ACM 113 Introduction to Optimization - Final Exam

Ling Li, ling@cs.caltech.edu

June 6, 2001

1 The Baywatch Problem

As shown in Figure 1, denote the position of the victim as (a,b). The lifeguard is at $(0,-d)$ $(d>0)$, and will enter the sea at point $(x,0)$. The problem is

$$\min_x f(x) = \frac{\sqrt{x^2+d^2}}{v_1} + \frac{\sqrt{(a-x)^2+b^2}}{v_2}.$$

We need to solve $f'(x) = \frac{x}{v_1\sqrt{x^2+d^2}} + \frac{x-a}{v_2\sqrt{(a-x)^2+b^2}} = 0$ to get the minimizer x^*.

If represented in θ_1 and θ_2 $(-\frac{\pi}{2} < \theta_1, \theta_2 < \frac{\pi}{2})$, the problem is

$$\min_\theta f(\theta) = \frac{d}{v_1\cos\theta_1} + \frac{b}{v_2\cos\theta_2},$$

subject to $d\tan\theta_1 + b\tan\theta_2 = a$. The Lagrangian is

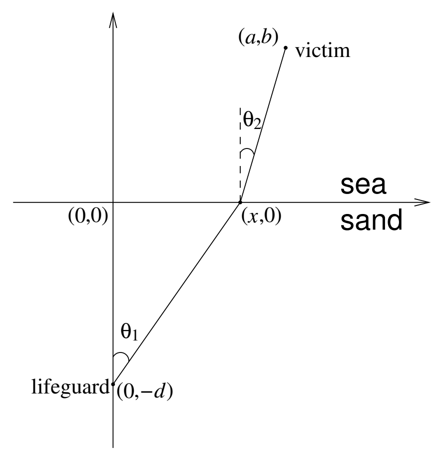


Figure 1: The path of the lifeguard.

$$\mathcal{L}(\theta,\lambda) = \frac{d}{v_1\cos\theta_1} + \frac{b}{v_2\cos\theta_2} - \lambda(d\tan\theta_1 + b\tan\theta_2 - a).$$

Solving

$$\nabla_\theta \mathcal{L}(\theta,\lambda) = \begin{bmatrix} \frac{d\sin\theta_1}{v_1\cos^2\theta_1} - \frac{\lambda d}{\cos^2\theta_1} \\ \frac{b\sin\theta_2}{v_2\cos^2\theta_2} - \frac{\lambda b}{\cos^2\theta_2} \end{bmatrix} = 0$$

gives $\sin\theta_1 = \lambda v_1$ and $\sin\theta_2 = \lambda v_2$. If $a \neq 0$, then the constraint requests $\lambda \neq 0$, so

$$\frac{\sin\theta_1}{\sin\theta_2} = \frac{v_1}{v_2},$$

which means the lifeguard should make a larger angle where his/her speed is faster.

2 Local Convergence of Trust Region Methods

a) The Cauchy point is

$$p_k^c = -\lambda^* \frac{\nabla f_k}{\|\nabla f_k\|}, \quad 0 \le \lambda^* \le \Delta_k,$$

and p_k^c minimizes $m_k(p)$ subject to $\|p\| \le \Delta_k$ along direction $-\nabla f_k$. Thus

$$\begin{aligned} m_k(0) - m_k(p_k^c) &= -\nabla f_k^T p_k^c - \frac{1}{2} p_k^{c\,T} \nabla^2 f_k p_k^c \\ &= \lambda^* \|\nabla f_k\| - \frac{1}{2} \lambda^{*2} \frac{\nabla f_k^T \nabla^2 f_k \nabla f_k}{\|\nabla f_k\|^2} \\ &\ge \lambda \|\nabla f_k\| - \frac{1}{2} \lambda^2 \frac{\nabla f_k^T \nabla^2 f_k \nabla f_k}{\|\nabla f_k\|^2} \quad \text{for any } 0 \le \lambda \le \Delta_k. \end{aligned} \tag{1}$$

