it is invertible. (Why?) Further, the reduced cost of the basic variable s_{m+1} is zero. (Why?) Therefore, we fulfil the condition for primal optimality. In order to check if the additional constraint changes the optimal solution, it remains therefore to check if $s_{m+1} \geq 0$ (that is, that the primal feasibility condition is fulfilled), which is simply done by inserting the current BFS in the new constraint. If the new constraint is fulfilled, then the BFS is both optimal and feasible and we are done; otherwise, we have identified a violated primal constraint in the current BFS, and may proceed to find the optimal solution to the new problem by using, for example, the dual simplex method as explained in Section 10.4.

10.6 Column generation in linear programming

In Section 1.1 we discussed the modeling of optimization problems. Specifically, Remark 1.3 contains a discussion on a potential difficulty in explicitly representing all the variables in a truly large-scale problem, and the term "column generation" was introduced. The purpose of this section is to explain, in the context of linear programming, what column generation is and how it can be applied to a linear problem to yield a large-scale linear programming algorithm. Our discussion will be centered around a concrete LP problem, to be introduced in the following.

10.6.1 The minimum cost multicommodity network flow problem

A network is a finite set $\mathcal{N}$ of nodes $i = 1, \dots, m$ and a finite set $\mathcal{L}$ of ordered pairs of nodes, $\ell = (i, j)$, called links. Through the links flow can be sent; link $\ell \in \mathcal{L}$ is assumed to have a finite (positive) capacity u_ℓ of flow. The goal is to send a particular amount of flows through the network at minimum cost, where the flows are of different types (called commodities), originating at and terminating in different nodes in the network. Suppose we denote by the superscript $k \in \mathcal{K}$ the different commodities. Each commodity k is assumed to have a single origin and terminal node, $o^k \in \mathcal{N}$ and $t^k \in \mathcal{N}$, respectively, and the demand for transportation in commodity k is d^k units of flow. Suppose further that associated with the flow of commodity k along link ℓ is a unit transportation cost of $c_\ell^k > 0$.

We denote by x_ℓ^k the flow along link ℓ associated with a given commodity k. In order to describe a feasible flow we must represent the constraints for meeting supplies and demands, keeping flow conservation at transshipment nodes, and fulfilling the capacity constraints. To this end, we introduce a first model, based on the node–link representation.

Let

$$\mathcal{F}_i := \{\, j \in \mathcal{N} \mid (i,j) \in \mathcal{L} \,\} \quad \text{and} \quad \mathcal{B}_i := \{\, j \in \mathcal{N} \mid (j,i) \in \mathcal{L} \,\}$$

denote the set of links initiated and terminating at node $i \in \mathcal{N}$ (the forward and backward star, respectively). Then, the capacity and flow conservation constraints can be written elementwise as

$$\sum_{k\in\mathcal{K}} x_{ij}^k \leq u_{ij}, \qquad (i,j) \in \mathcal{L}, \tag{10.17a}$$

$$\sum_{j\in\mathcal{B}_i} x_{ji}^k - \sum_{j\in\mathcal{F}_i} x_{ij}^k = d^k, \qquad i \in \mathcal{N}, \quad k \in \mathcal{K}, \tag{10.17b}$$

$$x_{ij}^k \geq 0, \qquad (i,j) \in \mathcal{L}, \quad k \in \mathcal{K}. \tag{10.17c}$$

Hence, the minimum cost multicommodity network flow problem is that to

$$\begin{aligned}
&\underset{\boldsymbol{x}}{\text{minimize}} \quad z = \sum_{k\in\mathcal{K}} \sum_{\ell\in\mathcal{L}} c_\ell^k x_\ell^k, && \text{(10.18a)}\\
&\text{subject to} \quad \sum_{k\in\mathcal{K}} x_{ij}^k \leq u_{ij}, \qquad (i,j) \in \mathcal{L}, && \text{(10.18b)}\\
&\sum_{j\in\mathcal{B}_i} x_{ji}^k - \sum_{j\in\mathcal{F}_i} x_{ij}^k = d^k, \qquad i \in \mathcal{N}, \quad k \in \mathcal{K}, && \text{(10.18c)}\\
&x_{ij}^k \geq 0, \qquad (i,j) \in \mathcal{L}, \quad k \in \mathcal{K}. && \text{(10.18d)}
\end{aligned}$$

We develop however the column generation principle from a reformulation of the problem wherein we utilize the Representation Theorem 8.9. Therefore, let $r \in \mathcal{R}^k$ denote a route from node o^k to node t^k in the network; we assume that all such routes utilize the links in their right direction, that is, in the sequence of links comprising the route the node orderings are kept. Let further h_r^k denote the flow on route $r \in \mathcal{R}^k$, and c_r^k denote its unit cost.

