Lecture 9

Distributed Control Using Price Mechanisms

For large scale production planning and other optimization problems, it is essential to have methods for decomposition into smaller subproblems. This is for several reasons. One is computational complexity. Solving several subproblems in parallel makes it possible to solve the original problem more quickly. Another reason is flexibility. If the problem data changes somewhere it can be very costly and time-consuming to redo the entire optimization of all variables. A decomposition into subproblems makes it possible to account for new data in a simpler and more effective manner. The main ideas will be explained in terms of an example:

Example 1. A company consists of two sub-divisions. One sub-division manufactures garden furniture (by sawing and assembling), the other sub-division manufactures sleds (also by sawing and assembling). Each division manufactures two different kinds of their respective products. Both sub-divisions send their products to a common painting station. The objective is to maximize company income given the resources.

Product	# of items	Income / item
Garden Furniture 1	x_1	c_1
Garden Furniture 2	x_2	c_2
Sled 1	x_3	c_3
Sled 2	x_4	c_4

Constraints for furniture division:

$$\begin{aligned} 7x_1 + 10x_2 &\leq 100 && \text{(Sawing)} \\ 16x_1 + 12x_2 &\leq 135 && \text{(Assembling)} \end{aligned}$$

Constraints for sled division:

$$\begin{aligned} 10x_3 + 9x_4 &\leq 70 && \text{(Sawing)} \\ 6x_3 + 9x_4 &\leq 60 && \text{(Assembling)} \end{aligned}$$

Painting Constraint:

$$5x_1 + 3x_2 + 3x_3 + 2x_4 \leq 45$$

Altogether, an optimal production plan for the two sub-divisions can be found by solving the linear programming problem

$$\begin{array}{ll} \text{Maximize}_{x\geq 0} & c_1x_1 + c_2x_2 + c_3x_3 + c_4x_4 \\ \text{subject to} & 7x_1 + 10x_2 \leq 100 \\ & 16x_1 + 12x_2 \leq 135 \\ & 10x_3 + 9x_4 \leq 70 \\ & 6x_3 + 9x_4 \leq 60 \\ & 5x_1 + 3x_2 + 3x_3 + 2x_4 \leq 45 \end{array} \tag{9.1}$$

The production plans for the two sub-divisions are coupled through the common painting station. To decompose the optimization problem, we will replace the constraint on painting capacity by a price to payed for painting. Hence, instead of (9.1) consider the optimization problem, where $\lambda \geq 0$ is the painting "price".

$$\begin{array}{ll} \text{Maximize}_{x\geq 0} & c^Tx + \overbrace{\lambda(45 - 5x_1 - 3x_2 - 3x_3 - 2x_4)}^{\text{cost for exceeding the painting capacity}} \\ \text{subject to} & 7x_1 + 10x_2 \leq 100 \\ & 16x_1 + 12x_2 \leq 135 \\ & 10x_3 + 9x_4 \leq 70 \\ & 6x_3 + 9x_4 \leq 60 \end{array} \tag{9.2}$$

Notice that the maximum of (9.2) must be at least as big as that of (9.1). This is because every solution to the inequalities in (9.1) gives a value in (9.2) which is at least as big as the value in (9.1). In practice, this means that if the painting price is set too low, the optimization will result in a solution with a painting demand that exceeds the supply and which is therefore not implementable. On the contrary, a painting price that is too high will lead to smaller production volumes than desirable and the painting facility gets underutilized. However, as will be seen later, there is always value $\lambda = \lambda^*$ for which the maximum of (9.2) recovers the optimum of (9.1).

An important aspect of the formulation with price variable is that there is no longer any direct coupling between the two sub-divisions, so they can each optimize their production separately:

$$\begin{array}{ll} \text{Maximize}_{x_1,x_2\geq 0} & c_1x_1 + c_2x_2 - \lambda(5x_1 + 3x_2) \\ \text{subject to} & 7x_1 + 10x_2 \leq 100 \\ & 16x_1 + 12x_2 \leq 135 \end{array} \tag{9.3}$$

$$\begin{array}{ll} \text{Maximize}_{x_3,x_4\geq 0} & c_3x_3 + c_4x_4 - \lambda(3x_3 + 2x_4) \\ \text{subject to} & 10x_3 + 9x_4 \leq 70 \\ & 6x_3 + 9x_4 \leq 60 \end{array} \tag{9.4}$$

When $\lambda = \lambda^*$, the optimal values $x_1^*, \ldots, x_4^*$ will satisfy the painting capacity constraint $5x_1^* + 3x_2^* + 3x_3^* + 2x_4^* \leq 45$. □

The example above illustrates how the introduction of prices can be used to decompose an optimization problem into smaller ones. This principle is very general. In fact, it is the basis for our entire economy. However, before continuing the discussion of distributed optimization, we will recall the main ideas of duality theory for linear programming, where price variables appear in a natural way.

