{\mathcal{C}}$ defined by

$$X^{\mathcal{C}}_{\mathbf{v},\mathbf{w}} = \begin{cases} 1 & \text{if } (\mathbf{v}, \mathbf{w}) \in \mathcal{C}^2 \\ 0 & \text{otherwise.} \end{cases}$$

This matrix is positive semidefinite, since it can be written in the form

$$X^{\mathcal{C}} = \mathbf{x}^{\mathcal{C}} (\mathbf{x}^{\mathcal{C}})^T,$$

where $\mathbf{x}^{\mathcal{C}}$ is the characteristic vector of $\mathcal{C}$:

$$\mathbf{x}^{\mathcal{C}}_{\mathbf{w}} := \begin{cases} 1 & \text{if } \mathbf{w} \in \mathcal{C} \\ 0 & \text{otherwise.} \end{cases}$$

Let Π be the automorphism group of $\{0,1\}^n$, consisting of the $n!2^n$ bijections that permute indices and swap 0's with 1's at selected positions. With τ chosen uniformly at random from Π, we obtain[2]

$$\begin{aligned} \text{Prob}\Big[(\mathbf{v}, \mathbf{w}) \in \tau(\mathcal{C})^2\Big] &= \frac{1}{|\Pi|} \sum_{\pi \in \Pi} \Big[(\mathbf{v}, \mathbf{w}) \in \pi(\mathcal{C})^2\Big] \\ &= \frac{1}{|\Pi|} \sum_{\pi \in \Pi} X^{\pi(\mathcal{C})}_{\mathbf{v},\mathbf{w}}. \end{aligned}$$

On the other hand,

$$\text{Prob}\Big[(\mathbf{v}, \mathbf{w}) \in \tau(\mathcal{C})^2\Big] = \text{Prob}\Big[(\tau^{-1}(\mathbf{v}), \tau^{-1}(\mathbf{w})) \in \mathcal{C}^2\Big] = \tilde{y}_i,$$

since τ^{-1} is easily shown to map $(\mathbf{v}, \mathbf{w})$ to a *random* pair of words with Hamming distance i.

[2] The *indicator variable* $[P]$ of a statement P has value 1 if P holds and 0 otherwise.

Using (8.8), this shows that

$$\tilde{M} = \frac{1}{|\Pi|} \sum_{\pi \in \Pi} X^{\pi(\mathcal{C})}$$

is a positive linear combination of positive semidefinite matrices and is therefore positive semidefinite itself. The lemma is proved. □

After diagonalization of $\tilde{M}$ by the matrix U, the statement of Lemma 8.4.7 can equivalently be written as

$$\sum_{i=0}^{n} \tilde{y}_i (U^T M_i U)_{\mathbf{w},\mathbf{w}} = \sum_{i=0}^{n} \tilde{y}_i K_i(n, |\mathbf{w}|) \geq 0, \quad \mathbf{w} \in \{0,1\}^n.$$

This is true since diagonalization preserves the property of being positive semidefinite, which for diagonal matrices is equivalent to nonnegativity of all diagonal entries.

Taking into account the relation (8.7) between the $\tilde{y}_i$ and our original $\tilde{x}_i$, this implies the following inequalities for any code $\mathcal{C}$.

$$\sum_{i=0}^{n} \tilde{x}_i \frac{K_i(n,t)}{\binom{n}{i}} \geq 0, \quad t = 1, \ldots, n. \tag{8.9}$$

Observing that

$$\frac{\binom{t}{j}\binom{n-t}{i-j}}{\binom{n}{i}} = \frac{\binom{i}{j}\binom{n-i}{t-j}}{\binom{n}{t}},$$

we get (cf. the definition of K_t in Theorem 8.4.3)

$$\frac{K_i(n,t)}{\binom{n}{i}} = \frac{K_t(n,i)}{\binom{n}{t}}.$$

Under this equation, the inequalities in (8.9) are equivalent to those in Proposition 8.4.4 and we recover the Delsarte bound.

Beyond the Delsarte bound. Alexander Schrijver generalized Delsarte's approach and improved the upper bounds on $A(n,d)$ significantly in many cases:

> A. Schrijver: New code upper bounds from the Terwilliger algebra and semidefinite programming, *IEEE Trans. Inform. Theory* 51 (2005), pages 2859–2866.

His work uses semidefinite programming instead of linear programming.

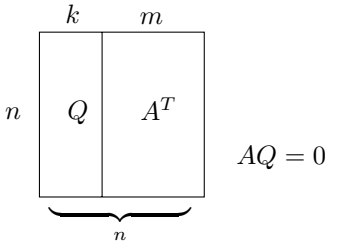
8.5 Sparse Solutions of Linear Systems

A coding problem. We begin with discussing error-correcting codes again, but this time we want to send a sequence $\mathbf{w} \in \mathbb{R}^k$ of k real numbers. Or rather not we, but a deep-space probe which needs to transmit its priceless measurements represented by $\mathbf{w}$ back to Earth. We want to make sure that all components of $\mathbf{w}$ can be recovered correctly even if some fraction, say 8%, of the transmitted numbers are corrupted, due to random errors or even maliciously (imagine that the secret *Brotherhood for Promoting the Only Truth* can somehow tamper with the signal slightly in order to document the presence of supernatural phenomena in outer space). We admit *gross errors*; that is, if the number 3.1415 is sent and it gets corrupted, it can be received as 2152.66, or 3.1425, or -10^{11}, or any other real number.

Here is a way of encoding $\mathbf{w}$: We choose a suitable number $n > k$ and a suitable $n\times k$ *encoding matrix* Q of rank k, and we send the vector $\mathbf{z} = Q\mathbf{w} \in \mathbb{R}^n$. Because of the errors, the received vector is not $\mathbf{z}$ but $\tilde{\mathbf{z}} = \mathbf{z} + \mathbf{x}$, where $\mathbf{x} \in \mathbb{R}^n$ is a vector with at most $r = \lfloor 0.08n \rfloor$ nonzero components. We ask, under what conditions on Q can $\mathbf{z}$ be recovered from $\tilde{\mathbf{z}}$?

Somewhat counterintuitively, we will concentrate on the task of finding the "error vector" $\mathbf{x}$. Indeed, once we know $\mathbf{x}$, we can compute $\mathbf{w}$ by solving the system of linear equations $Q\mathbf{w} = \mathbf{z} = \tilde{\mathbf{z}} - \mathbf{x}$. The solution, if one exists, is unique, since we assume that Q has rank k and hence the mapping $\mathbf{w} \mapsto Q\mathbf{w}$ is injective.

Sparse solutions of underdetermined linear systems. In order to compute $\mathbf{x}$, we first reformulate the recovery problem. Let $m = n-k$ and let A be an $m\times n$ matrix such that $AQ = 0$. That is, considering the k-dimensional linear subspace of $\mathbb{R}^n$ generated by the columns of Q, the rows of A form a basis of its orthogonal complement. The following picture illustrates the dimensions of the matrices:



In the recovery problem we have $\tilde{\mathbf{z}} = Q\mathbf{w} + \mathbf{x}$. Multiplying both sides by A from the left, we obtain $A\tilde{\mathbf{z}} = AQ\mathbf{w} + A\mathbf{x} = A\mathbf{x}$. Setting $\mathbf{b} = A\tilde{\mathbf{z}}$, we thus get that the unknown $\mathbf{x}$ has to satisfy the system of linear equations $A\mathbf{x} = \mathbf{b}$. We have $m = n - k$ equations and $n > m$ unknowns; the system is *underdetermined* and it has infinitely many solutions. In general, not all of these solutions can appear as an error vector in the decoding problem (we note that

the multiplication by A above is not necessarily an equivalent transformation and so it may give rise to spurious solutions). However, we seek a solution $\mathbf{x}$ with the extra property $|\mathrm{supp}(\mathbf{x})| \le r$, where we introduce the notation

$$\mathrm{supp}(\mathbf{x}) = \{i \in \{1, 2, \ldots, n\} : x_i \neq 0\}.$$

As we will see, under suitable conditions relating n, m, r and A, such a *sparse* solution of $A\mathbf{x} = \mathbf{b}$ turns out to be unique (and thus it has to be the desired error vector), and it can be computed efficiently by linear programming!

Let us summarize the resulting problem once again:

Sparse solution of underdetermined system of linear equations

Given an $m \times n$ matrix A with $m < n$, a vector $\mathbf{b} \in \mathbb{R}^m$, and an integer r, find an $\mathbf{x} \in \mathbb{R}^n$ such that

$$A\mathbf{x} = \mathbf{b} \quad \text{and} \quad |\mathrm{supp}(\mathbf{x})| \le r \tag{8.10}$$

if one exists.

The coding problem above is only one among several important practical problems leading to the computation of sparse solutions of underdetermined systems. We will mention other applications at the end of this section. From now on, we call any $\mathbf{x}$ satisfying (8.10) a **sparse solution** to $A\mathbf{x} = \mathbf{b}$. (Warning: This shouldn't be confused with solutions of *sparse systems* of equations, which is an even more popular topic in numerical mathematics and scientific computing.)

A linear algebra view. There is a simple necessary and sufficient condition guaranteeing that there is at most one sparse solution of $A\mathbf{x} = \mathbf{b}$.

8.5.1 Observation. *With n, m, r fixed, the following two conditions on an $m \times n$ matrix A are equivalent:*

(i) *The system $A\mathbf{x} = \mathbf{b}$ has at most one sparse solution $\mathbf{x}$ for every $\mathbf{b}$.*
(ii) *Every $2r$ or fewer columns of A are linearly independent.*

Proof. To prove the (more interesting) implication (ii)⇒(i), let us assume that $\mathbf{x}'$ and $\mathbf{x}''$ are two different sparse solutions of $A\mathbf{x} = \mathbf{b}$. Then $\mathbf{y} = \mathbf{x}' - \mathbf{x}'' \neq \mathbf{0}$ has at most $2r$ nonzero components and satisfies $A\mathbf{y} = A\mathbf{x}' - A\mathbf{x}'' = \mathbf{0}$, and hence it defines a linear dependence of at most $2r$ columns of A.