- If $\nabla f_k^T \nabla^2 f_k \nabla f_k \le 0$, for $\lambda = \Delta_k$ in (1), we have

$$m_k(0) - m_k(p_k^c) \ge \Delta_k \|\nabla f_k\| \ge \frac{1}{2} \|\nabla f_k\| \cdot \min\left(\Delta_k, \frac{\|\nabla f_k\|}{\|\nabla^2 f_k\|}\right).$$

- Otherwise, $\nabla f_k^T \nabla^2 f_k \nabla f_k > 0$. From (ref. Homework 2.1)

$$\frac{\nabla f_k^T \nabla^2 f_k \nabla f_k}{\|\nabla f_k\|^2} \le \left\|\nabla^2 f_k\right\|,$$

  we get from (1),

$$\begin{aligned} m_k(0) - m_k(p_k^c) &\ge \lambda \|\nabla f_k\| - \frac{1}{2} \lambda^2 \left\|\nabla^2 f_k\right\| \\ &= \frac{1}{2} \frac{\|\nabla f_k\|^2}{\|\nabla^2 f_k\|} - \frac{\left\|\nabla^2 f_k\right\|}{2} \left(\lambda - \frac{\|\nabla f_k\|}{\|\nabla^2 f_k\|}\right)^2. \end{aligned} \tag{2}$$

  If $\frac{\|\nabla f_k\|}{\|\nabla^2 f_k\|} \le \Delta_k$, then for $\lambda = \frac{\|\nabla f_k\|}{\|\nabla^2 f_k\|}$, (2) is

$$m_k(0) - m_k(p_k^c) \ge \frac{1}{2} \|\nabla f_k\| \frac{\|\nabla f_k\|}{\|\nabla^2 f_k\|} = \frac{1}{2} \|\nabla f_k\| \cdot \min\left(\Delta_k, \frac{\|\nabla f_k\|}{\|\nabla^2 f_k\|}\right);$$

  otherwise for $\lambda = \Delta_k$, (2) becomes

$$\begin{aligned} m_k(0) - m_k(p_k^c) &\ge \Delta_k \|\nabla f_k\| - \frac{\Delta_k^2}{2} \left\|\nabla^2 f_k\right\| \\ &= \frac{1}{2} \|\nabla f_k\| \Delta_k + \frac{\Delta_k}{2} \left\|\nabla^2 f_k\right\| \left(\frac{\|\nabla f_k\|}{\|\nabla^2 f_k\|} - \Delta_k\right) \\ &\ge \frac{1}{2} \|\nabla f_k\| \Delta_k = \frac{1}{2} \|\nabla f_k\| \cdot \min\left(\Delta_k, \frac{\|\nabla f_k\|}{\|\nabla^2 f_k\|}\right). \end{aligned}$$

So, overall, we have

$$m_k(0) - m_k(p_k^c) \ge \frac{1}{2} \|\nabla f_k\| \cdot \min\left(\Delta_k, \frac{\|\nabla f_k\|}{\|\nabla^2 f_k\|}\right).$$

In the CG-Newton trust region method, we initialize the inner CG by $p^{(0)} = 0$ and $d^{(0)} = -\nabla f_k$. If negative curvature is met during the first check, $p_k = p_k^c$ is returned. Otherwise, the CG iteration starts from p_k^c,* and ensures descent. So the CG-Newton method yields at least much reduction as p_k^c.