Duality in Linear Programming

Consider the following general linear programming problem.

$$\begin{aligned} p^* = \max \quad & c^T x \\ \text{subject to} \quad & Ax \preceq b, \quad x \succeq 0 \end{aligned}$$

Introduce a vector of dual variables $\lambda \succeq 0$ (prices) for the constraints $Ax \preceq b$ and define the corresponding dual function $g(\lambda)$:

$$g(\lambda) = \max_{x \succeq 0} \left[c^T x + \lambda^T (b - Ax) \right]$$

The second term of the bracket is non-negative when $Ax \preceq b$, so $g(\lambda) \geq p^*$. As in the example above, we hope to achieve equality for some "optimal" price vector. For this, note that

$$g(\lambda) = \lambda^T b + \max_{x \succeq 0} (c - A^T \lambda)^T x = \begin{cases} \lambda^T b & \text{if } A^T \lambda \succeq c \\ \infty & \text{otherwise} \end{cases}$$

Define

$$\begin{aligned} d^* = \min_{\lambda \succeq 0} g(\lambda) = \min \quad & \lambda^T b \\ \text{subject to} \quad & A^T \lambda \succeq c, \quad \lambda \succeq 0 \end{aligned}$$

Notice the symmetry between the optimizations defining p^* and d^*. As we have have seen, it is always true that $p^* \leq d^*$. However, there is a theorem proving that

$$p^* = d^*$$

whenever the inequalities $Ax \preceq b$, $x \succeq 0$ have a feasible solution. In particular, when a finite optimal value p^* exists, there is an "optimal price vector" λ^* such that

$$p^* = \max_{x \succeq 0} \left[c^T x + (\lambda^*)^T (b - Ax) \right]$$

Hence, as in the example, once the price vector λ^* is known all constraints coupling different subsystems to each other can be dropped and each subproblem can be treated separately.

An optimal pair (x^*, λ^*) is characterized by the following conditions, called the Karush-Kuhn-Tucker (KKT) conditions.

$$\begin{aligned} Ax^* &\preceq b & A^T \lambda^* &\succeq c & \\ \lambda^* &\succeq 0 & x^* &\succeq 0 & \\ (A_i x^* - b_i)\lambda_i^* &= 0 & (A_j^T \lambda^* - c_j) x_j^* &= 0 & \text{for all } i, j \end{aligned}$$

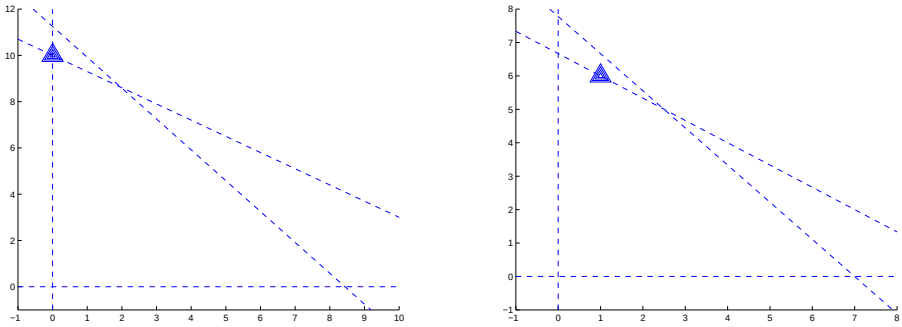

Figure 9.1 Optimal solution for Furniture Division (left) and Sled Division (right). The common painting constraint is active (i.e. equality holds) but not visible in the diagrams.

Dual variables can be interpreted as marginal price for resources: If the capacity for a resource is increased by ϵ, the total profit is increased by ϵ times the corresponding dual variable. This gives insight about which resource is most critical.

Let us now go back to the example and to see how the variables can be interpreted.

Example 2. Each of the constraints in the optimization (9.1) has a corresponding dual variable:

	Constraint	Dual variable
Sawing in furniture division	$7x_1 + 10x_2 \leq 100$	1.04
Assembly in furniture division	$16x_1 + 12x_2 \leq 135$	0
Sawing in sled division	$10x_3 + 9x_4 \leq 70$	0
Assembly in sled division	$6x_3 + 9x_4 \leq 60$	0.4
Painting	$5x_1 + 3x_2 + 3x_3 + 2x_4 \leq 45$	3.2

The optimal value is $p^* = 272$. The most critical capacity constraint is painting. If the painting capacity is increased to 46, the optimal value will become $272 + 3.2 = 275.2$. □

Optimization through Distributed Iterations

We are now ready to return to our example to see how the optimal production planning can be found through distributed iterations.