In order to relate flows and costs on routes and links we introduce a link–route incidence matrix:

$$\gamma_{\ell r}^k := \begin{cases} 1, & \text{if } \ell \in \mathcal{R}^k, \\ 0, & \text{otherwise,} \end{cases} \qquad \ell \in \mathcal{L}, \quad r \in \mathcal{R}^k, \quad k \in \mathcal{K}.$$

Hence, we have that

$$c_r^k = \sum_{\ell\in\mathcal{L}} \gamma_{\ell r}^k c_\ell^k. \tag{10.19}$$

The optimization problem stemming from the link–route formulation then is that to

$$\begin{aligned}
\underset{\boldsymbol{h}}{\text{minimize}} \quad & z = \sum_{k\in\mathcal{K}} \sum_{r\in\mathcal{R}^k} c_r^k h_r^k, && && (10.20\text{a})\\
\text{subject to} \quad & \sum_{k\in\mathcal{K}} \sum_{r\in\mathcal{R}^k} \gamma_{\ell r}^k h_r^k \leq u_\ell, && \ell\in\mathcal{L}, && (10.20\text{b})\\
& \sum_{r\in\mathcal{R}^k} h_r^k = d^k, && k\in\mathcal{K}, && (10.20\text{c})\\
& h_r^k \geq 0, && r\in\mathcal{R}^k, \quad k\in\mathcal{K}. && (10.20\text{d})
\end{aligned}$$

In order to utilize our development of the constraints of the problem at hand in a discussion on the Representation Theorem 8.9 we perform one last formal manipulation, namely that to scale the variables h_r^k by the demand in the commodity: introduce the new variables

$$\lambda_r^k := \frac{h_r^k}{d^k}, \qquad r\in\mathcal{R}^k, \quad k\in\mathcal{K}.$$

The problem (10.20) then becomes that to

$$\begin{aligned}
\underset{\boldsymbol{\lambda}}{\text{minimize}} \quad & z = \sum_{k\in\mathcal{K}} \sum_{r\in\mathcal{R}^k} d^k c_r^k \lambda_r^k, && && (10.21\text{a})\\
\text{subject to} \quad & \sum_{k\in\mathcal{K}} \sum_{r\in\mathcal{R}^k} d^k \gamma_{\ell r}^k \lambda_r^k \leq u_\ell, && \ell\in\mathcal{L}, && (10.21\text{b})\\
& \sum_{r\in\mathcal{R}^k} \lambda_r^k = 1, && k\in\mathcal{K}, && (10.21\text{c})\\
& \lambda_r^k \geq 0, && r\in\mathcal{R}^k, \quad k\in\mathcal{K}. && (10.21\text{d})
\end{aligned}$$

We can now compare the formulations (10.18) and (10.21). Suppose that we let

$$X^k := \{\, \boldsymbol{x}^k \in \mathbb{R}^{|\mathcal{L}|} \mid (10.18\text{c})\text{–}(10.18\text{d}) \text{ is fulfilled} \,\}.$$

Then, what we have done in the above transformation is to represent a flow in X^k as $x_l^k = d^k \sum_{r\in\mathcal{R}^k} \gamma_{\ell r}^k \lambda_r^k$, $l\in\mathcal{L}$, for some vector

$$\boldsymbol{\lambda}^k \in \Lambda^k := \left\{ \boldsymbol{\lambda}^k \in \mathbb{R}_+^{|\mathcal{R}^k|} \;\middle|\; \sum_{r\in\mathcal{R}^k} \lambda_r^k = 1 \right\}.$$

According to the Representation Theorem 8.9 the polyhedron X^k is the sum of a polytope (the convex hull of its extreme points) and a polyhedral cone (the cone of its extreme directions). The set Λ^k is a polytope, described by the convex hull of flows along single routes. Let us lastly introduce a further notion from graph theory. A *cycle* is a finite ordering of nodes such that the last node equals the first and the node ordering throughout follows the right orientation of the links formed by the consequtive ordered node pairs. As flows along cycles do not change the node balance and do no not reduce the flow in any link, such flows correspond to feasible directions of the polyhedra X^k, and a subset of simple such flows constitute the extremal vectors of the polyhedron X^k. Since such flows are not represented in the problem statement (10.21) this problem is not equivalent to that in (10.18), precisely because of the lack in the former model of any flows along cycles in the network.

