10.1 TYPES OF CONSTRAINED OPTIMIZATION ALGORITHMS
Quadratic Programming Problems

• Algorithms for such problems are interested to explore because
  - 1. Their structure can be efficiently exploited.
  - 2. They form the basis for other algorithms, such as augmented Lagrangian and Sequential quadratic programming problems.

\[
\min_{x} \quad q(x) = \frac{1}{2} x^T G x + x^T c \\
\text{subject to} \quad a_i^T x = b_i, \quad i \in \mathcal{E}, \\
\quad a_i^T x \geq b_i, \quad i \in \mathcal{I},
\]
Penalty Methods

- Idea: Replace the constraints by a penalty term.
- Inexact penalties: parameter driven to infinity to recover solution. Example:
  \[ x^* = \text{arg min } f(x) \text{ subject to } c(x) = 0 \iff x^\mu = \text{arg min } f(x) + \frac{\mu}{2} \sum_{i \in \mathcal{E}} c_i^2(x); \ x^* = \lim_{\mu \to \infty} x^\mu = x^* \]
- Exact but nonsmooth penalty – the penalty parameter can stay finite.

\[ x^* = \text{arg min } f(x) \text{ subject to } c(x) = 0 \iff x^* = \text{arg min } f(x) + \mu \sum_{i \in \mathcal{E}} |c_i(x)|; \ \mu \geq \mu_0 \]
Augmented Lagrangian Methods

• Mix the Lagrangian point of view with a penalty point of view.

\[
x^* = \arg \min f(x) \text{ subject to } c(x) = 0 \iff \\
x^{\mu, \lambda} = \arg \min f(x) - \sum_{i \in \mathcal{E}} \lambda_i c_i(x) + \frac{\mu}{2} \sum_{i \in \mathcal{E}} c_i^2(x) \Rightarrow \\
x^* = \lim_{\lambda \to \lambda^*} x^{\mu, \lambda} \text{ for some } \mu \geq \mu_0 > 0
\]
Sequential Quadratic Programming Algorithms

- Solve successively Quadratic Programs.

\[
\min_p \quad \frac{1}{2} p^T B_k p + \nabla f(x_k)
\]

subject to \( \nabla c_i(x_k) d + c_i(x_k) = 0 \quad i \in \mathcal{E} \)

\( \nabla c_i(x_k) d + c_i(x_k) \geq 0 \quad i \in \mathcal{I} \)

- It is the analogous of Newton’s method for the case of constraints if

\[
B_k = \nabla^2_{xx} \mathcal{L}(x_k, \lambda_k)
\]

- But how do you solve the subproblem? It is possible with extensions of simplex which I do not cover.

- An option is BFGS which makes it convex.
Interior Point Methods

• Reduce the inequality constraints with a barrier

\[ \min_{x,s} \quad f(x) - \mu \sum_{i=1}^{m} \log s_i \]

subject to \[ c_i(x) = 0 \quad i \in \mathcal{E} \]
\[ c_i(x) - s_i = 0 \quad i \in \mathcal{I} \]

• An alternative, is use to use a penalty as well:

\[ \min_{x} \quad f(x) - \mu \sum_{i \in \mathcal{I}} \log s_i + \frac{1}{2\mu} \sum_{i \in \mathcal{I}} (c_i(x) - s)^2 + \frac{1}{2\mu} \sum_{i \in \mathcal{E}} (c_i(x))^2 \]

• And I can solve it as a sequence of unconstrained problems!
10.2 MERIT FUNCTIONS AND FILTERS
Feasible algorithms

- If I can afford to maintain feasibility at all steps, then I just monitor decrease in objective function.
- I accept a point if I have enough descent.
- But this works only for very particular constraints, such as linear constraints or bound constraints (and we will use it).
- Algorithms that do that are called **feasible algorithms**.
Infeasible algorithms