Example 3. Let $g_1(\lambda)$ and $g_2(\lambda)$ be the maximal values of (9.3) and (9.4) respectively. Given that the price for painting is λ, the furniture division expects to deliver the income $g_1(\lambda)$ while the sled division expects to deliver the income $g_2(\lambda)$. However, the payments for painting from the subdivisions are internal payments collected by the headquarters, so the total expected income for the company becomes

$$45\lambda + g_1(\lambda) + g_2(\lambda)$$

This is also equal to the maximal value of (9.2) and upper bound on the optimal value p^* of (9.1). If the painting price λ is too high, the painting

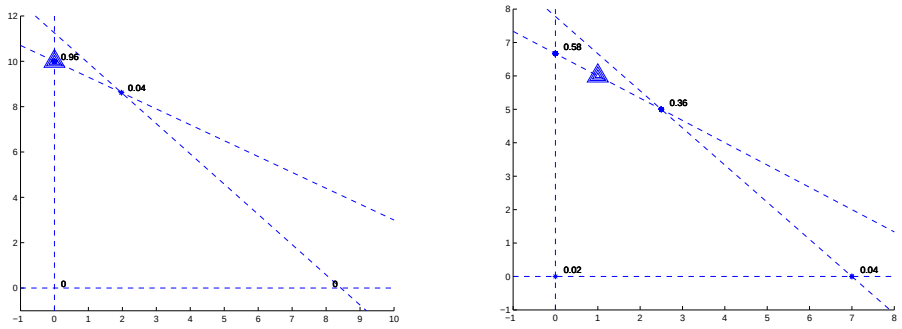

Figure 9.2 Iterates of x_1, x_2 for furniture division (left) and x_3, x_4 for sled division (right) with their respective local constraints. Triangles show optimal solution (which is not in a corner in division 2 due to the painting constraint). The numbers show the fraction of iterates in that corner.

facility will be underutilized and the actual internal payments will be less than the expected 45λ. On the other hand, if the painting price λ is too low, the optimizations (9.3) and (9.4) will create a painting demand that exceeds the capacity 45 and which is therefore infeasible.

The optimal painting price can be found by an iterative scheme. For a given painting price, each of the sub-divisions compute their optimal production volumes. If these production volumes lead to a painting demand that exceeds the capacity, the painting price is raised before the next iteration. On the contrary, if the painting facility becomes underutilized, the painting price is lowered before the next iteration. Mathematically the price update scheme can be written as follows:

1. Initialize algorithm by $\lambda^0 = 0$ and $x^0 = 0$.
2. For fixed $\lambda = \lambda^k$ let the sub-divisions solve (9.3) and (9.4) respectively to find the state vector x^k.
3. Define $\lambda^{k+1} = \max(0, \lambda^k - \alpha^k(45 - 5x_1^k + 3x_2^k + 3x_3^k + 2x_4^k))$
4. Set $k \leftarrow k + 1$ and go to step 2.

An iteration of this scheme is illustrated in Figure 9.2. If instead of the latest state iterate plot the average of the states found so far, we get Figure 9.3. □

To summarize, we have introduced a scheme for decomposition of large optimization problems using dual variables (prices). The scheme is often called dual decomposition and is applicable also to other convex optimization problems than linear programming. It is particularly suitable for problems where relatively few constraints involve all variables.

A simple proof of convergence for the scheme described in the example is given in the note

- *Subgradient Methods* by Stephen Boyd and Almir Mutapcic. Notes for EE364b, Stanford University, Winter 2006-07.

which is available from th course web page. Page 2-6 of this note should be considered as a complement of these lecture notes.

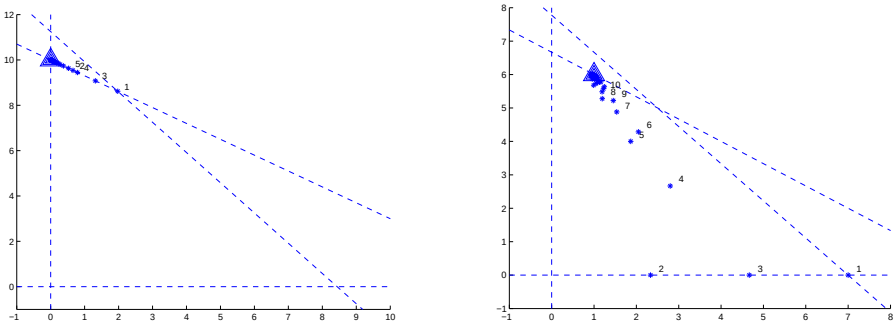

Figure 9.3 If we plot the average of the states found so far, these converge toward the optimal solution. The numbers correspond to iterate number.