Notes on Decomposition Methods

Stephen Boyd, Lin Xiao, and Almir Mutapcic
Notes for EE392o, Stanford University, Autumn, 2003

October 1, 2003

Decomposition is a general approach to solving a problem by breaking it up into smaller ones and solving each of the smaller ones separately, either in parallel or sequentially. (When it is done sequentially, the advantage comes from the fact that problem complexity grows more than linearly.)

Problems for which decomposition works in one step are called (block) *separable*, or *trivially parallelizable*. As a general example of such a problem, suppose the variable x can be partitioned into subvectors $x_1, \ldots, x_k$, the objective is a sum of functions of x_i, and each constraint involves only variables from one of the subvectors x_i. Then evidently we can solve each problem involving x_i separately (and in parallel), and then re-assemble the solution x. Of course this is a trivial, and not too interesting, case.

A more interesting situation occurs when there is some coupling or interaction between the subvectors, so the problems cannot be solved independently. For these cases there are techniques that solve the overall problem by iteratively solving a sequence of smaller problems. There are many ways to do this; in this note we consider some simple examples that illustrate the ideas.

Decomposition is an old idea, and appears in early work on large-scale LPs from the 1960s [DW60]. A good reference on decomposition methods is chapter 6 of Bertsekas [Ber99].

1 Basic (primal) decomposition

We'll consider the simplest possible case, an unconstrained optimization problem that splits into two subproblems. (But note that the most impressive applications of decomposition occur when the problem is split into many subproblems.) In our first example, we consider an unconstrained minimization problem, of the form

$$\text{minimize} \quad f(x) = f_1(u, y) + f_2(v, y) \tag{1}$$

where the variable is $x = (u, v, y)$. Although the dimensions don't matter here, it's useful to think of u and v as having relatively high dimension, and y having relatively small dimenion. The objective is almost block separable in u and v; indeed, if we fix the subvector y, the problem becomes separable in u and v, and therefore can be solved by solving the two subproblems independently. For this reason, y is called the *complicating variable*, because

when it is fixed, the problem splits or decomposes. In other words, the variable y complicates the problem. It is the variable that couples the two subproblems. We can think of u (v) as the *private variable* or *local variable* associated with the first (second) subproblem, and y as the *interface variable* or *boundary variable* between the two subproblems.

The observation that the problem becomes separable when y is fixed suggests a method for solving the problem (1). Let $\phi_1(y)$ denote the optimal value of the problem

$$\text{minimize}_u \quad f_1(u, y), \tag{2}$$

and similarly, let $\phi_2(y)$ denote the optimal value of the problem

$$\text{minimize}_v \quad f_2(v, y). \tag{3}$$

(Note that if f_1 and f_2 are convex, so are $\phi_1(y)$ and $\phi_2(y)$.) We refer to (2) as subproblem 1, and (3) as subproblem 2.

Then the original problem (1) is equivalent to the problem

$$\text{minimize}_y \quad \phi_1(y) + \phi_2(y). \tag{4}$$

This problem is called the *master problem*. If the original problem is convex, so is the master problem. The variables of the master problem are the complicating or coupling variables of the original problem. The objective of the master problem is the sum of the optimal values of the subproblems.

A *decomposition method* solves the problem (1) by solving the master problem, using an iterative method such as the subgradient method. Each iteration requires solving the two subproblems in order to evaluate $\phi_1(y)$ and $\phi_2(y)$ and their gradients or subgradients. This can be done in parallel, but even if it is done sequentially, there will substantial savings if the computational complexity of the problems grows more than linearly.

Let's see how to evaluate a subgradient of ϕ_1 at y, assuming the problem is convex. We first solve the associated subproblem, *i.e.*, we find $\bar{u}(y)$ that minimizes $f_1(u, y)$. Thus, there is a subgradient of f_1 of the form $(0, g)$, and not surprisingly, g is a subgradient of ϕ_1 at y.

We can interpret this decomposition method as follows. We have two subproblems, with *private variables* or *local variables* u and v, respectively. We also have the complicating variable y which appears in both subproblems. At each step of the master algorithm the complicating variable is fixed, which allows the two subproblems to be solved independently. From the two local solutions, we construct a subgradient for the master problem, and using this, we update the complicating variable. Then we repeat the process.