• But, sometimes it is VERY HARD to enforce feasibility at all steps (e.g. nonlinear equality constraints).
• And I need feasibility only in the limit; so there is benefit to allow algorithms to move on the outside of the feasible set.
• But then, how do I measure progress since I have two, apparently contradictory requirements:
  – Reduce infeasibility (e.g. \( \sum_{i \in \mathcal{E}} |c_i(x)| + \sum_{i \in \mathcal{I}} \max \{-c_i(x), 0\} \))
  – Reduce objective function.
  – It has a multiobjective optimization nature!
10.2.1 MERIT FUNCTIONS
• One idea also from multiobjective optimization: minimize a weighted combination of the 2 criteria.

\[
\phi(x) = w_1 f(x) + w_2 \left[ \sum_{i \in \mathcal{E}} |c_i(x)| + \sum_{i \in \mathcal{I}} \max\{-c_i(x), 0\} \right]; \quad w_1, w_2 > 0
\]

• But I can scale it so that the weight of the objective is 1.

• In that case, the weight of the infeasibility measure is called “penalty parameter”.

• I can monitor progress by ensuring that \( \phi(x) \) decreases, as in unconstrained optimization.
Nonsmooth Penalty Merit Functions

\[ \phi_1(x; \mu) = f(x) + \mu \sum_{i \in \mathcal{E}} |c_i(x)| + \mu \sum_{i \in \mathcal{I}} [c_i(x)]^- , \quad [z]^- = \max\{0, -z\}. \]

- It is called the l1 merit function.
- Sometimes, they can be even EXACT.

**Definition 15.1 (Exact Merit Function).**

A merit function \( \phi(x; \mu) \) is exact if there is a positive scalar \( \mu^* \) such that for any \( \mu > \mu^* \), any local solution of the nonlinear programming problem (15.1) is a local minimizer of \( \phi(x; \mu) \).

We show in Theorem 17.3 that, under certain assumptions, the \( \ell_1 \) merit function \( \phi_1(x; \mu) \) is exact and that the threshold value \( \mu^* \) is given by

\[ \mu^* = \max\{|\lambda_i^*|, \ i \in \mathcal{E} \cup \mathcal{I}\}, \]
Smooth and Exact Penalty Functions

- Excellent convergence properties, but very expensive to compute.
- Fletcher’s augmented Lagrangian:

\[ \phi_F(x; \mu) = f(x) - \lambda(x)^T c(x) + \frac{1}{2} \mu \sum c_i(x)^2, \]

\[ \lambda(x) = [A(x)A(x)^T]^{-1} A(x)\nabla f(x). \]

- It is both smooth and exact, but perhaps impractical due to the linear solve.
Augmented Lagrangian

- Smooth, but inexact. \[ \phi(x) = f(x) - \sum_{i \in \mathcal{E}} \lambda_i c_i(x) + \frac{\mu}{2} \sum_{i \in \mathcal{E}} c_i^2(x) \Rightarrow \]

- An update of the Lagrange Multiplier is needed.
- We will not use it, except with Augmented Lagrangian methods themselves.
Line-search (Armijo) for Nonsmooth Merit Functions

\[ \phi_1(x; \mu) = f(x) + \mu \sum_{i \in \mathcal{E}} |c_i(x)| + \mu \sum_{i \in \mathcal{I}} [c_i(x)]^- , \]

- How do we carry out the “progress search”?
- That is the line search or the sufficient reduction in trust region?
- In the unconstrained case, we had

- But we cannot use this anymore, since the function is not differentiable.

\[ f(x_k) - f(x_k + \beta^m d_k) \geq -\rho \beta^m \nabla f(x_k)^T d_k ; \quad 0 < \beta < 1, 0 < \rho < 0.5 \]
Directional Derivatives of Nonsmooth Merit Function

\[ \phi_1(x; \mu) = f(x) + \mu \sum_{i \in \mathcal{E}} |c_i(x)| + \mu \sum_{i \in \mathcal{I}} [c_i(x)]^-, \]

