

# Lecture 11: Linearly constrained NLP

Michael Patriksson

2010-11-26

## Feasible-direction methods, I

- Consider the problem to find

$$f^* = \text{infimum } f(\mathbf{x}), \quad (1a)$$

$$\text{subject to } \mathbf{x} \in X, \quad (1b)$$

$X \subseteq \mathbb{R}^n$  nonempty, closed and convex;  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  is  $C^1$  on  $X$

- Most methods for (1) manipulate (that is, **relax**) the constraints defining  $X$ ; in some cases even such that the sequence  $\{\mathbf{x}_k\}$  is infeasible until convergence. Why?

# Feasible-direction methods, I

- Consider a constraint " $g_i(\mathbf{x}) \leq b_i$ ," where  $g_i$  is nonlinear
- Checking whether  $\mathbf{p}$  is a feasible direction at  $\mathbf{x}$ , or what the maximum feasible step from  $\mathbf{x}$  in the direction of  $\mathbf{p}$  is, is very difficult
- For which step length  $\alpha > 0$  is  $g_i(\mathbf{x} + \alpha\mathbf{p}) = b_i$ ? This is a nonlinear equation in  $\alpha$ !
- Assuming that  $X$  is polyhedral, these problems are not present
- Note: KKT always necessary for a local min for polyhedral sets; methods will find such points

# Feasible-direction descent methods

- Step 0. Determine a *starting point*  $\mathbf{x}_0 \in \mathbb{R}^n$  such that  $\mathbf{x}_0 \in X$ . Set  $k := 0$
- Step 1. Determine a *search direction*  $\mathbf{p}_k \in \mathbb{R}^n$  such that  $\mathbf{p}_k$  is a feasible descent direction
- Step 2. Determine a *step length*  $\alpha_k > 0$  such that  $f(\mathbf{x}_k + \alpha_k \mathbf{p}_k) < f(\mathbf{x}_k)$  and  $\mathbf{x}_k + \alpha_k \mathbf{p}_k \in X$
- Step 3. Let  $\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k \mathbf{p}_k$
- Step 4. If a *termination criterion* is fulfilled, then stop!  
Otherwise, let  $k := k + 1$  and go to Step 1

# Notes

- Similar form as the general method for unconstrained optimization
- Just as *local* as methods for unconstrained optimization
- Search directions typically based on the approximation of  $f$ —a “relaxation”
- Search direction often of the form  $\mathbf{p}_k = \mathbf{y}_k - \mathbf{x}_k$ , where  $\mathbf{y}_k \in X$  solves an approximate problem
- Line searches similar; note the maximum step
- Termination criteria and descent based on first-order optimality and/or fixed-point theory ( $\mathbf{p}_k \approx \mathbf{0}^n$ )

## LP-based algorithm, I: The Frank–Wolfe method

- The Frank–Wolfe method is based on a first-order approximation of  $f$  around the iterate  $\mathbf{x}_k$ . This means that the relaxed problems are LPs, which can then be solved by using the Simplex method
- Remember the first-order optimality condition: *If  $\mathbf{x}^* \in X$  is a local minimum of  $f$  on  $X$  then*

$$\nabla f(\mathbf{x}^*)^\top (\mathbf{x} - \mathbf{x}^*) \geq 0, \quad \mathbf{x} \in X,$$

*holds*

- Remember also the following equivalent statement:

$$\underset{\mathbf{x} \in X}{\text{minimum}} \nabla f(\mathbf{x}^*)^\top (\mathbf{x} - \mathbf{x}^*) = 0$$

## LP-based algorithm, I: The Frank–Wolfe method

- Follows that if, given an iterate  $\mathbf{x}_k \in X$ ,

$$\text{minimum}_{\mathbf{y} \in X} \nabla f(\mathbf{x}_k)^\top (\mathbf{y} - \mathbf{x}_k) < 0,$$

and  $\mathbf{y}_k$  is an optimal solution to this LP problem, then the direction of  $\mathbf{p}_k := \mathbf{y}_k - \mathbf{x}_k$  is a feasible descent direction with respect to  $f$  at  $\mathbf{x}$