While the above two formulations are not equivalent in terms of the size of their respective feasible sets, it is true that under the given assumption that the link costs are positive they are computationally equivalent, in the sense that their optimal solutions are the same and no computations of flows along cycles need ever be made. The reason is that flows along cycles only add to the total cost, so an optimal solution cannot contain a cyclic flow.

10.6.2 The column generation principle

Suppose that the link flow capacity constraints (10.18b) [or, (10.21b)] were not present. As can easily be seen from what remains of the problem formulation (10.21) the resulting problem is that to find, for each individual commodity $k \in \mathcal{K}$, a cheapest route from o^k to t^k, shipping the full demand d^k along it. The problem (10.21) can therefore be seen as that to find a number of routes in each commodity, such that a selection of proportions of the demands to send along them results in a total link flow that obeys the link flow capacity restrictions and that at the same time provides the smallest total transportation cost.

"Column generation" refers to a principle for generating these routes algorithmically rather than a priori. The word "column" refers to the fact that each variable in a linear program corresponds to a given column in the constraint matrix, so the word also signals that column generation is a principle for generating variables in the optimization problem that we believe should have non-zero values in an optimal solution. Note that in the problem discussed in Remark 1.3 it is even impossible to enumerate all the variables a priori, but the main point is that it might not be advantageous to do so even if one could. The column generation is performed through an ingeneous use of the simplex method's criterion for the selection of the incoming basic variable. The remainder of this section explains in detail how column generation is performed and relates the development both to the simplex method and to Lagrangian relaxation.

To provide a framework for our discussion we develop an LP dual of the link–route formulation (10.21). Let α_ℓ, $\ell \in \mathcal{L}$, and β^k, $k \in \mathcal{K}$, denote the dual variables associated with the linear constraints (10.21b) and (10.21c), respectively. Then, we can write the LP dual of (10.21) as the problem to

$$\underset{(\boldsymbol{\alpha},\boldsymbol{\beta})}{\text{maximize}}\ z = \sum_{k\in\mathcal{K}} \beta^k - \sum_{\ell\in\mathcal{L}} u_\ell \alpha_\ell, \tag{10.22a}$$

$$\text{subject to} \quad \beta^k - \sum_{\ell\in\mathcal{L}} d^k y^k_{\ell r} \alpha_\ell \leq d^k c^k_r, \quad r \in \mathcal{R}^k, \quad k \in \mathcal{K}, \tag{10.22b}$$

$$\alpha_\ell \geq 0, \qquad \ell \in \mathcal{L}. \tag{10.22c}$$

As explained in Section 10.3 we attain an optimal primal BFS with the simplex method precisely when the complementary dual basis is feasible. Further, the incoming variable criterion of the simplex method is to increase from zero a primal variable currently not in the basis (and therefore at level zero) which has the most negative reduced cost, which is equivalent to the primal variable corresponding to the most violated dual constraint. (See the discussion following Theorem 10.15.)

Suppose then that we have access to a primal BFS in (10.21) and a corresponding infeasible dual basis in (10.22), and wish to make progress by using the simplex method. From the appearance of the dual problem (10.22) we see that for each $k \in \mathcal{K}$ we should find either a route $r \in \mathcal{R}^k$ that violates the dual constraint (10.22b), or choose a variable α_ℓ that violates a dual constraint (10.22c). (The former are the dual constraints corresponding to the primal variables λ^k_r, while the latter are the dual constraints corresponding to the primal slack variables that one must introduce into the link flow capacity constraints before using the simplex method.) Following the usual standard of how to operate the simplex method we should moreover choose an incoming variable with the least reduced cost, which of course here corresponds to the most violated constraint in the dual problem (10.22). While it appears to be an expensive operation to find the most violated constraint among those in (10.22b) since the number of routes is so large, it actually corresponds to solving a specially structured LP problem, as we shall see. Clearly, the appearance of the common constants β^k and d^k does not influence this choice; further, note that the route cost c^k_r is given through (10.19). Hence, finding

$$\underset{r\in\mathcal{R}^k}{\text{minimum}}\ d^k c^k_r + \sum_{\ell\in\mathcal{L}} d^k y^k_{\ell r} \alpha_l$$

is the same as the problem of finding, for each $k \in \mathcal{K}$,

$$\underset{\boldsymbol{x} \in X^k}{\text{minimum}} \sum_{\ell \in \mathcal{L}} (c_\ell^k + \alpha_\ell) x_\ell^k. \tag{10.23}$$

This is nothing but the problem of finding a shortest route from node o^k to node t^k when the link costs are given by $c_\ell^k + \alpha_\ell$, $\ell \in \mathcal{L}$. (Notice that since each $c_\ell^k > 0$ and $\alpha_\ell \geq 0$, no negative cycles can appear.) The appearance of the additional link cost α_ℓ stemming from the capacity constraint, illustrates that "good" routes for solving the overall problem are not necessarily those that are the cheapest in terms of the original costs.