To prove (i)⇒(ii), we essentially reverse the above argument. Supposing that there exists nonzero $\mathbf{y} \in \mathbb{R}^n$ with $A\mathbf{y} = \mathbf{0}$ and $|\mathrm{supp}(\mathbf{y})| \le 2r$, we write $\mathbf{y} = \mathbf{x}' - \mathbf{x}''$, where both $\mathbf{x}'$ and $\mathbf{x}''$ have at most r nonzero components. For example, $\mathbf{x}'$ may agree with $\mathbf{y}$ in the first $\lfloor |\mathrm{supp}(\mathbf{y})|/2 \rfloor$ nonzero components and have 0's elsewhere, and $\mathbf{x}'' = \mathbf{x}' - \mathbf{y}$ has the remaining at most r nonzero components of $\mathbf{y}$ with opposite sign. We set $\mathbf{b} = A\mathbf{x}'$, so that $\mathbf{x}'$ is a sparse

solution of $A\mathbf{x} = \mathbf{b}$, and we note that $\mathbf{x}''$ is another sparse solution since $A\mathbf{x}'' = A\mathbf{x}' - A\mathbf{y} = A\mathbf{x}' = \mathbf{b}$. □

Let us note that (ii) implies that, in particular, $m \geq 2r$. On the other hand, if we choose a "random" $2r \times n$ matrix A, we almost surely have every $2r$ columns linearly independent.[3] So in the coding problem, if we set n so that $n = k + 2r$, choose A randomly, and let the columns of Q form a basis of the orthogonal complement of the row space of A, we seem to be done—a random A has almost surely every $2r$ columns linearly independent, and in such case, assuming that no more than r errors occurred, the sparse error vector $\mathbf{x}$ is always determined uniquely, and so is the original message $\mathbf{w}$.

Efficiency? But a major question remains—how can we *find* the unknown sparse solution $\mathbf{x}$? Unfortunately, it turns out that the problem of computing a sparse solution of $A\mathbf{x} = \mathbf{b}$ is difficult (NP-hard) in general, even for A satisfying the conditions of Observation 8.5.1.

Since the problem of finding a sparse solution of $A\mathbf{x} = \mathbf{b}$ is important and computationally difficult, several heuristic methods have been proposed for solving it at least approximately and at least in some cases. One of them, described next, turned out to be considerably more powerful than the others.

Basis pursuit. A sparse solution $\mathbf{x}$ is "small" in the sense of having few nonzero components. The idea is to look for $\mathbf{x}$ that is "small" in another sense that is easier to deal with, namely, with small $|x_1| + |x_2| + \cdots + |x_n|$. The last quantity is commonly denoted by $\|\mathbf{x}\|_1$ and called the $\boldsymbol{\ell_1}$**-norm** of $\mathbf{x}$ (while $\|\mathbf{x}\| = \|\mathbf{x}\|_2 = \sqrt{x_1^2 + \cdots + x_n^2}$ is the usual Euclidean norm, which can also be called the ℓ_2-norm).[4] We thus arrive at the following optimization problem (usually called *basis pursuit* in the literature):

$$\text{Minimize } \|\mathbf{x}\|_1 \text{ subject to } \mathbf{x} \in \mathbb{R}^n \text{ and } A\mathbf{x} = \mathbf{b}. \qquad \text{(BP)}$$

[3] In this book we don't want to assume or introduce the knowledge required to state and prove this claim rigorously. Instead, we offer the following semiformal argument relying on a famous but nontrivial theorem. The condition of linear independence of every $2r$ columns can be reformulated as $\det(A_I) \neq 0$ for every $2r$-element $I \subset \{1, 2, \ldots, n\}$. Now for I fixed, $\det(A_I)$ is a polynomial of degree $2r$ in the $2rn$ entries of A (it really depends only on $4r^2$ entries but never mind), and the set of the matrices A with $\det(A_I) = 0$ is the zero set of this polynomial in $\mathbb{R}^{2rn}$. The zero set of any nonzero polynomial is very "thin"; by Sard's theorem, it has Lebesgue measure 0. Hence the matrices A with $\det(A_I) = 0$ for at least one I correspond to points in $\mathbb{R}^{2nr}$ lying on the union of $\binom{n}{2r}$ zero sets, each of measure 0, and altogether they have measure 0. Therefore, such matrices appear with zero probability in any "reasonable" continuous distribution on matrices, for example, if the entries of A are chosen independently and uniformly from the interval $[-1, 1]$.

[4] The letter ℓ here can be traced back to the surname of Henri Lebesgue, the founder of modern integration theory. A certain space of integrable functions on $[0, 1]$ is denoted by $L_1(0, 1)$ in his honor, and ℓ_1 is a "pocket version" of this space consisting of countable sequences instead of functions.

By a trick we have learned in Section 2.4, this problem can be reformulated as a linear program:

$$\begin{array}{ll} \text{Minimize} & u_1 + u_2 + \cdots + u_n \\ \text{subject to} & A\mathbf{x} = \mathbf{b} \\ & -\mathbf{u} \le \mathbf{x} \le \mathbf{u} \\ & \mathbf{x}, \mathbf{u} \in \mathbb{R}^n,\ \mathbf{u} \ge \mathbf{0}. \end{array} \qquad (\mathrm{BP}')$$

To check the equivalence of (BP) and (BP′), we just note that in an optimal solution of (BP′) we have $u_i = |x_i|$ for every i.

The basis pursuit approach to finding a sparse solution of $A\mathbf{x} = \mathbf{b}$ thus consists in computing an optimal solution $\mathbf{x}^*$ of (BP) by linear programming, and hoping that, with some luck, this $\mathbf{x}^*$ might also be the sparse solution or at least close to it.

At first sight it is not clear why basis pursuit should have any chance of finding a sparse solution. After all, the desired sparse solution might have a few huge components, while $\mathbf{x}^*$, a minimizer of the ℓ_1-norm, might have all components nonzero but tiny.

Surprisingly, experiments have revealed that basis pursuit actually performs excellently, and it usually finds the sparse solution exactly even in conditions that don't look very favorable. Later these findings were confirmed by rather precise theoretical results. Here we state the following particular case of such results:

8.5.2 Theorem (Guaranteed success of basis pursuit). *Let*

$$m = \lfloor 0.75n \rfloor,$$

and let A be a random $m \times n$ matrix, where each entry is drawn from the standard normal distribution $N(0,1)$ and the entries are mutually independent.[5] *Then with probability at least $1 - e^{-cm}$, where $c > 0$ is a positive constant, the matrix A has the following property:*

> *If $\mathbf{b} \in \mathbb{R}^m$ is such that the system $A\mathbf{x} = \mathbf{b}$ has a solution $\tilde{\mathbf{x}}$ with at most $r = \lfloor 0.08n \rfloor$ nonzero components, then $\tilde{\mathbf{x}}$ is a unique optimal solution of (BP).*

For brevity, we call a matrix A with the property as in the theorem **BP-exact** (more precisely, we should say "BP-exact for r," where r specifies the maximum number of nonzero components). For a BP-exact matrix A we can

[5] We recall that the distribution $N(0,1)$ has density given by the Gaussian "bell curve" $\frac{1}{\sqrt{2\pi}} e^{-x^2/2}$. How can we generate a random number with this distribution? This is implemented in many software packages, and methods for doing it can be found, for instance, in

D. Knuth: *The Art of Computer Programming*, Vol. 2: Seminumerical Algorithms, Addison-Wesley, Reading, Massachusetts, 1973.

thus find a sparse solution of $A\mathbf{x} = \mathbf{b}$ exactly and efficiently, by solving the linear program (BP′).

Returning to the coding problem from the beginning of the section, we immediately obtain the following statement:

8.5.3 Corollary. *Let k be a sufficiently large integer, let us set $n = 4k$, $m = 3k$, let a random $m \times n$ matrix A be generated as in Theorem 8.5.2, and let Q be an $n \times k$ matrix of rank k with $AQ = 0$ (in other words, the column space of Q is the orthogonal complement of the row space of A). Then the following holds with probability overwhelmingly close to 1: If Q is used as a coding matrix to transmit a vector $\mathbf{w} \in \mathbb{R}^k$, by sending the vector $\mathbf{z} = Q\mathbf{w} \in \mathbb{R}^n$, then even if any at most 8% of the entries of $\mathbf{z}$ are corrupted, we can still reconstruct $\mathbf{w}$ exactly and efficiently, by solving the appropriate instance of (BP′).*

Drawing the elements of A from the standard normal distribution is not the only known way of generating a BP-exact matrix. Results similar to Theorem 8.5.2, perhaps with worse constants, can be proved by known techniques for random matrices with other distributions. The perhaps simplest such distribution is obtained by choosing each entry to be $+1$ or -1, each with probability $\frac{1}{2}$ (and again with all entries mutually independent).

A somewhat unpleasant feature of Theorem 8.5.2 and of similar results is that they provide a BP-exact matrix only with high probability. No efficient method for *verifying* that a given matrix is BP-exact is known at present, and so we cannot be absolutely sure. In practice this is not really a problem, since the probability of failure (i.e., of generating a matrix that is not BP-exact) can be made very small by choosing the parameters appropriately, much smaller than the probability of sudden death of all people in the team that wants to compute the sparse solution, for instance. Still, it would be nice to have explicit constructions of BP-exact matrices with good parameters.