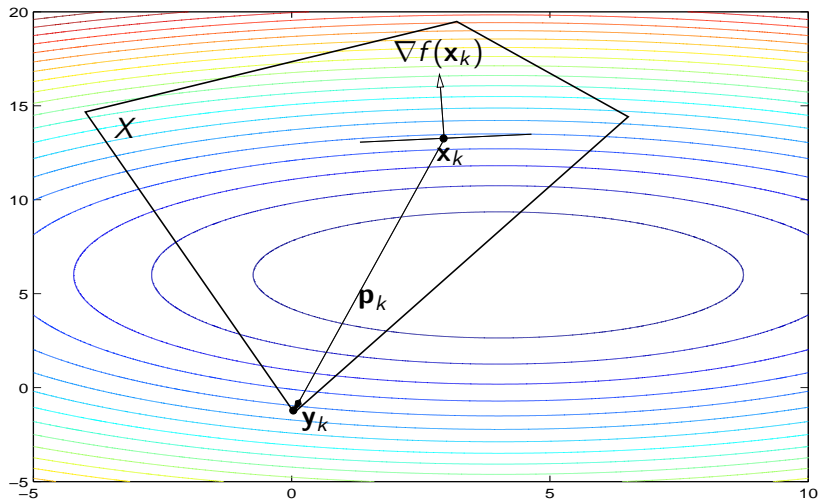
- Search direction towards an extreme point of  $X$  [one that is optimal in the LP over  $X$  with costs  $\mathbf{c} = \nabla f(\mathbf{x}_k)$ ]
- This is the basis of the *Frank–Wolfe algorithm*

# LP-based algorithm, I: The Frank–Wolfe method

- We assume that  $X$  is bounded in order to ensure that the LP always has a finite optimal solution. The algorithm can be extended to allow for unbounded polyhedra
- The search directions then are either towards an extreme point (finite optimal solution to LP) or in the direction of an extreme ray of  $X$  (unbounded solution to LP)
- Both cases identified in the Simplex method



# The search-direction problem



# Algorithm description, Frank-Wolfe

**Step 0.** Find  $\mathbf{x}_0 \in X$  (for example any extreme point in  $X$ ).  
Set  $k := 0$

**Step 1.** Find an optimal solution  $\mathbf{y}_k$  to the problem to

$$\underset{\mathbf{y} \in X}{\text{minimize}} \quad z_k(\mathbf{y}) := \nabla f(\mathbf{x}_k)^T (\mathbf{y} - \mathbf{x}_k) \quad (2)$$

Let  $\mathbf{p}_k := \mathbf{y}_k - \mathbf{x}_k$  be the search direction

**Step 2.** Approximately solve the problem to minimize  $f(\mathbf{x}_k + \alpha \mathbf{p}_k)$  over  $\alpha \in [0, 1]$ . Let  $\alpha_k$  be the step length

**Step 3.** Let  $\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k \mathbf{p}_k$

**Step 4.** If, for example,  $z_k(\mathbf{y}_k)$  or  $\alpha_k$  is close to zero, then terminate! Otherwise, let  $k := k + 1$  and go to Step 1

# \*Convergence

- *Suppose  $X \subset \mathbb{R}^n$  nonempty polytope;  $f$  in  $C^1$  on  $X$*
- *In Step 2 of the Frank–Wolfe algorithm, we either use an exact line search or the Armijo step length rule*
- *Then: the sequence  $\{\mathbf{x}_k\}$  is bounded and every limit point (at least one exists) is stationary;*
- *$\{f(\mathbf{x}_k)\}$  is descending, and therefore has a limit;*
- *$z_k(\mathbf{y}_k) \rightarrow 0$  ( $\nabla f(\mathbf{x}_k)^T \mathbf{p}_k \rightarrow 0$ )*
- *If  $f$  is convex on  $X$ , then every limit point is globally optimal*

# The convex case: Lower bounds

- Remember the following characterization of convex functions in  $C^1$  on  $X$ :  $f$  is convex on  $X \iff$

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^T(\mathbf{y} - \mathbf{x}), \quad \mathbf{x}, \mathbf{y} \in X$$

- Suppose  $f$  is convex on  $X$ . Then,  $f(\mathbf{x}_k) + z_k(\mathbf{x}_k) \leq f^*$  (lower bound, LBD), and  $f(\mathbf{x}_k) + z_k(\mathbf{x}_k) = f^*$  if and only if  $\mathbf{x}_k$  is globally optimal. *A relaxation—cf. the Relaxation Theorem!*
- Utilize the lower bound as follows: we know that  $f^* \in [f(\mathbf{x}_k) + z_k(\mathbf{x}_k), f(\mathbf{x}_k)]$ . Store the best LBD, and check in Step 4 whether  $[f(\mathbf{x}_k) - \text{LBD}]/|\text{LBD}|$  is small, and if so terminate

# Notes

- Frank-Wolfe uses linear approximations—works best for almost linear problems
- For highly nonlinear problems, the approximation is bad—the optimal solution may be far from an extreme point
- In order to find a near-optimum requires many iterations—the algorithm is slow
- Another reason is that the information generated (the extreme points) is forgotten. If we keep the linear subproblem, we can do much better by storing and utilizing this information