The decomposition method works well when there are few complicating variables, and we have some good or fast methods for solving the subproblems. For example, if one of the subproblems is quadratic, we can solve it analytically; in this case the optimal value is also quadratic, and given by a Schur complement of the local quadratic cost function. (But this trick is so simple that most people would not call it decomposition.)

The basic decomposition method is called *primal decomposition* because the master algorithm manipulates (some of the) primal variables.

2 Dual decomposition

We can apply decomposition to the problem (1) after introducing some new variables, and working with the dual problem. We first express the problem as

$$\begin{array}{ll} \text{minimize} & f(x) = f_1(u, y_1) + f_2(v, y_2) \\ \text{subject to} & y_1 = y_2, \end{array} \tag{5}$$

by introducing a new variable and equality constraint. We have introduced a *local version* of the complicating variable y, along with a *consistency constraint* that requires the two local versions to be equal. Note that the objective is now separable, with the variable partition (u, y_1) and (v, y_2).

Now we form the dual problem. The Lagrangian is

$$L(u, y_1, v, y_2, \nu) = f_1(u, y_1) + f_2(v, y_2) + \nu^T y_1 - \nu^T y_2,$$

which is separable. The dual function is

$$g(\nu) = g_1(\nu) + g_2(\nu),$$

where

$$g_1(\nu) = \inf_{u,y_1} \left(f_1(u, y_1) + \nu^T y_1 \right), \qquad g_2(\nu) = \inf_{v,y_2} \left(f_2(v, y_2) - \nu^T y_2 \right).$$

Note that g_1 and g_2 can be evaluated completely independently, *e.g.*, in parallel. Also note that g_1 and g_2 can be expressed in terms of the conjugates of f_1 and f_2:

$$g_1(\nu) = -f_1^*(0, -\nu), \qquad g_2(\nu) = -f_2^*(0, \nu).$$

The dual problem is

$$\text{maximize} \quad g_1(\nu) + g_2(\nu) = -f_1^*(0, -\nu) - f_2^*(0, \nu). \tag{6}$$

This is the master problem in dual decomposition. The master algorithm solves this problem using a subgradient, cutting-plane, or other method.

To evaluate a subgradient of $-g_1$ (or $-g_2$) is easy. We find $\bar{u}$ and $\bar{y}_1$ that minimize $f_1(u, y_1) + \nu^T y_1$ over u and y_1. Then a subgradient of $-g_1$ at ν is given by $-\bar{y}_1$. Similarly, if $\bar{v}$ and $\bar{y}_2$ minimize $f_2(v, y_2) - \nu^T y_2$ over v and y_2, then a subgradient of $-g_2$ at ν is given by $\bar{y}_2$. Thus, a subgradient of the negative dual function $-g$ is given by $\bar{y}_2 - \bar{y}_1$, which is nothing more than the consistency constraint residual.

Dual decomposition has an interesting economic interpretation. We imagine two separate economic units, each with its own private variables and cost function, but also with some coupled variables. We can think of y_1 as the amounts of some resources consumed by the first unit, and y_2 as the amounts of some resources generated by the second unit. Then, the consistency condition $y_1 = y_2$ means that supply is equal to demand. In primal decomposition, the master algorithm simply fixes the amount of resources to be transfered from one unit to the other, and updates these fixed transfer amounts until the total cost is minimized. In dual decomposition, we interpret ν as a set of prices for the resources. The

master algorithm sets the prices, not the actual amount of the transfer from one unit to the other. Then, each unit independently operates in such a way that its cost, including the cost of the resource transfer (or profit generated from it), is minimized. The dual decomposition master algorithm adjusts the prices in order to bring the supply into consistency with the demand. In economics, the master algorithm is called a price adjustment algorithm, or *tatonnement* procedure.

There is one subtlety in dual decomposition. Even if we do find the optimal prices $\nu^\star$, there is the question of finding the optimal values of u, v, and y. When f_1 and f_2 are strictly convex, the points found in evaluating g_1 and g_2 are guaranteed to converge to optimal, but in general the situation can be more difficult. (For more on finding the primal solution from the dual, see Boyd and Vandenberghe [BV03, §5.5.5].) There are also some standard tricks for regularizing the subproblems that work very well in practice.