Random errors. In our coding problem, we allow for completely arbitrary (even malicious) errors; all we need is that there aren't too many errors. However, in practice one may often assume that the errors occur at random positions, and we want to be able to decode correctly only with high probability, that is, for most (say 99.999%) of the $\binom{n}{r}$ possible locations of the r errors. It turns out that considerably stronger numerical bounds can be obtained in this setting: For example, Theorem 8.5.2 tells us that for $m = \lfloor 0.75n \rfloor$ and $k = n - m$, we are guaranteed to fix *any* $0.08n$ errors with high probability, but it turns out that we can also fix *most* of the possible r-tuples of errors for r as large as $0.36n$! For a precise statement see the paper of Donoho quoted near the end of the section.

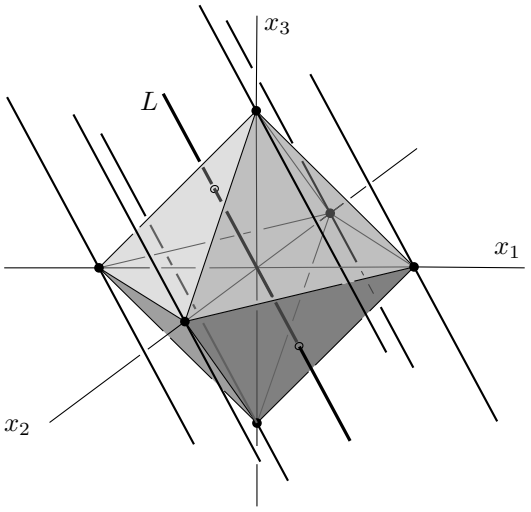
Geometric meaning of BP-exactness. Known proofs of Theorem 8.5.2 or similar results use a good deal of geometric probability and high-dimensional geometry, knowledge which we want neither to assume nor to introduce in this book. We thus have to omit a proof. Instead, we present an appealing geometric characterization of BP-exact matrices, which is a starting point of existing proofs.

For its statement we need to recall the *crosspolytope*, a convex polytope already mentioned in Section 4.3. We will denote the n-dimensional crosspolytope by B_1^n, which should suggest that it is the unit ball of the ℓ_1-norm:

$$B_1^n = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\|_1 \le 1\}.$$

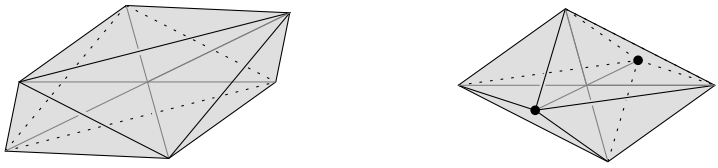
8.5.4 Lemma (Reformulation of BP-exactness). *Let A be an $m\times n$ matrix, $m < n$, let $r \le m$, and let $L = \{\mathbf{x} \in \mathbb{R}^n : A\mathbf{x} = \mathbf{0}\}$ be the kernel (null space) of A. Then A is BP-exact for r if and only if the following holds: For every $\mathbf{z} \in \mathbb{R}^n$ with $\|\mathbf{z}\|_1 = 1$ (i. e., $\mathbf{z}$ is a boundary point of the crosspolytope) and $|\mathrm{supp}(\mathbf{z})| \le r$, the affine subspace $L+\mathbf{z}$ intersects the crosspolytope only at $\mathbf{z}$; that is, $(L+\mathbf{z}) \cap B_1^n = \{\mathbf{z}\}$.*

Let us discuss an example with $n = 3$, $m = 2$, and $r = 1$, about the only sensible setting for which one can draw a picture. If the considered 2×3 matrix A has full rank, which we may assume, then the kernel L is a one-dimensional linear subspace of $\mathbb{R}^3$, that is, a line passing through the origin. The points $\mathbf{z}$ coming into consideration have at most $r = 1$ nonzero coordinate, and they lie on the boundary of the regular octahedron B_1^3, and hence they are precisely the 6 vertices of B_1^3:



The line L through the origin is drawn thick, and the condition in the lemma says that each of the 6 translates of L to the vertices should only touch the

crosspolytope. Another way of visualizing this is to look at the projection of B_1^3 to the plane orthogonal to L. Each of the translates of L is projected to a point, and the crosspolytope is projected to a convex polygon. The condition then means that all the 6 vertices should appear on the boundary in the projection, as in the left picture below,



while the right picture corresponds to a bad L (the condition is violated at the two vertices marked by dots that lie inside the projection). In general, of course, L is not a line but a k-dimensional linear subspace of $\mathbb{R}^n$, and the considered points $\mathbf{z}$ are not only vertices of B_1^n, but they can lie in all $(r-1)$-dimensional faces of B_1^n. Indeed, we note that the points $\mathbf{z}$ on the surface of B_1^n with at most r nonzero components are exactly the points of the union of all $(r-1)$-dimensional faces, omitting the easy proof of this fact (but look at least at the case $n = 3$, $r = 2$).

Proof of Lemma 8.5.4. First we assume that A is BP-exact, we consider a point $\mathbf{z}$ with $\|\mathbf{z}\|_1 = 1$ and $|\mathrm{supp}(\mathbf{z})| \leq r$, and we set $\mathbf{b} = A\mathbf{z}$. Then the system $A\mathbf{x} = \mathbf{b}$ has a sparse solution, namely $\mathbf{z}$, and hence $\mathbf{z}$ has to be the unique point among all solutions of $A\mathbf{x} = \mathbf{b}$ that minimize the ℓ_1-norm. Noting that the set of all solutions of $A\mathbf{x} = \mathbf{b}$ is exactly the affine subspace $L + \mathbf{z}$, we get that $\mathbf{z}$ is the only point in $L + \mathbf{z}$ with ℓ_1-norm at most 1. That is, $(L + \mathbf{z}) \cap B_1^n = \{\mathbf{z}\}$ as claimed.

Conversely, we assume that L satisfies the condition in the lemma and we consider $\mathbf{b} \in \mathbb{R}^m$. Let us suppose that the system $A\mathbf{x} = \mathbf{b}$ has a solution $\tilde{\mathbf{x}}$ with at most r nonzero components. If $\tilde{\mathbf{x}} = \mathbf{0}$, then $\mathbf{b} = \mathbf{0}$, and clearly, $\mathbf{0}$ is also the only optimum of (BP). For $\tilde{\mathbf{x}} \neq \mathbf{0}$, we set $\mathbf{z} = \frac{\tilde{\mathbf{x}}}{\|\tilde{\mathbf{x}}\|_1}$. Then $\|\mathbf{z}\|_1 = 1$ and $|\mathrm{supp}(\mathbf{z})| \leq r$, and so by the assumption, $\mathbf{z}$ is the only point in $L + \mathbf{z}$ of ℓ_1-norm at most 1. By rescaling we get that $\tilde{\mathbf{x}}$ is the only point in $L + \tilde{\mathbf{x}}$ of ℓ_1-norm at most $\|\tilde{\mathbf{x}}\|_1$, and since $L + \tilde{\mathbf{x}}$ is the set of all solutions of $A\mathbf{x} = \mathbf{b}$, we get that A is BP-exact. □

Intuition for BP-exactness. We don't have means for proving Theorem 8.5.2, but now, using the lemma just proved, we can at least try to convey some intuition as to why a claim like Theorem 8.5.2 is plausible, and what kind of calculations are needed to prove it.

The kernel L of a random matrix A defines a random k-dimensional subspace[6] of $\mathbb{R}^n$, where $k = n-m$. For proving Theorem 8.5.2, we need

[6] The question, "What is a random k-dimensional subspace?" is a subtle one. For us, the simplest way out is to define a random k-dimensional subspace as the

to verify that L is *good* for every boundary point $\mathbf{z}$ of the crosspolytope with $|\mathrm{supp}(\mathbf{z})| \le r$, where we say that L is good for $\mathbf{z}$ if $(L+\mathbf{z}) \cap B_1^n = \{\mathbf{z}\}$.

For $\mathbf{z}$ as above, let us define a convex cone $C_{\mathbf{z}} = \{t(\mathbf{x}-\mathbf{z}) : t \ge 0,\ \mathbf{x} \in B_1^n\}$. Geometrically, we take the cone generated by all rays emanating from $\mathbf{z}$ and intersecting the crosspolytope in a point other than $\mathbf{z}$, and we translate the cone so that $\mathbf{z}$ is moved to the origin. Then L good for $\mathbf{z}$ means exactly that $L \cap C_{\mathbf{z}} = \{\mathbf{0}\}$.

The points $\mathbf{z}$ on the boundary with at most r nonzero coordinates fill out exactly the union of all $(r-1)$-dimensional faces of the crosspolytope. Let F be one of these faces. It can be checked that the cone $C_{\mathbf{z}}$ is the same for all $\mathbf{z}$ in the relative interior of F, so we can define the cone C_F associated with the face F (the reader may want to consider some examples for the 3-dimensional regular octahedron). Moreover, if $\mathbf{y}$ is a boundary point of F, then $C_{\mathbf{y}} \subseteq C_F$, and so if L is good for some point in the relative interior of F, then it is good for all points of F including the boundary.

Let p_F denote the probability that a random L is *bad* (i.e., not good) for some $\mathbf{z} \in F$. Then the probability that L is bad for any $\mathbf{z}$ at all is no more than $\sum_F p_F$, where the sum is over all $(r-1)$-dimensional faces of the crosspolytope.

It is not too difficult to see that the number of $(r-1)$-dimensional faces is $\binom{n}{r}2^r$, and that the cones C_F of all of these faces are congruent (they differ only by rotation around the origin). Therefore, all p_F equal the same number $p = p(n,k,r)$, and the probability of L bad for at least one $\mathbf{z}$ is at most $\binom{n}{r}2^r p$. If we manage to show, for some values of n, k, and r, that the expression $\binom{n}{r}2^r p$ is much smaller than 1, then we can conclude that a random matrix A is BP-exact with high probability.