## LP-based algorithm, II: Simplicial decomposition

- Remember the Representation Theorem (special case for polytopes): Let  $P = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{Ax} = \mathbf{b}; \mathbf{x} \geq \mathbf{0}^n\}$ , be nonempty and bounded, and  $V = \{\mathbf{v}^1, \dots, \mathbf{v}^K\}$  be the set of extreme points of  $P$ . Every  $\mathbf{x} \in P$  can be represented as a convex combination of the points in  $V$ , that is,

$$\mathbf{x} = \sum_{i=1}^K \alpha_i \mathbf{v}^i,$$

for some  $\alpha_1, \dots, \alpha_K \geq 0$  such that  $\sum_{i=1}^K \alpha_i = 1$

## LP-based algorithm, II: Simplicial decomposition

- The idea behind the Simplicial decomposition method is to generate the extreme points  $\mathbf{v}^i$  which can be used to describe an optimal solution  $\mathbf{x}^*$ , that is, the vectors  $\mathbf{v}^i$  with positive weights  $\alpha_i$  in

$$\mathbf{x}^* = \sum_{i=1}^K \alpha_i \mathbf{v}^i$$

- The process is still iterative: we generate a “working set”  $\mathcal{P}_k$  of indices  $i$ , optimize the function  $f$  over the convex hull of the known points, and check for stationarity and/or generate a new extreme point

# Algorithm description, Simplicial decomposition

**Step 0.** Find  $\mathbf{x}_0 \in X$ , for example any extreme point in  $X$ .  
Set  $k := 0$ . Let  $\mathcal{P}_0 := \emptyset$

**Step 1.** Let  $\mathbf{y}_k$  be an optimal solution to the LP problem (2)  
Let  $\mathcal{P}_{k+1} := \mathcal{P}_k \cup \{k\}$



## Algorithm description, Simplicial decomposition

Step 2. Let  $(\mu_k, \nu_{k+1})$  be an approximate solution to the *restricted master problem* (RMP) to

$$\underset{(\mu, \nu)}{\text{minimize}} \quad f \left( \mu \mathbf{x}_k + \sum_{i \in \mathcal{P}_{k+1}} \nu_i \mathbf{y}^i \right), \quad (3a)$$

$$\text{subject to} \quad \mu + \sum_{i \in \mathcal{P}_{k+1}} \nu_i = 1, \quad (3b)$$

$$\mu, \nu_i \geq 0, \quad i \in \mathcal{P}_{k+1} \quad (3c)$$

Step 3. Let  $\mathbf{x}_{k+1} := \mu_{k+1} \mathbf{x}_k + \sum_{i \in \mathcal{P}_{k+1}} (\nu_{k+1})_i \mathbf{y}^i$

Step 4. If, for example,  $z_k(\mathbf{y}_k)$  is close to zero, or if  $\mathcal{P}_{k+1} = \mathcal{P}_k$ , then terminate! Otherwise, let  $k := k + 1$  and go to Step 1

# Algorithm description, Simplicial decomposition

- This basic algorithm keeps all information generated, and adds one new extreme point in every iteration
- An alternative is to drop columns (vectors  $\mathbf{y}^i$ ) that have received a zero (or, low) weight, or to keep only a maximum number of vectors
- Special case: maximum number of vectors kept = 1  $\implies$  the Frank–Wolfe algorithm!
- We obviously improve the Frank–Wolfe algorithm by utilizing more information
- Compare with the difference between Newton and steepest descent in unconstrained optimization

## \*Convergence

- It does at least as well as the Frank–Wolfe algorithm: line segment  $[\mathbf{x}_k, \mathbf{y}_k]$  feasible in RMP
- If  $\mathbf{x}^*$  unique then convergence is finite if the RMPs are solved exactly, and the maximum number of vectors kept is  $\geq$  the number needed to span  $\mathbf{x}^*$
- Much more efficient than the Frank–Wolfe algorithm in practice (consider the above FW example!)
- We can solve the RMPs efficiently, since the constraints are simple

# An illustration of FW vs. SD, I

- A large-scale nonlinear network flow problem which is used to estimate traffic flows in cities
- Model over the small city of Sioux Falls in North Dakota, USA; 24 nodes, 76 links, and 528 pairs of origin and destination
- Three algorithms for the RMPs were tested—a Newton method and two gradient projection methods (see the next section). A MATLAB implementation
- Remarkable difference—The Frank-Wolfe method suffers from very small steps being taken. Why? Many extreme points active = many routes used

