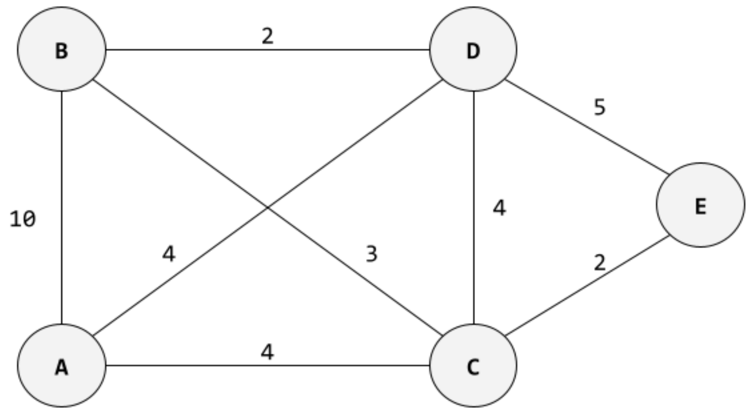
1 Simulated Annealing (Section 11.6)

This is typically an algorithm for a discrete search. This particular algoritihm does not use derivatives, either.

(Example: Traveling Salesperson.) The problem statement is as follows: Suppose we have a network of different cities. Our goal is to find the minimum cost to visit each city once (and only once).



For large networks, multiple local minima may exists.

1.1 A Brute Force Approach

Suppose $F(x) : \mathbb{R}^n \to \mathbb{R}$ represents the cost of traveling a particular route. The simulated annealing also uses random components. At iteration k, given $F(x^{(k)})$ and m, we want to generate candidates $u_1, u_2, u_3, \ldots, u_m \in \mathbb{R}^n$. This is typically done by random process: in the traveling salesperson problem, we consider the different edges we could take. By evaluating the function at each candidate point, we can find *one* of them that has the least cost; that is,

$$u_j = \min_{1 \leq i \leq m} F(u_i).$$

If $F(u_j) \leq F(x^{(k)})$, then we can update the iterate,

$$x^{(k+1)} = u_j.$$

Otherwise, we can do the following:

- Compute the probability for each u_i,

$$\tilde{p}_i = e^{\alpha(F(x^{(k)}) - F(u_i))} \qquad q \leq i \leq m, \quad \alpha > 0, \quad \tilde{p}_i \in \mathbb{R}.$$

- Rescale (normalize) the probability,

$$\tilde{p}_i = \frac{\tilde{p}_i}{\sum_{k=1}^{m} \tilde{p}_k}.$$

- For a random uniformly distributed variable $\rho \in [0, 1]$, we want to find the index ℓ such that

$$\tilde{p}_1 + \tilde{p}_2 + \ldots + \tilde{p}_{\ell-1} \leq \rho \leq \tilde{p}_1 + \tilde{p}_2 + \ldots + \tilde{p}_{\ell-1} + \tilde{p}_\ell.$$

  Based on the index, we have

$$x^{(k+1)} = u_\ell.$$

1.2 Coordinate Descent & Pattern Search

This method is somewhat similar to the Nelder-Mead method. It also does not use derivatives. Coordinate descent aims to minimize one variable at a time, when $F : \mathbb{R}^n \to \mathbb{R}$. The idea is to cycle through the m coordinate directions, corresponding to $e_1, e_2, \dots, e_n$. Recall that $e_i = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$, with the 1 being in the ith position. We want to line-search along each coordinate direction. The iteration relies on the finding of α_k,

$$\alpha_k = \min_{\alpha} F(x^{(k)} + \alpha e_k).$$

From there,

$$x^{(k+1)} = x^{(k)} + \alpha_k e_k \qquad k = 1, 2, \dots.$$

When $k = 1$, we count forward. When $k = m$, we count backwards. So, we would do something like

$$e_1, e_2, \dots, e_{n-1}, e_n, e_{n-1}, e_{n-2}, \dots, e_2, e_1, e_2, \dots$$

One modification we can do to this process is to search along a line after a few iterations of the method. However, this modification converges pretty slowly.