Estimating $p(n,k,r)$ is a nontrivial task; its difficulty heavily depends on the accuracy we want to attain. Getting an estimate that is more or less accurate including numerical constants, such as is needed to prove Theorem 8.5.2, is quite challenging. On the other hand, if we don't care about numerical constants so much and want just a rough asymptotic result, standard methods from high-dimensional convexity theory lead to the goal much faster.

---

kernel of a random $m \times n$ matrix with independent normal entries as in Theorem 8.5.2. Fortunately, this turns out to be equivalent to the usual (and "right") definition, which is as follows. One fixes a particular k-dimensional subspace R_0, say the span of the first k vectors of the standard basis, and defines a random subspace as a random rotation of R_0. This may not sound like great progress, since we have just used the equally problematic-looking notion of random rotation. But the group $\mathrm{SO}(n)$ of all rotations in $\mathbb{R}^n$ around the origin is a compact group and hence it has a unique invariant probability measure (Haar measure), which defines "random rotation" satisfactorily and uniquely.

Here we conclude this very rough outline of the argument with a few words on why one should expect $p(n,k,r)$ to be very small for k and r much smaller than n. Roughly speaking, this is because for n large, the n-dimensional crosspolytope is a very lean and spiky body, and the cones C_F are very narrow for low-dimensional faces F. Hence a random subspace L of not too large dimension is very likely to avoid C_F. As a very simplified example of this phenomenon, we may consider $k = r = 1$. Then F is a vertex and C_F is easily described. As a manageable exercise, the reader may try to estimate the fraction of the unit sphere centered at $\mathbf{0}$ that is covered by C_F; this quantity is exactly half of the probability $p(n,1,1)$ that a random line through $\mathbf{0}$ intersects C_F nontrivially.

References. Basis pursuit was introduced in

S. Chen, D. L. Donoho, and M. A. Saunders: Atomic decomposition by basis pursuit, *SIAM J. Scientific Computing* 20, 1(1999) 33–61.

A classical approach to finding a "good" solution of an underdetermined system $A\mathbf{x} = \mathbf{b}$ would be to minimize $\|\mathbf{x}\|_2$, rather than $\|\mathbf{x}\|_1$ (a "least squares" or "generalized inverse" method), which typically yields a solution with many nonzero components and is much less successful in applications such as the decoding problem. Basis pursuit, by minimizing the ℓ_1-norm instead, yields a basic solution with only a few nonzero components.

Interestingly, several groups of researchers independently arrived at the concept of BP-exactness and obtained the following general version of Theorem 8.5.2: *For every constant* $\alpha \in (0,1)$ *there exists* $\beta = \beta(\alpha) > 0$ *such that a random* $\lfloor \alpha n \rfloor \times n$ *matrix* A *is BP-exact for* $r = \lfloor \beta n \rfloor$ *with probability exponentially close to* 1. Combined results of two of these groups can be found in

E. J. Candès, M. Rudelson, T. Tao, and R. Vershynin: Error correction via linear programming, *Proc. 46th IEEE Symposium on Foundations of Computer Science* (FOCS), 2005, pages 295–308.

A third independent proof was given by Donoho. Later, and by yet another method, he obtained the strongest known quantitative bounds, including those in Theorem 8.5.2 (the previously mentioned proofs yield a much smaller constant for $\alpha = 0.75$ than 0.08). Among his several papers on the subject we cite

D. Donoho: High-dimensional centrally symmetric polytopes with neighborliness proportional to dimension, *Discrete and Computational Geometry* 35(2006), 617–652,

where he establishes a connection to the classical theory of convex polytopes using a result in the spirit of Lemma 8.5.4 and the remarks following it (but more elaborate). Through this connection he obtained an interesting *upper* bound on $\beta(\alpha)$. For example, in the setting of Theorem 8.5.2, there exists no $\lfloor 0.75n \rfloor \times n$ BP-exact matrix at all with $r > 0.25n$ (assuming n large). Additional upper bounds, essentially showing the existence results for BP-matrices in the above papers to be asymptotically optimal, were proved by

> N. Linial and I. Novik: How neighborly can a centrally symmetric polytope be?, *Discr. Comput. Geom.*, in press.

We remark that our notation is a compromise among the notations of the papers quoted above and doesn't follow any of them exactly, and that the term "BP-exact" is ours.

More applications of sparse solutions of underdetermined systems. The problem of computing a sparse solution of a system of linear equations arises in *signal processing*. The signal considered may be a recording of a sound, a measurement of seismic waves, a picture taken by a digital camera, or any of a number of other things. A classical method of analyzing signals is *Fourier analysis*, which from a linear-algebraic point of view means expressing a given periodic function in a basis consisting of the functions 1, $\cos x$, $\sin x$, $\cos 2x$, $\sin 2x$, $\cos 3x$, $\sin 3x$,... (the closely related *cosine transform* is used in the *JPEG encoding* of digital pictures). These functions are linearly independent, and so the expression (Fourier series) is unique. In the more recent *wavelet* analysis[7] one typically has a larger collection of basic functions, the wavelets, which can be of various kinds, depending on the particular application. They are no longer linearly independent, and hence there are many different ways of representing a given signal as a linear combination of wavelets. So one looks for a representation satisfying some additional criteria, and sparsity (small number of nonzero coefficients) is a very natural criterion: It leads to an economic (compressed) representation, and sometimes it may also help in analyzing or filtering the signal. For example, let us imagine that there is a smooth signal that has a nice representation by sine and cosine functions, and then an impulsive noise made of "spike" functions is added to it. We let the basic functions be sines and cosines *and* suitable spike functions, and by computing a sparse representation in such a basis we can often isolate the noise component very well, something that the classical Fourier analysis cannot do. Thus, we naturally arrive at computing sparse solutions of underdetermined linear systems.

Another source is computer tomography (CT), where one has an unknown vector $\mathbf{x}$ (each x_i is the density of some small area of tissue,

[7] Indeed, the newer picture encoding standard JPEG 2000 employs wavelets.

say), and the CT scanner measures various linear combinations of the x_i, corresponding to rays through the tissue in various directions. Sometimes there are reasons to expect that only a small number of the pixels will have values x_i different from the background level, and when we want to reconstruct $\mathbf{x}$ from the scan, we again ask for a sparse solution of a linear system. (More realistically, although less intuitively, we don't expect a small number of nonzero *pixels*, but rather a small number of significantly nonzero coefficients in a suitable wavelet representation.)

8.6 Transversals of d-Intervals

This section describes an application of the duality theorem in discrete geometry and combinatorics. We begin with a particular geometric result. Then we discuss concepts appearing in the proof in a more general context.

Helly's and Gallai's theorems. First let $\mathcal{I} = \{I_1, I_2, \ldots, I_n\}$ be a family of closed intervals on the real line such that every two of the intervals intersect. It is easily seen that there exists a point common to all of the intervals in $\mathcal{I}$: Indeed, the rightmost among the left endpoints of the I_i is such a point. In more detail, writing $I_i = [a_i, b_i]$ and setting $a = \max\{a_1, \ldots, a_n\}$, we necessarily have $a_i \le a \le b_i$ for all i, since $a_i \le a$ is immediate from the definition of a, and if we had $a > b_i$ for some i, then $I_i = [a_i, b_i]$ would be disjoint from the interval beginning with a.

The statement just proved is a special (one-dimensional) case of a beautiful and important theorem of Helly: *If $C_1, C_2, \ldots, C_n$ are convex sets in $\mathbb{R}^d$ such that any at most $d+1$ of them have a point in common, then there is a point common to all of the C_i.* We will not prove this result; it is mentioned as a background against which the forthcoming results can be better appreciated.

It is easily seen that in general we cannot replace $d+1$ by any smaller number in Helly's theorem. For example, in the plane, the assumption of Helly's theorem requires every three of the sets to have a common point, and pairwise intersections are not enough. To see this, we consider n lines in general position. They are convex sets, every two of them intersect, but no three have a common point.

Let us now consider planar convex sets of a special kind, namely, circular disks. One can easily draw three disks such that every two intersect, but no point is common to all three. However, there is a theorem (Gallai's) for pairwise intersecting disks in the spirit of Helly's theorem: *If $\mathcal{D} = \{D_1, D_2, \ldots, D_n\}$ is a family of disks in the plane in which every two disks intersect, then there exist 4 points such that each D_i contains at least*

one of them. With the (best possible) constant 4 this is a quite difficult theorem, but it is not too hard to prove a similar theorem with 4 replaced by some large constant. The reader is invited to solve this as a puzzle.

A set of points as in the theorem that intersects every member of $\mathcal{D}$ is called a **transversal** of $\mathcal{D}$ (sometimes one also speaks of *piercing* or *stabbing* $\mathcal{D}$ by a small number of points). Thus pairwise intersecting disks in the plane always have a 4-point transversal, and Helly's theorem asserts that $(d+1)$-wise intersecting convex sets in $\mathbb{R}^d$ have a one-point transversal.

What conditions on a family of sets guarantee that it has a small transversal? This fairly general question subsumes many interesting particular problems and it has been the subject of much research. Here we consider a one-dimensional situation. At the beginning of the section we dealt with a family of intervals, and now we admit intervals with some bounded number of "holes."

Transversals for pairwise intersecting *d*-intervals. For an integer $d \geq 1$, a ***d*-interval** is defined as the union of d closed intervals on the real line. The following picture shows three pairwise intersecting 2-intervals (drawn solid, dashed, and dash-dotted, respectively) with no point common to all three:

Thus, we cannot expect a one-point transversal for pairwise intersecting d-intervals. But the following theorem shows the existence of a transversal whose size depends only on d:

8.6.1 Theorem. *Let $\mathcal{J}$ be a finite family of d-intervals such that $J_1 \cap J_2 \neq \emptyset$ for every $J_1, J_2 \in \mathcal{J}$. Then $\mathcal{J}$ has a transversal of size $2d^2$; that is, there exist $2d^2$ points such that each d-interval of $\mathcal{J}$ contains at least one of them.*

At first sight it is not obvious that there is any bound at all for the size of the transversal that depends only on d. This was first proved in 1970, with a bound exponential in d, in

> A. Gyárfás, J. Lehel: A Helly-type problem in trees, in *Combinatorial Theory and its Applications*, P. Erdős, A. Rényi, and V.T. Sós, editors, North-Holland, Amsterdam, 1970, pages 571–584.