- Nevertheless, the function has a directional derivative (follows from properties of max function). EXPAND

\[
D(\phi(x, \mu); p) = \lim_{t \to 0, t > 0} \frac{\phi(x + tp, \mu) - \phi(x, \mu)}{t}; \quad D(\max\{f_1, f_2\}, p) = \max\{\nabla f_1 p, \nabla f_1 p\}
\]

- Line Search: \[ \phi(x_k, \mu) - \phi(x_k + \beta^m p_k, \mu) \geq -\rho \beta^m D(\phi(x_k, \mu), p_k); \]

- Trust Region \[ \phi(x_k, \mu) - \phi(x_k + \beta^m p_k, \mu) \geq -\eta_1 (m(0) - m(p_k)); \]
  \[ 0 < \eta_1 < 0.5 \]
And .... How do I choose the penalty parameter?

• VERY tricky issue, highly dependent on the penalty function used.

• For the l1 function, guideline is:

\[ \mu^* = \max\{|\lambda_i^*|, \ i \in \mathcal{E} \cup \mathcal{I}|, \]  

• But almost always adaptive. Criterion: If optimality gets ahead of feasibility, make penalty parameter more stringent.

• E.g l1 function: the max of current value of multipliers plus safety factor (EXPAND)
10.2.2 FILTER APPROACHES
Principles of filters

- Originates in the multiobjective optimization philosophy: objective and infeasibility

\[ h(x) = \sum_{i \in \mathcal{E}} |c_i(x)| + \sum_{i \in \mathcal{I}} [c_i(x)]^- , \]

- The problem

\[ \min_x f(x) \quad \text{and} \quad \min_x h(x). \]
**Definition 15.2.**

(a) A pair \((f_k, h_k)\) is said to dominate another pair \((f_l, h_l)\) if both \(f_k \leq f_l\) and \(h_k \leq h_l\).

(b) A filter is a list of pairs \((f_l, h_l)\) such that no pair dominates any other.

(c) An iterate \(x_k\) is said to be acceptable to the filter if \((f_k, h_k)\) is not dominated by any pair in the filter.
Some Refinements

• Like in the line search approach, I cannot accept EVERY decrease since I may never converge.

• Modification:

A trial iterate $x^+$ is acceptable to the filter if, for all pairs $(f_j, h_j)$ in the filter, we have that

$$f(x^+) \leq f_j - \beta h_j \quad \text{or} \quad h(x^+) \leq h_j - \beta h_j, \quad \beta \sim 10^{-5} \quad (15.33)$$
10.3 MARATOS EFFECT AND CURVILINEAR SEARCH
Unfortunately, the Newton step may not be compatible with penalty

- This is called the Maratos effect.
- Problem:

\[
\min f(x_1, x_2) = 2(x_1^2 + x_2^2 - 1) - x_1,
\]

\[
x_1^2 + x_2^2 - 1 = 0.
\]

- Note: the closest point on search direction (Newton) will be rejected!
- So fast convergence does not occur
• Use Fletcher’s function that does not suffer from this problem.

• Following a step:

\[ A_k p_k + c(x_k) = 0. \]

• Use a correction that satisfies

\[ A_k \hat{p}_k + c(x_k + p_k) = 0. \]

\[ \hat{p}_k = -A_k^T (A_k A_k^T)^{-1} c(x_k + p_k), \]

• Followed by the update or line search:

\[ x_k + p_k + \hat{p}_k \]

\[ x_k + \tau p_k + \tau^2 \hat{p}_k \]

• Since

\[ c(x_k + p_k + \hat{p}_k) = O\left(\|x_k - x^*\|^3\right) \]

compared to

\[ c(x_k + p_k) = O\left(\|x_k - x^*\|^2\right) \]

corrected Newton step is likelier to be accepted.