*During the first iteration of CG, $\alpha^{(0)} = \frac{r^{(0)\,T} r^{(0)}}{d^{(0)\,T} B_k d^{(0)}} = \frac{\nabla f_k^T \nabla f_k}{\nabla f_k^T \nabla^2 f_k \nabla f_k}$ and $p^{(1)} = p^{(0)} + \alpha^{(0)} d^{(0)} = -\frac{\|\nabla f_k\|^2}{\nabla f_k^T \nabla^2 f_k \nabla f_k} \nabla f_k$ minimizes $m_k(p)$ along $-\nabla f_k$ without $\|p\| \le \Delta_k$ bound. Thus the following $\left\|p^{(1)}\right\| \ge \Delta_k$ check ensures $p^{(1)} = p_k^c$, or returns $p_k = p_k^c$.

b) During the CG-Newton iterations, $r_k = \nabla^2 f_k p_k + \nabla f_k$. Since the region constraint is inactive, the stopping criterion requests $\|r_k\| \le \eta_k \|\nabla f_k\|$. Thus

$$\begin{aligned}
\left\|p_k - p_k^N\right\| &= \left\|\nabla^2 f_k^{-1}(\nabla^2 f_k p_k + \nabla f_k)\right\| = \left\|\nabla^2 f_k^{-1} r_k\right\| \\
&\le \eta_k \left\|\nabla^2 f_k^{-1}\right\| \|\nabla f_k\| \\
&\le \eta_k \left\|\nabla^2 f_k^{-1}\right\| \left\|\nabla^2 f_k\right\| \left\|p_k^N\right\| = \eta_k \kappa_k \left\|p_k^N\right\|,
\end{aligned}$$

where κ_k is the condition number of $\nabla^2 f_k$, which is bounded for k sufficient large. Since $\eta_k \to 0$, so $\left\|p_k - p_k^N\right\| = o\left(\left\|p_k^N\right\|\right)$.

3 LP Sensitivity Analysis

The primal-dual optimality conditions are

$$Ax = b, \quad A^T y + s = c, \quad x \ge 0, \quad s \ge 0, \quad x^T s = 0.$$

The current basis is not affected iff these conditions are not affected by the perturbation.

a) $b \to b + \Delta b$. This doesn't change y and s. If $B^{-1}(b + \Delta b) = x_B + B^{-1}\Delta b \ge 0$, the basis is not affected. However, x_B is changed by $B^{-1}\Delta b$ and the objective is changed by $c_B^T B^{-1}\Delta b = y^T \Delta b$. If $x_B + B^{-1}\Delta b \ngeq 0$, the basis is affected. Since y, s have not been affected, (x, y, s) is primal infeasible but still dual feasible. We can use the dual simplex algorithm to restore the primal feasibility.

b) $c_{N_i} \to c_{N_i} + \Delta c_{N_i}$. This doesn't affect x, y. However, s_{N_i} is changed by Δc_{N_i}. If $s_{N_i} + \Delta c_{N_i} \ge 0$, the optimality conditions still hold, so the basis is not affected. And there is no change to the objective, since $x_N = 0$. Otherwise, the basis is affected. (x, y, s) is primal feasible but dual infeasible. The primal simplex algorithm can be used to restore the dual feasibility and optimality.

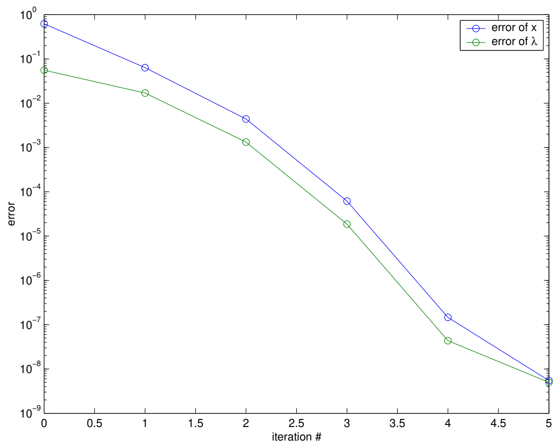
c) $N_i \to N_i + \Delta N_i$. This doesn't affect x, y. However, since $s_N = c_N - N^T y$, s_{N_i} is changed by $-\Delta N_i^T y$. If $s_{N_i} - \Delta N_i^T y \ge 0$, i.e., $s_{N_i} \ge \Delta N_i^T y$, the basic and the objective are not changed, since c, x remain the same. If $s_{N_i} < \Delta N_i^T y$, (x, y, s) is primal feasible but dual infeasible. We can use the primal simplex algorithm to restore the dual feasibility.

d) x_t added with c_t and A_t. Regard x_t as a non-basic variable and let $x_t = 0$. y is not affected. s_N is appended by $s_t = c_t - A_t^T y$. If $s_t \ge 0$, the basis has not been affected and the objective doesn't change. Otherwise $s_t < 0$. Since (x, y, s) now is primal feasible but dual infeasible. The primal simplex algorithm can be used to restore the dual feasibility and optimality.