The best bound known at present is d^2, and it has been established using algebraic topology in

> T. Kaiser: Transversals of d-intervals, *Discrete Comput. Geom.* 18(1997) 195–203.

We are going to prove a bound that is worse by a factor of 2, but the proof presented here is much simpler. It comes from

N. Alon: Piercing d-intervals, *Discrete Comput. Geom.* 19(1998) 333–334,

following a general method developed in

N. Alon, D. Kleitman: Piercing convex sets and the Hadwiger–Debrunner (p, q)-problem, *Adv. Math.* 96(1992) 103–112.

It is also known that the bound cannot be improved below a constant multiple of $\frac{d^2}{\log d}$; see

J. Matoušek: Lower bounds on the transversal numbers of d-intervals, *Discrete Comput. Geom.* 26(2001) 283–287.

Working toward a proof of the theorem, we first show that in a family of pairwise intersecting d-intervals, some point has to be contained in "many" members of the family. For reasons that will become apparent later, we formulate it for a finite *sequence* of d-intervals, so that repetitions are allowed.

8.6.2 Lemma. *Let $J_1, J_2, \ldots, J_n$ be d-intervals such that $J_i \cap J_j \neq \emptyset$ for all $i, j \in \{1, 2, \ldots, n\}$. Then there is an endpoint of some J_i that is contained in at least $n/2d$ of the d-intervals.*

Proof. Let T be the set of all ordered triples (p, i, j) such that p is one of the at most d *left* endpoints of J_i, and $p \in J_j$. We want to bound the size of T from below. Let us fix $i \leq j$ for a moment. Let p be the leftmost point of the (nonempty) intersection $J_i \cap J_j$. Clearly, p is a left endpoint of J_i or J_j, and thus T contains one of the triples (p, i, j) or (p, j, i). So every pair i, j, $i \leq j$, contributes at least one member of T, and thus $|T| \geq \binom{n}{2} + n \geq n^2/2$. Since there are at most dn possible values for the pair (p, i), it follows that there exist p_0 and i_0 such that $(p_0, i_0, j) \in T$ for at least $(n^2/2)/dn = n/2d$ values of j. This means that p_0 is contained in at least $n/2d$ of the J_j. □

Next, we show that for every family of pairwise intersecting d-intervals we can distribute weights on the endpoints such that every d-interval has relatively large weight (compared to the total weight of all points).

8.6.3 Lemma. *Let $\mathcal{J}$ be a finite family of pairwise intersecting d-intervals, and let P denote the set of endpoints of the d-intervals in $\mathcal{J}$. Then there are nonnegative real numbers x_p, $p \in P$, such that*

$$\begin{aligned} \sum_{p \in J \cap P} x_p &\geq 1 \text{ for every } J \in \mathcal{J}, \text{ and} \\ \sum_{p \in P} x_p &\leq 2d. \end{aligned}$$

Before proving the lemma, let us see how it implies Theorem 8.6.1. Given $\mathcal{J}$ and the weights as in the lemma, we will choose a set $X \subseteq P$ such that $|X| \le 2d^2$, and each of the $|X|+1$ open intervals into which X divides the real axis contains points of P of total weight less than $1/d$. Such an X can be selected from P by a simple left-to-right scan: Let $p_1 < p_2 < \cdots < p_m$ be the points of P. We consider them one by one in this order, and we include p_i into X whenever $\sum_{j:\, p_\ell < p_j \le p_i} x_{p_j} \ge \frac{1}{d}$, where p_ℓ is the last point already included in X (and we formally put $p_\ell = -\infty$ if no point has yet been included in X). It is clear that none of the open intervals determined by X contains weight $1/d$ or larger, and the bound on the size of X follows easily, using that the total weight of all points of P is at most $2d$.

We claim that X is a transversal of $\mathcal{J}$. Indeed, considering a $J \in \mathcal{J}$, at least one of the d components of J is an interval containing points of P of total weight at least $1/d$, and so it contains a point of X.

Proof of Lemma 8.6.3 by duality. We formulate the problem of choosing the weights x_p as a linear program with variables x_p, $p \in P$:

$$\begin{array}{ll} \text{Minimize} & \sum_{p\in P} x_p \\ \text{subject to} & \sum_{p\in J\cap P} x_p \ge 1 \text{ for every } J \in \mathcal{J}, \\ & \mathbf{x} \ge \mathbf{0}. \end{array}$$

The linear program is certainly feasible; for instance, $x_p = 1$ for all p is a feasible solution. We would like to show that the optimum is at most $2d$.

Using the dualization recipe from Section 6.2, we find that the dual linear program has variables y_J, $J \in \mathcal{J}$, and it looks as follows:

$$\begin{array}{ll} \text{Maximize} & \sum_{J\in\mathcal{J}} y_J \\ \text{subject to} & \sum_{J:\, p\in J\cap P} y_J \le 1 \text{ for every } p \in P, \\ & \mathbf{y} \ge \mathbf{0}. \end{array}$$

The dual linear program is feasible, too, since $\mathbf{y} = \mathbf{0}$ is feasible. Then by the duality theorem both the primal and the dual linear programs have optimal solutions, $\mathbf{x}^*$ and $\mathbf{y}^*$, respectively, that yield the same value of the objective functions.

We may assume that $\mathbf{y}^*$ is rational: Indeed, we may take it to be a basic feasible solution, and all basic feasible solutions are rational since all coefficients of the linear program are rational.

We now have some rational weight y^*_J for every $J \in \mathcal{J}$ such that no point of P is contained in d-intervals of total weight exceeding 1, and we want to show that the total weight of all $J \in \mathcal{J}$ cannot be larger than $2d$.

Lemma 8.6.2 tells us that if all the d-intervals had the same weight, then there would be a point contained in d-intervals of weight at least $W/2d$, where W is the total weight of all d-intervals. Our weights need not all be equal, but we will pass to the case of equal weights by replicating each d-interval a suitable number of times.

Let D be a common denominator of all the rational numbers y_J^*, $J \in \mathcal{J}$, and let $y_J^* = \frac{r_J}{D}$, with r_J integral. Let $(J_1, J_2, \ldots, J_n)$ be a sequence that includes each d-interval $J \in \mathcal{J}$ exactly r_J times (thus $n = \sum_{J\in\mathcal{J}} r_J$). By Lemma 8.6.2 there is a point $p \in P$ contained in at least $n/2d$ members of the sequence, which means that

$$\sum_{J\in\mathcal{J}:\, p\in J} r_J \geq \frac{n}{2d} = \frac{1}{2d} \sum_{J\in\mathcal{J}} r_J.$$

Dividing both sides by the common denominator D and multiplying by $2d$ gives

$$2d \cdot \sum_{J\in\mathcal{J}:\, p\in J} y_J^* \geq \sum_{J\in\mathcal{J}} y_J^*.$$

Since $\mathbf{y}^*$ is a feasible solution of the dual linear program, the left-hand side is at most $2d$, and so $\sum_{J\in\mathcal{J}} y_J^* \leq 2d$. This concludes the proof of Lemma 8.6.3 as well as of Theorem 8.6.1. □

Transversal number and matching number. Let us now look at some of the concepts appearing in the proof of Theorem 8.6.1 in a general context. Let V be an arbitrary finite set, and let $\mathcal{F}$ be a system of subsets of V.

A set $X \subseteq V$ is called a *transversal* of $\mathcal{F}$ if $F \cap X \neq \emptyset$ for every $F \in \mathcal{F}$. The **transversal number** $\tau(\mathcal{F})$ is the minimum possible number of elements of a transversal of $\mathcal{F}$.

Determining or estimating the transversal number of a given set system is an important basic problem in combinatorics and combinatorial optimization, including many other problems as special cases. For example, if we consider a graph $G = (V, E)$, and view the edges as two-element subsets of V, then a transversal of E is exactly what was called a vertex cover in Section 3.3.

Another useful notion is the **matching number** of $\mathcal{F}$, denoted by $\nu(\mathcal{F})$ and defined as the maximum number of sets in a subsystem $\mathcal{M} \subseteq \mathcal{F}$ such that no two distinct sets $F_1, F_2 \in \mathcal{M}$ intersect (such an $\mathcal{M}$ is called a *matching*).

It is easily seen that $\nu(\mathcal{F}) \leq \tau(\mathcal{F})$ for every $\mathcal{F}$: If $\mathcal{F}$ has matching number k, then $\mathcal{F}$ contains k pairwise disjoint sets; thus, we need at least k points to get a transversal of $\mathcal{F}$.

For the graph example, $\nu(E)$ is exactly the number of edges in a maximum matching. This is also where the name "matching number" comes from.

The condition in Theorem 8.6.1 that every two d-intervals in $\mathcal{J}$ intersect can be rephrased as $\nu(\mathcal{J}) = 1$. More generally, $\nu(\mathcal{F}) \leq k$ means that among every $k + 1$ members of $\mathcal{F}$ we can find two that intersect. There is a more general version of Theorem 8.6.1 stating

that $\tau(\mathcal{J}) \le 2d^2\nu(\mathcal{J})$ for every finite family of d-intervals. The proof is very similar to the one shown for Theorem 8.6.1, except that the analogue of Lemma 8.6.2 needs the well-known Turán's theorem from graph theory.

Fractional transversals and matchings. In the proof of Theorem 8.6.1 we have implicitly used another parameter of a set system, which always lies between $\nu(\mathcal{F})$ and $\tau(\mathcal{F})$ and which, unlike $\tau(\mathcal{F})$ and $\nu(\mathcal{F})$, is efficiently computable. This new parameter can be introduced in two seemingly different ways, which turn out to be equivalent by the duality theorem of linear programming.