In our context then, the incoming criterion of the simplex method corresponds to a special linear program. We may in fact derive the same problem equivalently by using arguments from Lagrangian duality, which we do next.

With the same arguments as before, namely that the capacity constraints (10.18b) are complicating, we introduce Lagrange multipliers α_ℓ, $\ell \in \mathcal{L}$, and form the Lagrangian

$$\begin{aligned} L(\boldsymbol{x}, \boldsymbol{\alpha}) &:= \sum_{k \in \mathcal{K}} \sum_{\ell \in \mathcal{L}} c_\ell^k x_\ell^k + \sum_{\ell \in \mathcal{L}} \alpha_\ell \left(\sum_{k \in \mathcal{K}} x_\ell^k - u_\ell \right) \\ &= -\sum_{\ell \in \mathcal{L}} \alpha_\ell u_\ell + \sum_{k \in \mathcal{K}} \sum_{\ell \in \mathcal{L}} (c_\ell^k + \alpha_\ell) x_\ell^k, \end{aligned}$$

to be minimized over the remaining constraints, that is, $\prod_{k \in \mathcal{K}} X^k$. We already knew from the development in Section 6.2.4 that Lagrangian duality is equivalent to LP duality in the context of linear programming, and we confirm that here, as the Lagrangian minimization problem is precisely (10.23).

As we know from the theory of strong duality developed above, if $\boldsymbol{\alpha} = \boldsymbol{\alpha}^*$ is optimally chosen then among the minimizers of $L(\cdot, \boldsymbol{\alpha}^*)$ over $\prod_{k \in \mathcal{K}} X^k$ we identify an optimal solution to (10.18). Therefore, by choosing proper additional "prices" $\boldsymbol{\alpha}$ for using the links, the commodities are made to "cooperate" to optimally utilize the link capacities. Moreover, these prices are the optimal dual variable values for the capacity constraints. It is therefore not surprising that the decomposition principle sometimes is referred to as "price-directive decomposition."

10.6.3 An algorithm instance

Constructing an algorithm from the above development is straightforward, as is the subject of convergence.

Given a BFS and the corresponding vector of dual multipliers, solve (10.23) for each $k \in \mathcal{K}$. If, at the solutions to these problems, (10.22b) is fulfilled and $\alpha_\ell \geq 0$ for all $\ell \in \mathcal{L}$, then we are done, since it means that the dual basis is dual feasible; the current BFS then describes an optimal multicommodity flow.

If, for some for some $k \in \mathcal{K}$, (10.22b) is not fulfilled, then let $\bar{r} \in \mathcal{R}^k$ be a route solving (10.23) for each $k \in \mathcal{K}$. In the problem (10.21) the column to be added to the basis has the form

$$\begin{pmatrix} (d^k \gamma_{\ell\bar{r}})_{\ell \in \mathcal{L}} \\ 1 \end{pmatrix}_{k \in \mathcal{K}}.$$

This is a vector of length $|\mathcal{K}|(|\mathcal{L}| + 1)$. In a streamlined version of the algorithm, we would store subvectors (that is, routes) individually for each commodity in order not to duplicate the information stored; see below. Notice that this incoming variable corresponds to a vector of concatenated extreme point vectors $\boldsymbol{x}^k \in X^k$ (together with additional 1:s).

If (10.22b) holds but $\alpha_\ell < 0$ for some link $\ell \in \mathcal{L}$, then the primal slack variable in the corresponding link capacity constraint is a candidate in the selection of the incoming variable, signaling that the flow should be decreased on that link.

The pricing step of the (revised version of the) simplex method for solving the problem (10.21) has exactly this appearance. If we let the primal variable with the most negative reduced cost be the incoming variable and pivot in the usual manner, then the algorithm is nothing but the primal simplex method for the problem (10.21). One of the main ideas behind a column generation method is however to utilize the information that has been generated to a greater extent than in the standard simplex method. Perhaps the most important motivation is the fact that the column generation can be a costly operation in general; in the present problem column generation is performed through the solution of a number of shortest route problems, which can be quite costly when the network is large. We next explain how a column generation algorithm would continue after the pricing step, and finally discuss the main differences to the standard simplex method.