3 Constraints

It's very easy to extend the ideas of decomposition to problems with constraints. In primal decomposition, we can include any separable contraints (*i.e.*, ones that affect only u or v, but not both). We can also extend decomposition to handle problems in which there are *complicating constraints*, *i.e.*, constraints that couple the two groups of variables. As a simple example, suppose our problem has the form

$$\begin{array}{ll} \text{minimize} & f_1(u) + f_2(v) \\ \text{subject to} & u \in \mathcal{C}_1 \\ & v \in \mathcal{C}_2 \\ & h_1(u) + h_2(v) \preceq 0. \end{array}$$

Here $\mathcal{C}_1$ and $\mathcal{C}_2$ are the feasible sets of the subproblems, presumably described by linear equalities and convex inequalities. The functions $h_1 : \mathbf{R}^n \to \mathbf{R}^p$ and $h_2 : \mathbf{R}^n \to \mathbf{R}^p$ have components that are convex. The subproblems are coupled via the p (complicating) constraints that involve both u and v.

To use primal decomposition, we can introduce a variable $t \in \mathbf{R}^p$ that represents the amount of the resources allocated to the first subproblem. As a result, $-t$ is allocated to the second subproblem. The first subproblem becomes

$$\begin{array}{ll} \text{minimize} & f_1(u) \\ \text{subject to} & u \in \mathcal{C}_1 \\ & h_1(u) \preceq t, \end{array}$$

and the second subproblem becomes

$$\begin{array}{ll} \text{minimize} & f_2(v) \\ \text{subject to} & v \in \mathcal{C}_2 \\ & h_2(v) \preceq -t. \end{array}$$

The primal decomposition master problem is to minimize the sum of the optimal values of the subproblems, over the variable t. These subproblems can be solved separately, when t

is fixed. The master algorithm updates t, and solves the two subproblems independently to obtain subgradients.

Not surprisingly, we can find a subgradient for the optimal value of each subproblem from an optimal dual variable associated with the coupling constraint. Let $p(z)$ be the optimal value of the convex optimization problem

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in X, \quad h(x) \preceq z, \end{array}$$

and suppose $z \in \mathbf{dom}\, p$. Let λ^* be an optimal dual variable associated with the constraint $h(x) \preceq z$. Then, $-\lambda^*$ is a subgradient of p at z. To see this, we consider the value of p at another point $\tilde{z}$:

$$\begin{aligned} p(\tilde{z}) &= \sup_{\lambda \succeq 0} \inf_{x \in X} \Big(f(x) + \lambda^T (h(x) - \tilde{z}) \Big) \\ &\geq \inf_{x \in X} \Big(f(x) + \lambda^{*T} (h(x) - \tilde{z}) \Big) \\ &= \inf_{x \in X} \Big(f(x) + \lambda^{*T} (h(x) - z + z - \tilde{z}) \Big) \\ &= \inf_{x \in X} \Big(f(x) + \lambda^{*T} (h(x) - z) \Big) + \lambda^{*T} (z - \tilde{z}) \\ &= \phi(z) + (-\lambda^*)^T (\tilde{z} - z). \end{aligned}$$

This holds for all points $\tilde{z} \in Z$, so $-\lambda^*$ is a subgradient of p at z. (See Boyd and Vandenberghe [BV03, §5.6].)

Dual decomposition for this example is straightforward: The Lagrangian is separable, so we can minimize over u and v separately, given the dual variable λ. The master algorithm updates λ.

4 Decomposition structures

It's possible to have far more complex decomposition structures. For example, the variables might be partitioned into subvectors, some of which are local (*i.e.*, appear in one term of a sum objective) and some of which are complicating (*i.e.*, appear in, say, two terms of the objective). This can be represented by an undirected graph. The nodes are associated with the local variables (and objectives), and the links are associated with complicating variables. Thus, the links represent coupling between the subproblems. In each step of the master algorithm of a primal decomposition method, the link values are fixed, which allows the subproblems to be solved independently. Then, the master algorithm adjusts the complicating variables on the links in such a way that the overall objective will (eventually) improve. In a dual decomposition method, each link is associated with a price vector that is updated at each step of the master algorithm. Our simple example above is represented by a very simple graph: two nodes with one link connecting them.