A **fractional transversal** of $\mathcal{F}$ is any feasible solution $\mathbf{x}$ to the linear program

$$\begin{array}{ll} \text{minimize} & \sum_{v\in V} x_v \\ \text{subject to} & \sum_{v\in F} x_v \ge 1 \text{ for every } F \in \mathcal{F}, \\ & \mathbf{x} \ge \mathbf{0}. \end{array}$$

So in a fractional transversal one can take, say, one-third of one point and two-thirds of another, but for each set, the fractions for points in that set must add up to at least 1, one full point. The **fractional transversal number** $\tau^*(\mathcal{F})$ is the optimal value of the objective function, i.e., the minimum possible total weight of a fractional transversal.

Every transversal T corresponds to a fractional transversal, given by $x_v = 1$ if $v \in T$ and $x_v = 0$ otherwise, and thus $\tau^*(\mathcal{F}) \le \tau(\mathcal{F})$ for every $\mathcal{F}$.

A **fractional matching** for $\mathcal{F}$ is any feasible solution $\mathbf{y}$ to the linear program

$$\begin{array}{ll} \text{maximize} & \sum y_F \\ \text{subject to} & \sum_{F:\, v\in F} y_F \le 1 \text{ for every } v \in V, \\ & \mathbf{y} \ge \mathbf{0}, \end{array}$$

and the optimal value of the objective function is the *fractional matching number* $\nu^*(\mathcal{F})$.

Every matching $\mathcal{M}$ yields a fractional matching (we put $y_F = 1$ for $F \in \mathcal{M}$ and $y_F = 0$ otherwise). Thus, $\nu(\mathcal{F}) \le \nu^*(\mathcal{F})$.

Since the linear programs for τ^* and for ν^* are dual to each other, we always have $\nu^*(\mathcal{F}) = \tau^*(\mathcal{F})$, and altogether we have the chain of inequalities

$$\nu(\mathcal{F}) \le \nu^*(\mathcal{F}) = \tau^*(\mathcal{F}) \le \tau(\mathcal{F}).$$

We remark that if $\mathcal{F}$ is the set of edges of a bipartite graph, then König's theorem (Theorem 8.2.2) asserts exactly that $\nu(\mathcal{F}) = \tau(\mathcal{F})$. On the other hand, if $\mathcal{F}$ is the edge set of a triangle (that is, $\mathcal{F} = \{\{1,2\},\{1,3\},\{2,3\}\}$), then $\nu(\mathcal{F}) = 1 < \nu^*(\mathcal{F}) = \frac{3}{2} < \tau(\mathcal{F}) = 2$.

The proof of Theorem 8.6.1 can now be viewed as follows: First one proves that $\nu^*(\mathcal{J}) \le 2d$ for every family of d-intervals with $\nu(\mathcal{J}) = 1$,

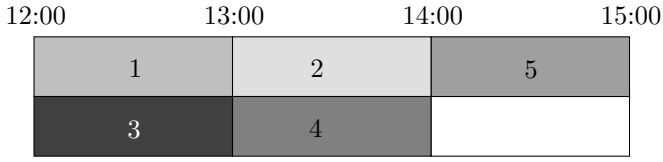
and then one shows that $\tau(\mathcal{J}) \leq d \cdot \tau^*(\mathcal{J})$. This proof scheme turned out to be very powerful and it works in many other cases as well.

Fractional concepts. Besides the fractional matching and transversal numbers, many other "fractional" quantities appear in combinatorics and combinatorial optimization. The general recipe is to take some useful integer-valued parameter Q of a graph, say, reformulate its definition as an integer program, and introduce a "fractional Q" as the optimum value of a suitable LP relaxation of the integer program. In many cases such a fractional Q is useful for studying or approximating the original Q. The book

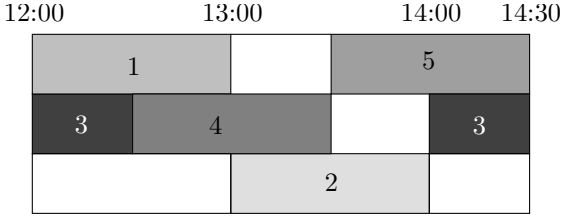
> E. R. Scheinerman and D. H. Ullman: *Fractional Graph Theory*, John Wiley & Sons, New York 1997,

nicely presents such developments. Let us conclude this section by quoting an example from that book. We consider five committees, numbered 1,2,...,5, such that 1 and 2 have a common member, and so have 2 and 3, 3 and 4, 4 and 5, and 5 and 1, while any other pair of committees is disjoint. A one-hour meeting should be scheduled for each committee, and meetings of committees with a common member must not overlap. What is the length of the shortest time interval in which all the five meetings can be scheduled?

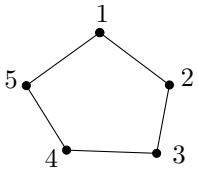
It seems that a 3-hour schedule like the one below should be optimal:



However, if one of the committees is willing to break its meeting into two half-hour parts, then a shorter schedule is possible:



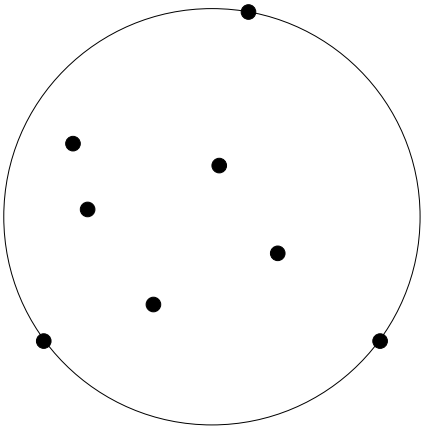
The first schedule corresponds to a proper coloring of the conflict graph



by three colors, while the second schedule corresponds to a *fractional coloring* of the same graph, with value 2.5.

8.7 Smallest Balls and Convex Programming

The smallest ball problem. We are given points $\mathbf{p}_1, \ldots, \mathbf{p}_n \in \mathbb{R}^d$, and we want to find a ball of the smallest radius that contains all the points.[8]



This looks similar to some of the geometric optimization problems that we have addressed in Chapter 2, such as the problem of placing a largest possible disk inside a convex polygon. For the smallest ball problem, however, there is no simple trick that lets us write the problem as a linear program.

We will see that it can be formulated as a *convex quadratic program*, which is in many respects the next best thing to a linear program. There are efficient solvers for convex quadratic programs, based on interior point methods or on simplex-type methods, and so this formulation can be used for computing a smallest enclosing ball in practice.

We will also derive from this formulation that the smallest enclosing ball always exists, and it is determined uniquely. (This is intuitively very plausible; think of a shrinking rubber ball.) In the course of the proof, we will establish

[8] In the plane this is sometimes referred to as the *smallest bomb problem*, but we prefer not to elaborate this association into a real-life story.

a powerful criterion for optimality of a feasible solution of a convex program, known (in a much more general context) as the *Karush–Kuhn–Tucker conditions*. These conditions are of outstanding theoretical value, and they are the basis of efficient solution methods for many classes of optimization problems, including convex quadratic programming. The reader might still wonder, how does all of this relate to linear programming? We will use the duality theorem of linear programming to derive the Karush–Kuhn–Tucker conditions.

We begin by introducing convex programming, and we will return to the smallest enclosing ball problem later.

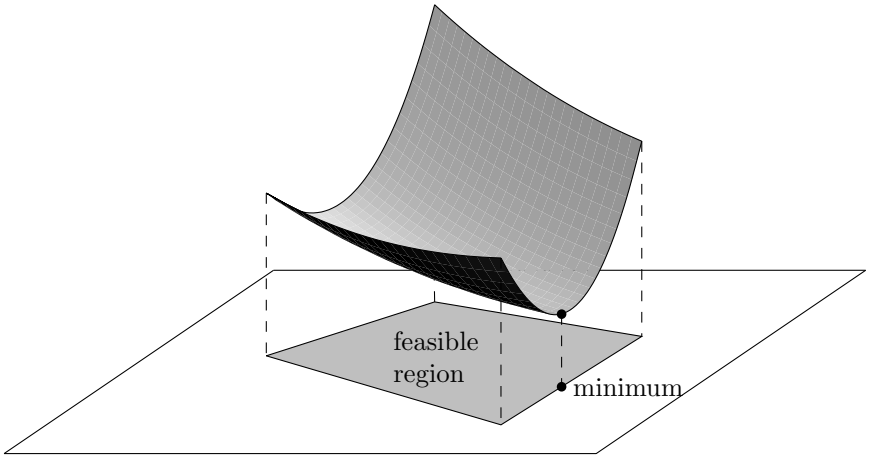
A short introduction to convex programming. Let us recall that an n-variate function $f\colon \mathbb{R}^n \to \mathbb{R}$ is *convex* if

$$f\big((1-t)\mathbf{x} + t\mathbf{y}\big) \le (1-t)f(\mathbf{x}) + tf(\mathbf{y})$$

for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and all $t \in [0,1]$. Geometrically, the segment connecting the points $(\mathbf{x}, f(\mathbf{x}))$ and $(\mathbf{y}, f(\mathbf{y}))$ in $\mathbb{R}^{n+1}$ never goes below the graph of f.

A **convex program** is the problem of minimizing a convex function in n variables subject to linear equality and inequality constraints.[9]

The following picture illustrates a 2-dimensional convex programming problem, with a planar feasible region given by four inequality constraints:



We note that the minimum need not occur at a vertex of the feasible region. Moreover, an optimal solution need not exist even if the convex function $f(\mathbf{x})$ is bounded from below; an example is the problem of minimizing e^{-x} subject to $x \ge 0$. We should also remark that, as is possible in linear programming, we cannot change minimization to maximization, since for f convex, $-f$ is typically not convex (unless f is linear). Actually, *maximizing* a convex function subject to linear constraints is a computationally difficult (NP-hard) problem.