Following a number of iterations, in particular a number of column generation steps corresponding to the pricing step of the simplex method, we have solved a series of shortest route problems for each commodity k; we have therefore also generated a subset of these, say $\bar{\mathcal{R}}^k \subset \mathcal{R}^k$, $k \in \mathcal{K}$. Rather than simply pivoting on the current basis and perform another pricing step, we will utilize the routes in $\bar{\mathcal{R}}^k$ to solve a *restricted master problem*. The name refers to the fact that the original (or, master) problem is (10.21) which includes all the routes in each set $\mathcal{R}^k$, while the restricted problem is the restriction of this problem to the route sets $\bar{\mathcal{R}}^k \subset \mathcal{R}^k$. The idea is to use the standard simplex method on this problem, which in particular means that the pricing part reduces to a simple comparison, for each commodity k, among the routes' reduced costs. (This is of course a much cheaper operation than to solve a shortest route problem for each commodity.) At convergence, we have access to a BFS in the original problem which solves the current restriction to the original problem (10.21). Only then follows the pricing step, amounting to solving k shortest route problems. If any new routes are generated, we store this information and thereby augment the sets $\bar{\mathcal{R}}^k$, and continue. The algorithm terminates when the BFS solving the current restriction has a non-negative reduced cost, as explained earlier.

Variations of this process exist, of course. For example, one could include more than one route in a given commodity if more than one route is found to have a negative reduced cost. One could also imagine a version of the algorithm, in which one does not solve each restriction to optimality before performing a pricing in the overall problem; this algorithm then is a variation somewhere in between the standard simplex method for the problem (10.21) and the column generation method developed above.

A feature of the column generation method is that the solution to the pricing problem (10.23) provides a lower bound on the optimal value of the problem (10.21). (The problem (10.23) is indeed the result of a Lagrangian relaxation, as already explained, so the Weak Duality Theorem 6.5 applies.) This is in contrast to the ordinary pricing step of the simplex method: it is based on solving the problem to

$$\begin{aligned} &\text{minimize} && \tilde{\boldsymbol{c}}_N^{\mathrm{T}} \boldsymbol{x}_N, \\ &\text{subject to} && \boldsymbol{x}_N \in [0,1]^{n-m}. \end{aligned}$$

Check that the solution to this problem provides the set of non-basic variables with the most negative reduced costs, and that this problem is not a Lagrangian relaxation.

10.7 Notes and further reading

For an account of the early history of LP duality theory, see [LRS91].

Linear programming duality theory was introduced by John von Neumann [vNe47]. His results build upon his earlier work in game theory. The first published proof of the Strong Duality Theorem is found in Gale, Kuhn, and Tucker [GKT51]. The Complementary Slackness Theorem is due to Dantzig and Orden [DaO53].

Text books that discuss LP duality and sensitivity analysis are [Dan63, Chv83, Mur83, Sch86, DaT97, Pad99, Van01, DaT03, DaM05].

More on the modelling of, and algorithms and duality for, linear network optimization can be found in [AMO93].

The decomposition principle is often referred to as the Dantzig–Wolfe algorithm, following [DaW60, DaW61]. (In his classic text book on linear programming George Dantzig himself refers to it as "centralized planning without complete information at the center;" see [Dan63, Chapter 23].) This algorithm is in fact an application of the decomposition principle to the outer representation (10.18), whereas column generation refers to the solution of the inner representation (10.21). The column genaration principle can also be applied to integer programs, whence the master problem always is a linear program while the column generation (or, Lagrangian relaxation) problem would be an integer program. Such a type of problem is in fact more common to use when introducing column generation. The best general description of column generation, and of classic algorithms based on this principle, is still that in Lasdon [Las70, Chapters 3–4]; column generation is there illustrated for the integer programming problem known as the cutting stock problem, originally formulated by Gilmore and Gomory [GiG61, GiG63]. More modern expositions on column generation are found in [Wol98, LaP06].

10.8 Exercises

Exercise 10.1 (constructing the LP dual) Consider the linear program

$$
\begin{array}{lrl}
\text{maximize} & z = & 6x_1 - 3x_2 - 2x_3 + 5x_4, \\
\text{subject to} & & 4x_1 + 3x_2 - 8x_3 + 7x_4 = 11, \\
& & 3x_1 + 2x_2 + 7x_3 + 6x_4 \geq 23, \\
& & 7x_1 + 4x_2 + 3x_3 + 2x_4 \leq 12, \\
& & x_1, \quad x_2 \geq 0, \\
& & x_3 \leq 0, \\
& & x_4 \quad \text{free}.
\end{array}
$$

Construct its linear programming dual.

Exercise 10.2 (constructing the LP dual) Consider the linear program

$$
\begin{array}{ll}
\text{minimize} & z = \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}, \\
\text{subject to} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \\
& \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}.
\end{array}
$$

(a) Construct its linear programming dual.