1.3 Pattern Search

We can incorporate pattern-search methods to generalize coordinate descent. Suppose there is a set of directions, $p_k \in D_k$. Here, D_k is called the *direction set*, and p_k is an $n \times 1$ vector representing a direction. The pattern-search methods are not only dependent on the direction vector, but also a fixed step size α_k.

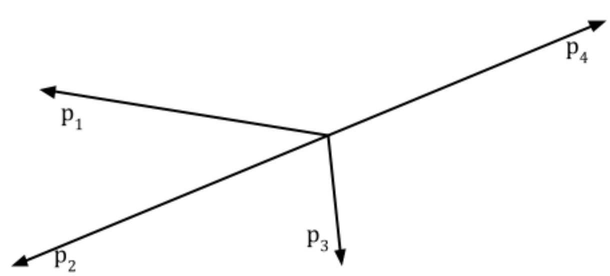
To update α_k, we'll consider a *frame* of all directions. We'll update the current value by considering every direction. The frame is defined by

$$x^{(k)} + \alpha_k p_k, \qquad p_k \in D_k.$$

(Example.) Suppose our direction set is defined by

$$D_k = \{p_i : 1 \le i \le n\} \cup \{p_{n+1}\},$$

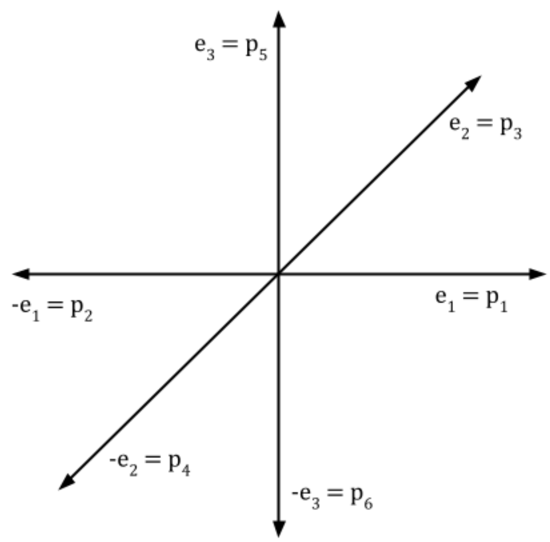
with $p_i = \frac{1}{2m}e - e_i$; and, $p_{n+1} = \frac{1}{2m}e$. This is known as the simplex direction set, and its frame looks like



(Example.) In $\mathbb{R}^3$, suppose our direction set is

$$D_k = \{e_1, e_2, e_3, -e_1, -e_2, -e_3\}.$$

Then, our frame is defined by



The corresponding algorithm takes the following arguments

- $\epsilon > 0$, the tolerance,
- $1 > \beta > 0$, the contraction,
- $\alpha > \epsilon$,
- $\gamma \geq 1$, the expansion
- D_0, the initial direction set,
- $M \geq 0$, the reduction measure.

Algorithm 1 Pattern Search Method

function PATTERNSEARCH$(\epsilon, \beta, \alpha, \gamma, D_0, M)$
    for $k \leftarrow 1$ to ∞ **do**
        if $\alpha \leq \epsilon$ **then**
            break
        end if
        if $F(x^{(k)} + \alpha) < F(x^{(k)}) - M\alpha^3$ **then**
            $p_k \in D_k$
            $x^{(k+1)} \leftarrow x^{(k)} + \alpha p_k$ ▷ For some such p_k
            $\alpha \leftarrow \gamma\alpha$
        else
            $x^{(k+1)} \leftarrow x^{(k)}$
            $\alpha \leftarrow \beta\alpha$
        end if
    end for
end function

1.4 Line Search

Recall that the line search was used for selecting the step length. In this context, we'll say that the updates of solution estimates are of the form

$$x^{(k+1)} = x^{(k)} + \alpha_k p_k,$$

where α_k is a scalar representing the step length and p_k is a vector representing the direction. To determine a step length, typically a 1D search is done. Given a direction p_k and $F(x) : \mathbb{R}^n \to \mathbb{R}$, $\phi : \mathbb{R} \to \mathbb{R}$, we have