[9] Some sources allow other types of convex constraints in a convex program, but we don't need this here.

Here we will consider convex programs in equational form:

$$\begin{array}{ll} \text{Minimize} & f(\mathbf{x}) \\ \text{subject to} & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq 0, \end{array} \tag{8.11}$$

where $A \in \mathbb{R}^{m\times n}$, $\mathbf{b} \in \mathbb{R}^m$, and $f\colon \mathbb{R}^n \to \mathbb{R}$ is a convex function.

In order to use calculus, we also assume that f is differentiable, with continuous partial derivatives. In this situation, the inequality

$$f(\mathbf{x}) \geq f(\mathbf{z}) + \nabla f(\mathbf{z})(\mathbf{x} - \mathbf{z}) \tag{8.12}$$

holds for all $\mathbf{x}, \mathbf{z} \in \mathbb{R}^n$, and this is an alternative characterization of convexity. We recall that

$$\nabla f(\mathbf{z}) = \left(\frac{\partial}{\partial x_1} f(\mathbf{x})|_{\mathbf{x}=\mathbf{z}}, \ldots, \frac{\partial}{\partial x_n} f(\mathbf{x})|_{\mathbf{x}=\mathbf{z}} \right)$$

is the gradient (vector of partial derivatives) of f at $\mathbf{z}$. Thus $\nabla f(\mathbf{z})(\mathbf{x} - \mathbf{z})$ is the scalar product of the row vector $\nabla f(\mathbf{z})$ with the column vector $\mathbf{x} - \mathbf{z}$.

Geometrically, the inequality says that the epigraph of f lies above all of its tangential hyperplanes. This has the following easy consequence.

8.7.1 Fact. *Let $C \subseteq \mathbb{R}^n$ be a convex set and $f\colon \mathbb{R}^n \to \mathbb{R}$ a differentiable convex function. A vector $\mathbf{x}^*$ minimizes $f(\mathbf{x})$ over C if and only if*

$$\nabla f(\mathbf{x}^*)(\mathbf{x} - \mathbf{x}^*) \geq 0 \quad \textit{for all } \mathbf{x} \in C.$$

Proof. First we prove that the inequality implies optimality of $\mathbf{x}^*$. Using (8.12), $\nabla f(\mathbf{x}^*)(\mathbf{x} - \mathbf{x}^*) \geq 0$ implies $f(\mathbf{x}) \geq f(\mathbf{x}^*)$, so if the former holds for all $\mathbf{x} \in C$, $\mathbf{x}^*$ is a minimizer of f over C.

For the other direction, let us assume that $\mathbf{x}^*$ is a minimizer and that $\mathbf{x} \in C$. We consider the convex combination

$$\mathbf{x}(t) := \mathbf{x}^* + t(\mathbf{x} - \mathbf{x}^*) \in C, \quad t \in [0, 1],$$

and we observe that

$$\frac{\partial}{\partial t} f(\mathbf{x}(t))|_{t=0} = \lim_{t\to 0} \frac{f(\mathbf{x}(t)) - f(\mathbf{x}^*)}{t} \geq 0$$

must hold, for otherwise, $f(\mathbf{x}(t)) < f(\mathbf{x}^*)$ for some small t. On the other hand, we have

$$\frac{\partial}{\partial t} f(\mathbf{x}(t))|_{t=0} = \nabla f(\mathbf{x}^*)(\mathbf{x} - \mathbf{x}^*),$$

by the chain rule. This completes the proof. □

Next we formulate and prove the promised optimality criterion for convex programming.

8.7.2 Proposition (Karush–Kuhn–Tucker conditions). *Let us consider the convex program*

$$\begin{array}{ll} \textit{minimize} & f(\mathbf{x}) \\ \textit{subject to} & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{array}$$

with f convex and differentiable, with continuous partial derivatives. A feasible solution $\mathbf{x}^* \in \mathbb{R}^n$ is optimal if and only if there is a vector $\tilde{\mathbf{y}} \in \mathbb{R}^m$ such that for all $j \in \{1, \ldots, n\}$,

$$\nabla f(\mathbf{x}^*)_j + \tilde{\mathbf{y}}^T \mathbf{a}_j \begin{cases} = 0 & \text{if } x_j^* > 0 \\ \geq 0 & \text{otherwise.} \end{cases}$$

Here $\mathbf{a}_j$ is the jth column of A.

The components of $\tilde{\mathbf{y}}$ are called the **Karush–Kuhn–Tucker multipliers**.

Proof. First we assume that there is a vector $\tilde{\mathbf{y}}$ with the above properties, and we let $\mathbf{x}$ be any feasible solution to the convex program. Then we get

$$\begin{array}{rcl} \Big(\nabla f(\mathbf{x}^*) + \tilde{\mathbf{y}}^T A\Big)\mathbf{x}^* & = & 0, \\ \Big(\nabla f(\mathbf{x}^*) + \tilde{\mathbf{y}}^T A\Big)\mathbf{x} & \geq & 0. \end{array}$$

Subtracting the first equation from the second, the contributions of $\tilde{\mathbf{y}}^T A$ cancel (since $A\mathbf{x} = \mathbf{b} = A\mathbf{x}^*$ by the feasibility of $\mathbf{x}$ and $\mathbf{x}^*$), and we conclude that

$$\nabla f(\mathbf{x}^*)(\mathbf{x} - \mathbf{x}^*) \geq 0.$$

Since this holds for all feasible solutions $\mathbf{x}$, the solution $\mathbf{x}^*$ is optimal by Fact 8.7.1.

Conversely, let $\mathbf{x}^*$ be optimal, and let us set $\mathbf{c}^T = -\nabla f(\mathbf{x}^*)$. By Fact 8.7.1 we have $\mathbf{c}^T(\mathbf{x} - \mathbf{x}^*) \leq 0$ for all feasible solutions $\mathbf{x}$, meaning that $\mathbf{x}^*$ is an optimal solution of the linear program

$$\begin{array}{ll} \text{maximize} & \mathbf{c}^T\mathbf{x} \\ \text{subject to} & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}. \end{array}$$

According to the dualization recipe (Section 6.2), its dual is the linear program

$$\begin{array}{ll} \text{minimize} & \mathbf{b}^T\mathbf{y} \\ \text{subject to} & A^T\mathbf{y} \geq \mathbf{c}, \end{array}$$

and the duality theorem implies that it has an optimal solution $\tilde{\mathbf{y}}$ satisfying

$$\mathbf{b}^T\tilde{\mathbf{y}} = \mathbf{c}^T\mathbf{x}^*. \tag{8.13}$$

Since $\tilde{\mathbf{y}}$ is a feasible solution of the dual linear program, we have $\tilde{\mathbf{y}}^T\mathbf{a}_j \geq c_j$ for all j, and (8.13) implies

$$(\tilde{\mathbf{y}}^T A - \mathbf{c}^T)\mathbf{x}^* = \mathbf{b}^T\tilde{\mathbf{y}} - \mathbf{c}^T\mathbf{x}^* = 0.$$

So we have $\nabla f(\mathbf{x}^*)_j + \tilde{\mathbf{y}}^T\mathbf{a}_j = -c_j + \tilde{\mathbf{y}}^T\mathbf{a}_j \geq 0$, with equality whenever $x_j^* > 0$. Therefore, we have found the desired multipliers $\tilde{\mathbf{y}}$. □

The fact that there is dualization for everyone implies Karush–Kuhn–Tucker conditions for everyone. We encourage the reader to work out the details, and we mention only one special case here: A feasible solution $\mathbf{x}^*$ of the convex programming problem

$$\begin{array}{ll} \text{minimize} & f(\mathbf{x}) \\ \text{subject to} & A\mathbf{x} = \mathbf{b} \end{array}$$

is optimal if and only if there exists a vector $\tilde{\mathbf{y}}$ such that

$$\nabla f(\mathbf{x}^*) + \tilde{\mathbf{y}}^T A = \mathbf{0}^T.$$

In this special case, the components of $\tilde{\mathbf{y}}$ are called **Lagrange multipliers** and can be obtained from $\mathbf{x}^*$ through Gaussian elimination (also see Section 7.2, where the method of Lagrange multipliers is briefly described in a more general setting). If, in addition, f is a quadratic function, its gradient is linear, so the minimization problem itself (finding an optimal $\mathbf{x}^*$ with a matching $\mathbf{y}$) can be solved through Gaussian elimination. For example, the problem of fitting a line by the method of *least squares*, mentioned in Section 2.4, is of this easy type, because its bivariate quadratic objective function (2.1) is convex.

Smallest enclosing ball as a convex program. In order to show that the smallest enclosing ball of a point set can be extracted from the solution of a suitable convex quadratic program, we use the Karush–Kuhn–Tucker conditions and the following geometric fact, which is interesting by itself.