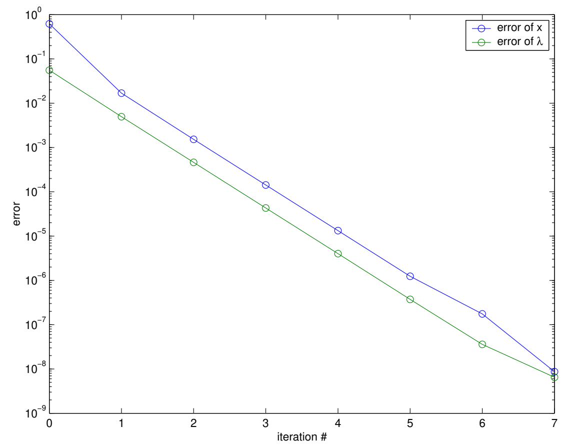
4 Augmented Lagrangian Methods

For less writing, define $f(x) = e^{x_1 x_2 x_3 x_4 x_5}$, $c(x) = (c_1(x), c_2(x), c_3(x))^T$, and

$$\begin{aligned}
c_1(x) &= x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10, \\
c_2(x) &= x_2 x_3 - 5 x_4 x_5, \\
c_3(x) &= x_1^3 + x_2^3 + 1.
\end{aligned}$$



(a) $a_0 = 0.1,\ a_{k+1} = 6a_k$.



(b) $a_k = 0.5$ fixed.

Figure 2: Error for $x^{(k)}$ and $\lambda^{(k)}$ are defined as $\left\|x^{(k)} - x^*\right\|$ and $\left\|\lambda^{(k)} - \lambda^*\right\|$, respectively.

a) Thus the augmented Lagrangian is

$$\begin{aligned}\mathcal{L}_a(x,\lambda) &= f(x) - \lambda^T c(x) + \frac{a}{2}\|c(x)\|^2 \\ &= f(x) - \lambda_1 c_1(x) - \lambda_2 c_2(x) - \lambda_3 c_3(x) + \frac{a}{2}\left[c_1^2(x) + c_2^2(x) + c_3^2(x)\right]. \qquad (3)\end{aligned}$$

And $\nabla_x \mathcal{L}_a(x,\lambda) = \nabla f(x) - \nabla c(x)\lambda + a\nabla c(x)c =$

$$\begin{pmatrix} x_2x_3x_4x_5 f(x) - 2\lambda_1 x_1 - 3\lambda_3 x_1^2 + a\left[2x_1c_1(x) + 3x_1^2c_3(x)\right] \\ x_1x_3x_4x_5 f(x) - 2\lambda_1 x_2 - \lambda_2 x_3 - 3\lambda_3 x_2^2 + a\left[2x_2c_1(x) + x_3c_2(x) + 3x_2^2c_3(x)\right] \\ x_1x_2x_4x_5 f(x) - 2\lambda_1 x_3 - \lambda_2 x_2 + a\left[2x_3c_1(x) + x_2c_2(x)\right] \\ x_1x_2x_3x_5 f(x) - 2\lambda_1 x_4 + 5\lambda_2 x_5 + a\left[2x_4c_1(x) - 5x_5c_2(x)\right] \\ x_1x_2x_3x_4 f(x) - 2\lambda_1 x_5 + 5\lambda_2 x_4 + a\left[2x_5c_1(x) - 5x_4c_2(x)\right] \end{pmatrix}. \qquad (4)$$

b) The program `trunc_newton.m` is reused as the unconstrained sub-iteration which is Hessian-free. Program `aug_lagran.m` controls the augmented Lagrangian method by calling `trunc_newton.m` and updating $\lambda^{(k)}$ as

$$\lambda^{(k+1)} = \lambda^{(k)} - a_k c(x^{(k+1)}),$$

a little different from the normal way: $\lambda^{(k+1)} = \lambda^{(k)} - a_k c(x^{(k)})$. The `objf_4.m` defines the augmented objective and derivative as (3) and (4).