As a slightly more interesting case, consider an optimal control problem,

$$\begin{array}{ll} \text{minimize} & \sum_{t=0}^{T-1} \phi_t(u(t)) + \sum_{t=1}^{T} \psi_t(x(t)) \\ \text{subject to} & x(t+1) = A(t)x(t) + B(t)u(t), \quad t = 0, \ldots, T-1, \end{array}$$

with variables $u(0), \ldots, u(T-1)$, and $x(1), \ldots, x(T)$, with initial state $x(0)$ given. The functions ϕ_t and ψ_t are convex control and state costs, respectively. For the optimal control problem we can give a very nice interpretation of the decomposition structure: the state is the complicating variable between the past and the future. In other words, if you fix the state in a dynamical system, the past and future have nothing to do with each other. (That's exactly what it means to be a state.) In terms of a decomposition graph, we have nodes associated with $(u(0), x(1)), \ldots, (u(T-1), x(T))$. Each of these is linked to the node before and after, by the state equations $x(t+1) = A(t)x(t) + B(t)u(t)$. Thus, the optimal control problem decomposition structure is represented by a simple linear chain with T nodes.

Primal decomposition for optimal control would fix some of the states, which chops up the optimal control problem into a set of smaller ones, which can be solved separately. We then update the fixed states so as to lower (hopefully) the overall objective. If we apply dual decomposition to the optimal control problem, we find that the dual variables are exactly the variables in the adjoint control problem. Evaluating the gradient for the master (dual) problem can be done recursively, starting from $\nu(T)$ and running backwards in time. Thus, we recover some standard algorithms for optimal control.

Decomposition structure arises in many applications. In network problems, for example, we can partition the network into subnets, that interact only via common flows or their boundary connections. In some image processing problems, pixels are only coupled to some of their neighbors. In this case, any strip with a width exceeding the interaction distance between pixels, and which disconnects the image plane can be taken as a set of complicating variables. You can solve an image restoration problem, then, by fixing a strip (say, down the middle), and then (in parallel) solving the left and right image problems. (This can clearly be done recursively as well.)

Decomposition is important in hierarchical design as well. Suppose we are designing (via convex optimization) a large circuit (say) that consists of some subcircuits. Each subcircuit has many private variables, and a few variables that interact with other subcircuits. For example, the device dimensions inside each subcircuit might be local or private variables; the shared variables correspond to electrical connections between the subcircuits (*e.g.*, the load presented to one subcircuit from another) or objectives or constraint that couple them (*e.g.*, a total power or area limit). We also suppose that for each subcircuit, we have a method for designing it, provided the shared variables are fixed. Primal decomposition then corresponds to hierarchical design. At each step of the master algorithm, we fix the coupling variables, and then ask each subcircuit to design itself. We then update the coupling variables in such as way that the total cost (say, power) eventually is minimized.

5 An LP example

To illustrate the idea of decomposition, we consider the LP

$$\begin{array}{ll} \text{minimize} & c^T u + \tilde{c}^T v \\ \text{subject to} & Au \preceq b \\ & \tilde{A} v \preceq \tilde{b} \\ & Fu + \tilde{F} v \preceq h, \end{array} \tag{7}$$

with variables u and v. The two subproblems are coupled by the constraints $Fu + \tilde{F}v \preceq h$, which might represent limits on some shared resources. Of course we can solve the problem as one LP; decomposition allows us solve it with a master algorithm, and solving two LPs (possibly in parallel) at each iteration. Decomposition is most attractive in the case when the two subproblems are relatively large, and there are just a few coupling constraints, *i.e.*, the dimension of h is relatively small.

5.1 Primal decomposition (allocation)

We introduce the complicating variable z to decouple the constraint between u and v. The coupling constraint $Fu + \tilde{F}v \preceq h$ becomes

$$Fu \preceq z, \qquad \tilde{F}v \preceq h - z.$$

The original problem is equivalent to the problem

$$\text{minimize}_z \quad \phi(z) + \tilde{\phi}(z)$$

where

$$\begin{array}{rcl} \phi(z) & = & \inf_u \ \{c^T u \mid Au \preceq b, \ Fu \preceq z\} \\ \tilde{\phi}(z) & = & \inf_v \ \{\tilde{c}^T v \mid \tilde{A}v \preceq b, \ \tilde{F}v \preceq h - z\}. \end{array} \tag{8}$$

The values of $\phi(z)$ and $\tilde{\phi}(z)$ can be evaluated by solving two LP subproblems. Let $\lambda(z)$ be an optimal dual variable for the constraint $Fu \preceq z$, and $\tilde{\lambda}(z)$ be an optimal dual variable for the constraint $\tilde{F}v \preceq h - z$. A subgradient of the function $\phi(z) + \tilde{\phi}(z)$ is then given by

$$g(z) = -\lambda(z) + \tilde{\lambda}(z).$$

We assume here that for any z, the two subproblems are feasible. It's not hard to extend the ideas to the case when one, or both, is infeasible.