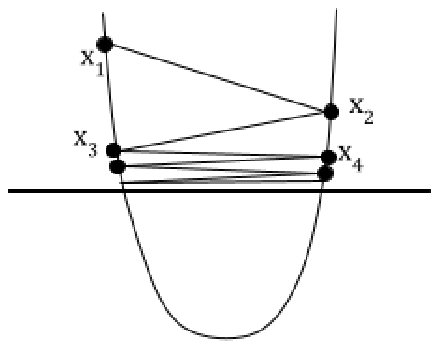
$$\phi(\alpha) = F(x^{(k)} + \alpha p_k)$$

$$\phi'(\alpha) = p_k^T \nabla F(x^{(k)} + \alpha p_k).$$

Note that p_k should be a descent direction; that is, $p_k^T \nabla F(x_k) \leq 0$. Now, to minimize ϕ (i.e., to approximate $\min_{\alpha>0} \phi(\alpha)$), a set of conditions are used:

- **Condition 1:** "Sufficient" decrease.

(Example.) Note that $F(x^{(k+1)}) < F(x^{(k)})$ alone isn't enough for this condition. To see why, consider $F(x) = x^2 - 1$. Suppose we have a sequence $\{x_k\}$ so that $F(x_k) = 4/k$.



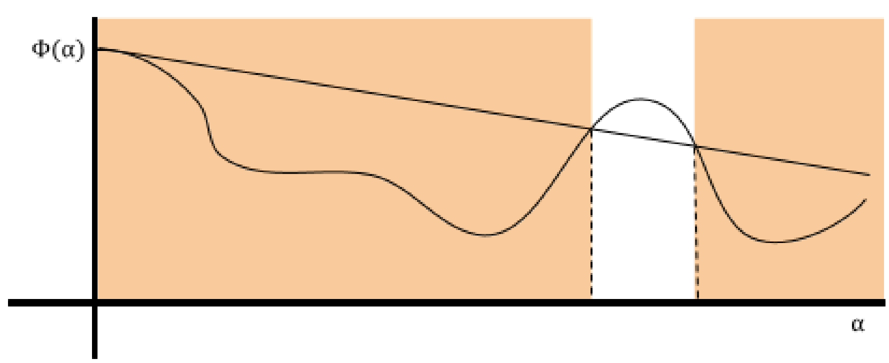
Notice that $F(x_{k+1}) < F(x_k)$, but the result does not converge to the minimum.

With this example in mind, a sufficient decrease is one that satisfies

$$\phi(\alpha_k) \leq \phi(0) + \alpha_k c_1 \phi'(0)$$

$$F(x^{(k)} + \alpha_k p_k) \leq F(x^{(k)}) + \alpha_k c_1 \nabla F^T(x^{(k)}) p_k.$$

Here, $c_1 \in (0, 1)$ is some parameter, with the most common value being $c_1 = 10^{-4}$; this means that the line often looks flat. Visually, this looks like



The orange region denotes a sufficient decrease (where ϕ is less than or equal to the value of the line). The line can be defined by $\ell(\alpha) = \phi(0) + \alpha c_1 \phi'(0)$. Note that the step α may be very small.

