Simplicial decomposition

Consider the problem to

$$\begin{aligned} &\text{minimize } f(x), && \text{(1a)}\\ &\text{subject to } x \in S, && \text{(1b)}\end{aligned}$$

where $S \subset \Re^n$ is a polyhedron and $f : \Re^n \mapsto \Re$ is a continuously differentiable function. *Simplicial decomposition* builds on the knowledge that a polyhedron has an inner representation in terms of extreme points and directions (Theorem 4.2 in Nash/Sofer): Suppose that $S = \{\, x \in \Re^n \mid Ax = b; \quad x \geq 0 \,\}$ is non-empty, and is bounded (for simplicity of the presentation). The set of extreme points is finite:

$$V = \{v_1, v_2, \ldots, v_I\}.$$

Each point $x \in S$ can be written as a convex combination of the extreme points:

$$x \in S \Longleftrightarrow x = \sum_{i=1}^{I} \alpha^i v_i,$$

where

$$\begin{aligned} \sum_{i=1}^{I} \alpha^i &= 1,\\ \alpha^i &\geq 0, \qquad i = 1, 2, \ldots, I. \end{aligned}$$

The advantage of this representation is that it is *much* simpler to deal with in an optimization algorithm than the linear constraints that define S above; essentially, we have only non-negativity left, since the "$\sum_{i=1}^{I} \alpha^i = 1$" constraint can be dealt with by the substitution of one of the variables.

The disadvantage of the representation is that the set V is both very large for a large-scale problem, and it is not known; compare with the case of the simplex method, where we cannot simply enumerate the extreme points in order to then pick the best one. In this nonlinear case, we may also need a large number of them in order to "span" an optimal solution. The trick is to consider the variables α^i to be zero for all those elements in V that we do not know, and to generate those that seem profitable by solving the linear Frank–Wolfe subproblem: at a feasible solution x_k we generate an extreme point by solving the problem to

$$\text{minimize } z_k(y) := f(x_k) + \nabla f(x_k)^{\mathrm{T}}(y - x_k), \tag{2a}$$
$$\text{subject to } y \in S. \tag{2b}$$

The method is as follows:

0. Choose an *initial solution*, $x_0 \in S$. Let $k := 0$.

   Here one typically chooses an arbitrary basic feasible solution, that is, an extreme point corresponding to an index i in $\{1, 2, \ldots, I\}$. We call this index i_0. Let $\hat{I} := \{i_0\}$ and $x_0 := v_{i_0}$.

1. Generate a new *extreme point*.

   Solve the LP problem (2). This problem gives us as an optimal solution the extreme point $i_k \in I$. If $i_k \in \hat{I}$ then the algorithm stops, because x_k is then stationary. (WHY?) Otherwise, it is a new extreme point (WHY?), whose inclusion will improve the solution, and we let $\hat{I} := \hat{I} \cup \{i_k\}$.

2. Solve the *restriction* to the original problem to the set $\hat{I} \subset I$.

   Solve the problem to

   $$\text{minimize } f(x),$$

   subject to

   $$x = \sum_{i \in \hat{I}} \alpha^i v_i,$$
   $$\sum_{i \in \hat{I}} \alpha^i = 1,$$
   $$\alpha^i \geq 0, \qquad i \in \hat{I}.$$

   The optimum is a vector α_k whose corresponding vector in the original space is $x_{k+1} := \sum_{i \in \hat{I}} \alpha_k^i v_i$.

3. If a *stopping criterion* is fulfilled $\longrightarrow$ Stop! x_{k+1} then is an approximation of x_*. Otherwise, let $k := k + 1$, and go to 1.

The algorithm is quite similar to the Frank–Wolfe algorithm. The main, and very important, difference, is that the Frank–Wolfe algorithm drops all the previous extreme points visited, and only optimizes over line segments. In Simplicial Decomposition, the extreme point information is stored—we could say that the algorithm has a "memory" of where it has been—and thanks to this extra information the algorithm can make much better progress: instead of the line search in the Frank–Wolfe algorithm, Step 2 is a *multi-dimensional* search over all the extreme points generated. The method therefore becomes much more efficient than the Frank–Wolfe algorithm in practice.