Primal decomposition combined with the subgradient method for the master problem gives the following algorithm:

repeat

Solve the subproblems.

Solve the LPs (8) to obtain optimal u, v, and associated dual variables λ, $\tilde{\lambda}$.

Master algorithm subgradient. $g := -\lambda + \tilde{\lambda}$.

Master algorithm update. $z := z - \alpha_k g$.

Here k denotes the iteration number, and α_k is the subgradient step size rule, which is any nonsummable positive sequence that converges to zero. In the primal decomposition algorithm, we have feasible u and v at each step; as the algorithm proceeds, they converge to optimal. (We assume here that the two subproblems are always feasible. It's not hard to modify the method to work when this isn't the case.)

The algorithm has a simple interpretation. At each step the master algorithm fixes the allocation of each resource, between the two subproblems. The two subproblems are then

solved (independently). The optimal Lagrange multiplier $-\lambda_i^*$ tells us how much worse the objective of the first subproblem would be, for a small decrease in resource i; $\tilde{\lambda}_i^*$ tells us how much better the objective of the second subproblem would be, for a small increase in resource i. Thus, $g_i = -\lambda_i^* + \tilde{\lambda}_i^*$ tells us how much better the total obective would be if we transfer some of resource i from the first to the second subsystem. The resource allocation update $z_i := z_i - \alpha_k g_i$ simply shifts some of resource i to the subsystem that can make better use of it.

Now we illustrate primal decomposition with a specific LP problem instance, with $n_u = n_v = 10$ variables, $m_u = m_v = 100$ private inequalities and $p = 5$ complicating inequalities. The problem data A, $\tilde{A}$, F, and $\tilde{F}$ were generated from a unit normal distribution, while c, $\tilde{c}$, b, $\tilde{b}$, and h were generated from a unit uniform distribution.

We use the subgradient method with diminishing step size rule to solve master problem. Figure 1 shows master problem's objective function value $\phi(z) + \tilde{\phi}(z)$ versus iteration number k when $\alpha_k = 0.1/\sqrt{k}$. Figure 2 shows convergence of the primal residual.

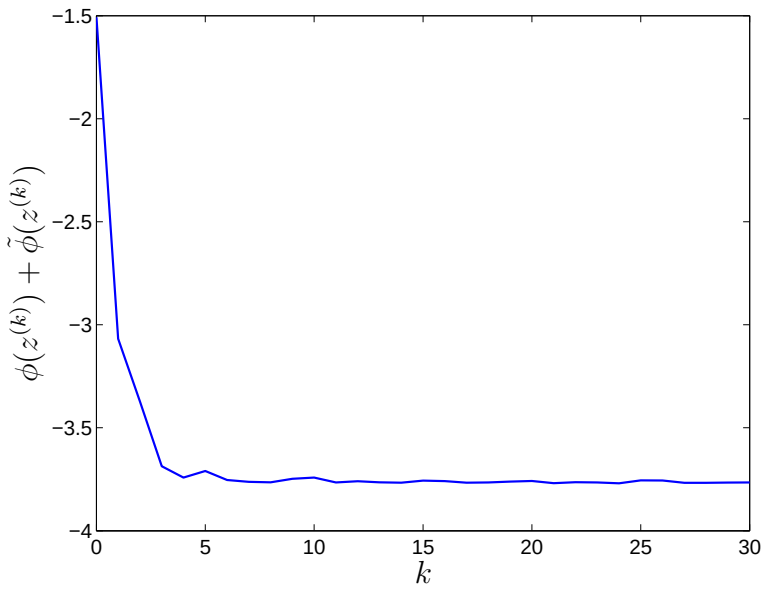

Figure 1: Objective function value versus iteration number k, when master problem is solved with subgradient method using diminishing rule $\alpha_k = 0.1/\sqrt{k}$.