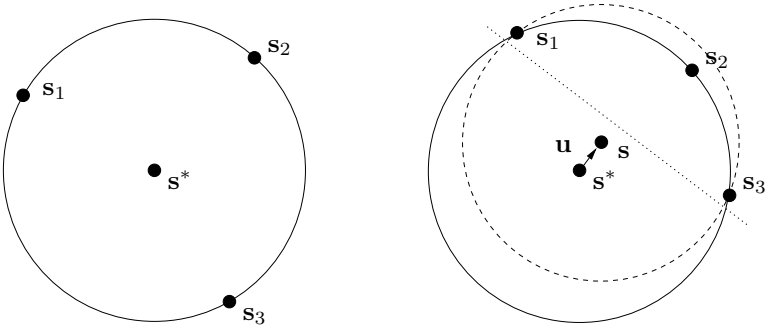
8.7.3 Lemma. *Let $S = \{\mathbf{s}_1, \ldots, \mathbf{s}_k\} \subseteq \mathbb{R}^d$ be a set of points on the boundary of a ball B with center $\mathbf{s}^* \in \mathbb{R}^d$. Then the following two statements are equivalent.*

(i) *B is the unique smallest enclosing ball of S.*
(ii) *For every $\mathbf{u} \in \mathbb{R}^d$, there is an index $j \in \{1, 2, \ldots, k\}$ such that*

$$\mathbf{u}^T(\mathbf{s}_j - \mathbf{s}^*) \leq 0.$$

It is a simple exercise to show that (ii) can be reexpressed as follows: There is no hyperplane that strictly separates S from $\mathbf{s}^*$. From the Farkas lemma (Section 6.4), one can in turn derive the following equivalent formulation: The point $\mathbf{s}^*$ is in the convex hull of the

points S. We thus have a simple geometric condition that characterizes smallest enclosing balls in terms of their boundary points. From the geometric intuition in the plane, this is quite plausible: If $\mathbf{s}^*$ is in the convex hull of S, then $\mathbf{s}^*$ cannot be moved without making the distance to at least one point larger. But if $\mathbf{s}^*$ is separated from S by a hyperplane, then moving $\mathbf{s}^*$ toward this hyperplane results in an enclosing ball of smaller radius. The direction $\mathbf{u}$ of movement satisfies $\mathbf{u}^T(\mathbf{s}_j - \mathbf{s}^*) > 0$ for all j.



Proof. We start by analyzing the distance between a point $\mathbf{s}_j \in S$ and a potential ball center $\mathbf{s} \neq \mathbf{s}^*$. Let r be the radius of the ball B. Given $\mathbf{s} \neq \mathbf{s}^*$, we can uniquely write it in the form

$$\mathbf{s} = \mathbf{s}^* + t\mathbf{u},$$

where $\mathbf{u}$ is a vector of length 1 and $t > 0$. For $j = 1, 2, \ldots, k$ we get

$$\begin{aligned}(\mathbf{s}_j - \mathbf{s})^T(\mathbf{s}_j - \mathbf{s}) &= (\mathbf{s}_j - \mathbf{s}^* - t\mathbf{u})^T(\mathbf{s}_j - \mathbf{s}^* - t\mathbf{u}) \\ &= (\mathbf{s}_j - \mathbf{s}^*)^T(\mathbf{s}_j - \mathbf{s}^*) + t^2\mathbf{u}^T\mathbf{u} - 2t\mathbf{u}^T(\mathbf{s}_j - \mathbf{s}^*) \\ &= r^2 + t^2 - 2t\mathbf{u}^T(\mathbf{s}_j - \mathbf{s}^*).\end{aligned}$$

Given $\alpha \in \mathbb{R}$ and sufficiently small $t > 0$, we have $t^2 - 2t\alpha > 0$ if and only if $\alpha \leq 0$. This shows that (for sufficiently small t)

$$(\mathbf{s}_j - \mathbf{s})^T(\mathbf{s}_j - \mathbf{s}) > r^2 \quad \Leftrightarrow \quad \mathbf{u}^T(\mathbf{s}_j - \mathbf{s}^*) \leq 0, \tag{8.14}$$

where the implication "$\Leftarrow$" holds for all $t > 0$.

This equivalence implies the two directions of the lemma. For (i)$\Rightarrow$(ii), we argue as follows: Since $\mathbf{s}^*$ is the unique point with distance at most r from all points in S, we know that for every $\mathbf{u}$ with $\|\mathbf{u}\| = 1$ and for *all* $t > 0$, the point $\mathbf{s} = \mathbf{s}^* + t\mathbf{u}$ has distance more than r to one of the points in S. By the implication "$\Rightarrow$" of (8.14), there is some j with $\mathbf{u}^T(\mathbf{s}_j - \mathbf{s}^*) \leq 0$. To show (ii)$\Rightarrow$(i), let us consider any point $\mathbf{s}$ of the form $\mathbf{s}^* + t\mathbf{u} \neq \mathbf{s}^*$. Since there is an index j with $\mathbf{u}^T(\mathbf{s}_j - \mathbf{s}^*) \leq 0$, implication "$\Leftarrow$" of (8.14) shows

that $\mathbf{s}$ has distance more than r to some point in S. It follows that B is the unique smallest enclosing ball of S. □

Now we can state and prove the main result.

8.7.4 Theorem. *Let $\mathbf{p}_1, \ldots, \mathbf{p}_n$ be points in $\mathbb{R}^d$, and let Q be the $d \times n$ matrix whose jth column is formed by the d coordinates of the point $\mathbf{p}_j$. Let us consider the optimization problem*

$$\begin{array}{ll} \textit{minimize} & \mathbf{x}^T Q^T Q \mathbf{x} - \sum_{j=1}^n x_j \mathbf{p}_j^T \mathbf{p}_j \\ \textit{subject to} & \sum_{j=1}^n x_j = 1 \\ & \mathbf{x} \geq \mathbf{0} \end{array} \tag{8.15}$$

in the variables $x_1, \ldots, x_n$. Then the objective function $f(\mathbf{x}) := \mathbf{x}^T Q^T Q \mathbf{x} - \sum_{j=1}^n x_j \mathbf{p}_j^T \mathbf{p}_j$ is convex, and the following statements hold.

(i) *Problem (8.15) has an optimal solution $\mathbf{x}^*$.*

(ii) *There exists a point $\mathbf{p}^*$ such that $\mathbf{p}^* = Q\mathbf{x}^*$ holds for* every *optimal solution $\mathbf{x}^*$. Moreover, the ball with center $\mathbf{p}^*$ and squared radius $-f(\mathbf{x}^*)$ is the unique ball of smallest radius containing P.*

Proof. The matrix $Q^T Q$ is positive semidefinite, and from this the convexity of f is easy to verify (we leave it as an exercise).

The feasible region of program (8.15) is a compact set (actually, a *simplex*), and we are minimizing a continuous function over it. Consequently, there exists an optimal solution $\mathbf{x}^*$. In order to apply the Karush–Kuhn–Tucker conditions, we need the gradient of the objective function:

$$\nabla f(\mathbf{x}) = 2\mathbf{x}^T Q^T Q - (\mathbf{p}_1^T \mathbf{p}_1, \mathbf{p}_2^T \mathbf{p}_2, \ldots, \mathbf{p}_n^T \mathbf{p}_n).$$

The program has only one equality constraint. With $\mathbf{p}^* = Q\mathbf{x}^* = \sum_{j=1}^n x_j^* \mathbf{p}_j$, Proposition 8.7.2 tells us that we find a 1-dimensional vector $\tilde{\mathbf{y}} = (\mu)$ such that

$$2\mathbf{p}_j^T \mathbf{p}^* - \mathbf{p}_j^T \mathbf{p}_j + \mu \begin{cases} = 0 & \text{if } x_j^* > 0 \\ \geq 0 & \text{otherwise.} \end{cases} \tag{8.16}$$

Subtracting ${\mathbf{p}^*}^T \mathbf{p}^*$ from both sides and multiplying by -1 yields

$$\|\mathbf{p}_j - \mathbf{p}^*\|^2 \begin{cases} = \mu + {\mathbf{p}^*}^T \mathbf{p}^* & \text{if } x_j^* > 0 \\ \leq \mu + {\mathbf{p}^*}^T \mathbf{p}^* & \text{otherwise.} \end{cases}$$

This means that $\mathbf{p}^*$ is the center of a ball of radius $r = \sqrt{\mu + {\mathbf{p}^*}^T \mathbf{p}^*}$ that contains all points from P and has the points $\mathbf{p}_j$ with $x_j^* > 0$ on the boundary. From (8.16) and $\mathbf{x} \geq \mathbf{0}$ we also get

$$\mu = \sum_{j=1}^n x_j^* \mu = \sum_{j=1}^n x_j^* \mathbf{p}_j^T \mathbf{p}_j - 2\sum_{j=1}^n x_j^* \mathbf{p}_j^T \mathbf{p}^* = \sum_{j=1}^n x_j^* \mathbf{p}_j^T \mathbf{p}_j - 2{\mathbf{p}^*}^T \mathbf{p}^*,$$

and $r^2 = \sum_{j=1}^{n} x_j^* \mathbf{p}_j^T \mathbf{p}_j - \mathbf{p}^{*T} \mathbf{p}^* = -f(\mathbf{x}^*)$ follows.

It remains to prove that there can be no other ball of radius at most r that contains all points from P (this also shows that $\mathbf{p}^*$ does not depend on the choice of $\mathbf{x}^*$).

We define $F = \{j \in \{1, 2, \ldots, n\} : x_j^* > 0\}$ and apply Lemma 8.7.3 with $\mathbf{s}^* = \mathbf{p}^*$ and

$$S = \{\mathbf{p}_j : j \in F\}.$$

We already know that these points are on the boundary of a ball B of radius r around $\mathbf{p}^* = \sum_{j \in F} x_j^* \mathbf{p}_j$. Using $\sum_{j \in F} x_j^* = 1$, we get that the following holds for all vectors $\mathbf{u}$:

$$\sum_{j \in F} x_j^* \mathbf{u}^T (\mathbf{p}_j - \mathbf{p}^*) = \mathbf{u}^T \left(\sum_{j \in F} x_j^* \mathbf{p}_j - \sum_{j \in F} x_j^* \mathbf{p}^* \right) = \mathbf{u}^T (\mathbf{p}^* - \mathbf{p}^*) = 0.$$

It follows that there must be some $j \in F$ with $\mathbf{u}^T (\mathbf{p}_j - \mathbf{p}^*) \leq 0$. By Lemma 8.7.3, B is the unique smallest enclosing ball of $S \subseteq P$, and this implies that B is the unique smallest enclosing ball of P as well. □

A recent book on the topics of this section is

S. Boyd and L. Vandenberghe: *Convex Optimization*, Cambridge University Press, Cambridge 2004.