- **Condition 2:** Curvature condition. In particular, we have

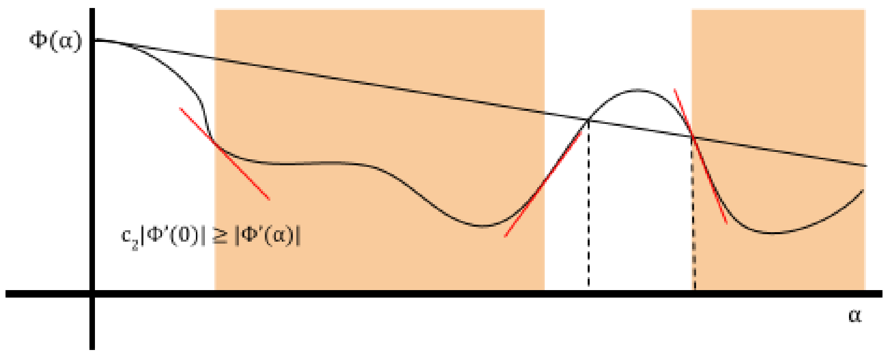
$$|\phi'(\alpha_k) \leq c_2 |\phi'(0)|, \qquad c_2 \in (c_1, 1).$$

With this in mind, we'll introduce the **Wolfe-Conditions**, which is essentially a combination of the above two conditions. For step lengths, we have

$$F(x^{(k)} + \alpha p_k) \leq F(x^{(k)}) + c_1 \alpha \nabla F(x^{(k)})^T p_k,$$

$$|\nabla F(x^{(k)} + \alpha p_k)^T p_k| \leq c_2 |\nabla F(x^{(k)})^T p_k|.$$

Visually, we want to add tangent lines to the points corresponding to $c_2|\phi'(0)| \geq |\phi'(\alpha)|$.



Here, the orange region denotes the region satisfied by the Wolfe-Condition. Wolfe line search is effective in practice, but generally difficult to implement.

1.4.1 Armijo Line Search

A generally effective simple line search method is the Armijo Search, also known as a backtracking line search. This is for a sufficient decrease, where the step size is not too small. In particular, for $\alpha = 1 > 0$, $c_1 = 10^{-4}$, and $\rho = \frac{1}{2}$, we have

Algorithm 2 Armijo Line Search

```
while F(x^(k) + αp_k) > F(x^(k)) + c_1 α ∇F(x^(k))^T p_k do
    α ← ρα
end while
```

1.4.2 Wolfe Line Search

The Wolfe Line Search algorithm has the following arguments:

- $\alpha_0 = 0$
- $\alpha_{\max} > 0$ (e.g., 100)
- $\alpha_1 \in (0, \alpha_{\max})$ (e.g., 1)
- $c_1 = 0.9$
- $c_2 = 10^{-4}$

Algorithm 3 Wolfe Line Search

function WOLFE($\alpha_i, \alpha_{\max}, c_1, c_2$)
    $i \leftarrow 0$
    while true **do**
        if $\phi(\alpha_i) > \phi(0) + c_1\alpha_i\phi'(0)$ or $\phi(\alpha_i) \geq \phi(\alpha_{i-1})$ and $i > 1$ **then**
            $\alpha^* \leftarrow \text{zoom}(\alpha_{i-1}, \alpha_i)$
            break
        end if
        if $\phi'(\alpha_i) \geq 0$ **then**
            $\alpha^* = \text{zoom}(\alpha_i, \alpha_{i-1})$
            break
        end if
        $\alpha_{i+1} \leftarrow 2\alpha_i$
        $i \leftarrow i + 1$
    end while
    $\text{zoom}(\alpha_{\text{low}}, \alpha_{\text{high}})$
    while true **do**
        $\alpha_j = \frac{1}{2}(\alpha_{\text{low}} + \alpha_{\text{high}})$
        if $\phi(\alpha_j) > \phi(0) + c_1\alpha_j\phi'(0)$ or $\phi(\alpha_j) \geq \phi(\alpha_{\text{low}})$ **then**
            $\alpha_{\text{high}} \leftarrow \alpha_j$
        else
            if $\phi'(\alpha_j)| \leq -c_2\phi'(0)$ **then**
                $\alpha^* \leftarrow \alpha_j$
                break
            end if
            if $\phi'(\alpha_j) - (\alpha_{\text{high}} - \alpha_{\text{low}}) \geq 0$ **then**
                $\alpha_{\text{high}} \leftarrow \alpha_{\text{low}}$
            end if
            $\alpha_{\text{low}} \leftarrow \alpha_j$
        end if
    end while
end function