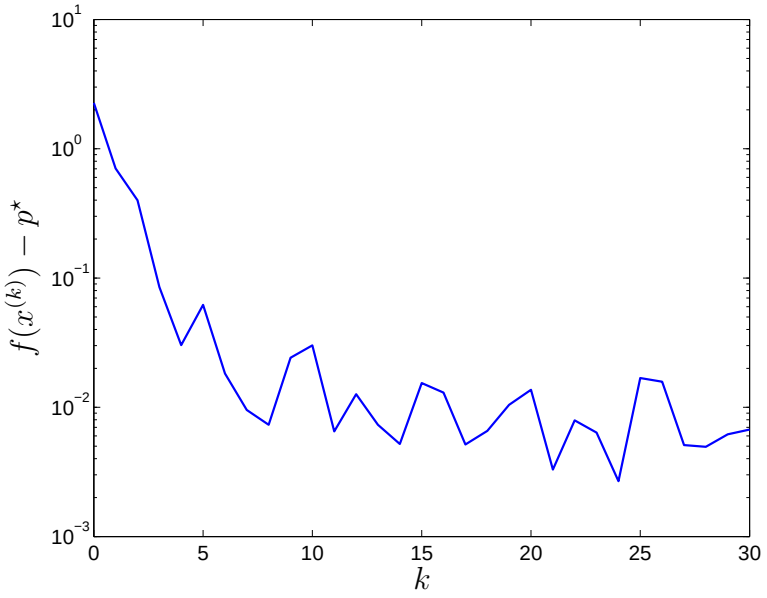

Figure 2: Primal residual versus iteration number k, when master problem is solved with subgradient method using diminishing rule $\alpha_k = 0.1/\sqrt{k}$.

5.2 Dual decomposition (pricing)

We first form the partial Lagrangian, by introducing Lagrange multipliers only for the coupling constraint $Fu + \tilde{F}v \preceq h$:

$$\begin{aligned} L(u, v, \lambda) &= c^T u + \tilde{c}^T v + \lambda^T (Fu + \tilde{F}v - h) \\ &= (F^T\lambda + c)^T u + (\tilde{F}^T\lambda + \tilde{c})^T v - \lambda^T h. \end{aligned}$$

The dual function is

$$\begin{aligned} q(\lambda) &= \inf_{u,v} \{L(u, v, \lambda) \mid Au \preceq b,\ \tilde{A}v \preceq \tilde{b}\} \\ &= -\lambda^T h + \inf_{Au \preceq b} (F^T\lambda + c)^T u + \inf_{\tilde{A}v \preceq \tilde{b}} (\tilde{F}^T\lambda + \tilde{c})^T v. \end{aligned}$$

The dual optimization problem is

$$\begin{array}{ll} \text{maximize} & q(\lambda) \\ \text{subject to} & \lambda \succeq 0. \end{array}$$

We'll solve this dual problem using the projected subgradient method, which requires a subgradient of $-q$ at each iteration. Given λ, we can evaluate the dual function by solving two separate linear programs:

$$\begin{array}{ll} \text{minimize} & (F^T\lambda + c)^T u \\ \text{subject to} & Au \preceq b \end{array} \quad \text{and} \quad \begin{array}{ll} \text{minimize} & (\tilde{F}^T\lambda + \tilde{c})^T v \\ \text{subject to} & \tilde{A}v \preceq \tilde{b}. \end{array} \tag{9}$$

Let the optimal solutions to the linear programs be $\bar{u}$ and $\bar{v}$ respectively. A subgradient of $-q$ is given by

$$g = -F\bar{u} - \tilde{F}\bar{v} + h.$$

Dual decomposition, with subgradient method for the master problem gives the following algorithm:

repeat

Solve the subproblems. Solve the two LP subproblems (9) to obtain optimal $\bar{u}$, $\bar{v}$.
Master algorithm subgradient. $g = -F\bar{u} - \tilde{F}\bar{v} + h$.
Master algorithm update. $\lambda := (\lambda - \alpha_k g)_+$.

Here $(\cdot)_+$ denotes the nonnegative part of a vector, *i.e.*, projection onto the nonnegative orthant.