## An illustration of FW vs. SD, I

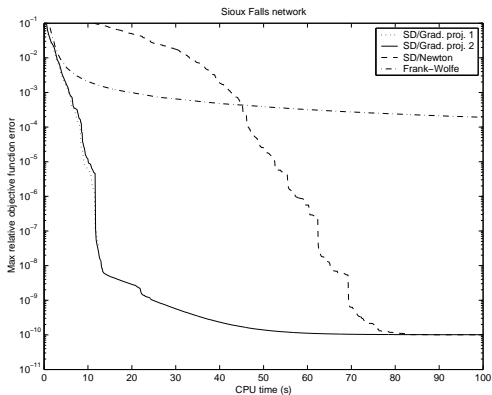
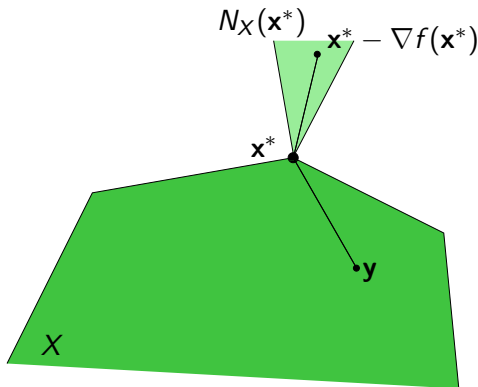


Figure: The performance of SD vs. FW on the Sioux Falls network

# QP-based algorithm: The gradient projection algorithm

- The gradient projection algorithm is based on the projection characterization of a stationary point:  $\mathbf{x}^* \in X$  is a stationary point if and only if, for any  $\alpha > 0$ ,

$$\mathbf{x}^* = \text{Proj}_X[\mathbf{x}^* - \alpha \nabla f(\mathbf{x}^*)]$$



# QP-based algorithm: The gradient projection algorithm

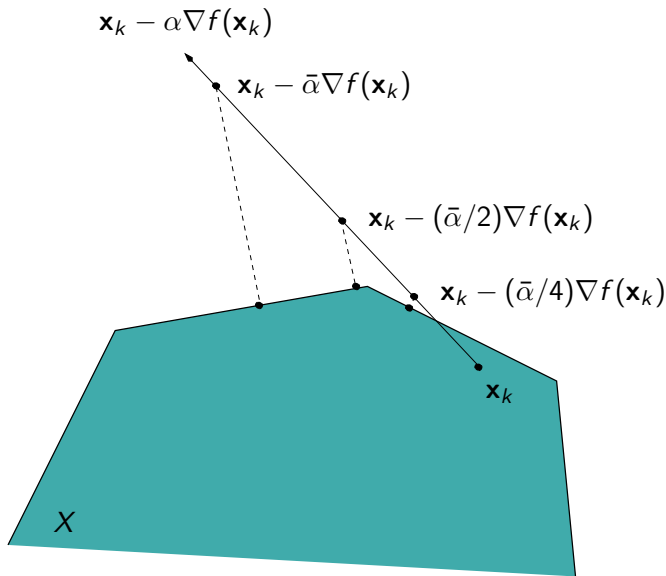
- Let  $\mathbf{p} := \text{Proj}_X[\mathbf{x} - \alpha \nabla f(\mathbf{x})] - \mathbf{x}$ , for any  $\alpha > 0$ . Then, if and only if  $\mathbf{x}$  is non-stationary,  $\mathbf{p}$  is a feasible descent direction of  $f$  at  $\mathbf{x}$
- The gradient projection algorithm is normally stated such that the line search is done over the *projection arc*, that is, we find a step length  $\alpha_k$  for which

$$\mathbf{x}_{k+1} := \text{Proj}_X[\mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k)], \quad k = 1, \dots \quad (4)$$

has a good objective value. Use the Armijo rule to determine  $\alpha_k$

- Note: gradient projection becomes steepest descent with Armijo line search when  $X = \mathbb{R}^n$ !

## QP-based algorithm: The gradient projection algorithm





## \*Convergence, I

- $X \subseteq \mathbb{R}^n$  nonempty, closed, convex;  $f \in C^1$  on  $X$ ;
- for the starting point  $\mathbf{x}_0 \in X$  it holds that the level set  $\text{lev}_f(f(\mathbf{x}_0))$  intersected with  $X$  is bounded
- In the algorithm (4), the step length  $\alpha_k$  is given by the Armijo step length rule along the projection arc
- Then: the sequence  $\{\mathbf{x}_k\}$  is bounded;
- every limit point of  $\{\mathbf{x}_k\}$  is stationary;
- $\{f(\mathbf{x}_k)\}$  descending, lower bounded, hence convergent
- Convergence arguments similar to steepest descent one

## \*Convergence, II

- Assume:  $X \subseteq \mathbb{R}^n$  nonempty, closed, convex;
- $f \in C^1$  on  $X$ ;  $f$  convex;
- an optimal solution  $\mathbf{x}^*$  exists
- In the algorithm (4), the step length  $\alpha_k$  is given by the Armijo step length rule along the projection arc
- Then: the sequence  $\{\mathbf{x}_k\}$  converges to an optimal solution
- Note: with  $X = \mathbb{R}^n \implies$  convergence of steepest descent for convex problems with optimal solutions!