The interpretation of this dual decomposition algorithm is as follows. At each step, the master algorithm sets the prices for the resources. The subsystems each optimize, independently, but taking into account the expense of using the resources, or income generated from not using the resource. The subgradient $g = -Fu - \tilde{F}v + h$ is nothing more than the margin of the original shared coupling constraint $Fu + \tilde{F}v \preceq h$. If $g_i < 0$, then too much of resource i is being consumed by the susbsystems; if $g_i > 0$, then it is possible for the two subsystems to use more of resource i. The master algorithm adjusts the prices in a very simple way: the

price for each resource that is over used is increased; the price for each resource that is not over the limit is decreased, but never made negative.

We use the subgradient method with diminishing step size rule $\alpha_k = 1/\sqrt{k}$ to solve master pricing problem, for the exampel considered above. Figure 3 shows the dual function value $q(\lambda^{(k)})$ versus iteration number k, while figure 4 shows the dual residual.

References

[Ber99] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, second edition, 1999.

[BV03] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2003.

[DW60] G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8:101–111, 1960.

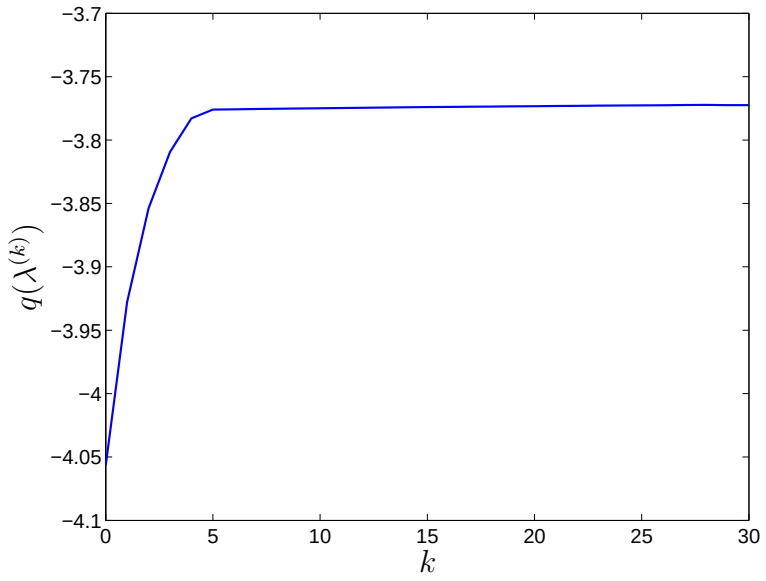

Figure 3: Dual function value $q(\lambda^{(k)})$ versus iteration number k, when master problem is solved with subgradient method using diminishing rule $\alpha_k = 1/\sqrt{k}$.

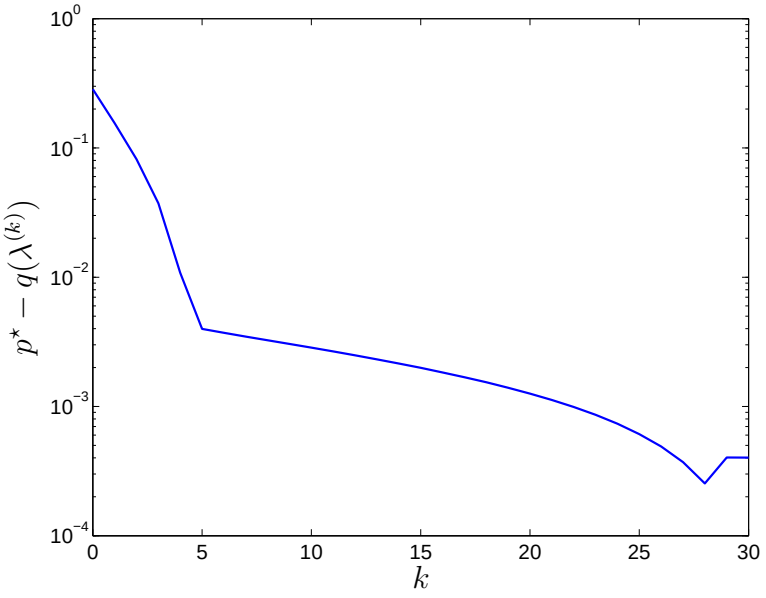

Figure 4: Dual residual versus iteration number k, when master problem is solved with subgradient method using diminishing rule $\alpha_k = 1/\sqrt{k}$.