c) Using (finite difference, superlinear convergence)

```
[x,l,f] = aug_lagran('objf_4',[-2;2;2;-1;-1],[0;0;0],[0.1,6],1,1.5,1e-8,5e-8);
```

we get at the 6$^{\text{th}}$ iteration,

$$x^* = \begin{pmatrix} -1.71714357 \\ 1.59570969 \\ 1.82724575 \\ -0.76364308 \\ -0.76364308 \end{pmatrix}, \quad \lambda^* = \begin{pmatrix} -0.04016275 \\ 0.03795778 \\ -0.00522264 \end{pmatrix}.$$

The convergence of x and λ can be seen in Figure 2(a). Here we let a_k increase along the iterations and the converge rate is more than linear. If a_k is fixed at 0.5 (Figure 2(b)), the rate is linear. The previous one uses only 6 iterations, while fixed a_k uses 8.

While $x \to x^*$ and $\lambda \to \lambda^*$, $c(x) \to 0$. The 'augmented part' of $\mathcal{L}_a(x, \lambda)$ tends to be close to 0 if a is fixed. So fixed a results in linear convergence, while $a_k \to \infty$ would speed up the convergence. However, larger a_k makes $\mathcal{L}_a(x, \lambda)$ more difficult to be optimized due to larger condition number.

5 Protein Design is NP-complete

a) The decision form is: *For p positions where position i has n_i amino acid side-chain alternatives, can we select the one side-chain at each position, s.t.,* $E_{\text{total}} = \sum\limits_{i} \sum\limits_{j, j<i} E(i_r, j_s) \leq L$*?*

b) If a PRODES has a "yes" solution, we can calculate E_{total} and verify whether $E_{\text{total}} \leq L$. This can be done with $\frac{p(p-1)}{2}$ additions and at most $\frac{p(p-1)}{2}$ table looking-up (to get $E(i_r, j_s)$). Thus this verification requires poly-time. Note that we may also verify the validity of the given solution by checking that exact one side-chain is selected at each position, in $O(p)$ time. Thus PRODES $\in NP$.

c) Make transformation from SAT to PRODES as shown in the table.

SAT	$\rightarrow$	PRODES ($E_{\text{total}} \leq 0$)
clause i		position i
literal		side-chain
# of literals in clause i		n_i

That is, convert each clause into a position. For each literal in clause i, convert it as one side-chain at position i. The pairwise interaction energy is defined as

$$E(i_r, j_s) = \begin{cases} 1, & \text{if } i_r \text{ and } j_s \text{ are a variable and its negative;} \\ 0, & \text{otherwise.} \end{cases}$$

For example, $E(x_1, \bar{x}_2) = 0$, $E(\bar{x}_1, x_1) = 1$, $E(x_1, x_1) = 0$. Such transformation requires $O\left(\sum\limits_{i=1}^{p} n_i\right)$ time, and the calculation of E requires $O\left(\sum\limits_{i=1}^{p} \sum\limits_{j<i} n_i n_j\right)$ time. So totally we need $O(n^2)$ time, where $n = \sum\limits_{i=1}^{p} n_i$ is the number of literals in SAT problem, or, the size of the problem.

If SAT has a solution, then at least one literal is true in clause i. Select any true literal as the selected side-chain. Since x and $\bar{x}$ can't both be true in the solution, by the definition of E, $E_{\text{total}} = 0$. If there is a solution to PRODES ($E_{\text{total}} \leq 0$), then make those selected literals to be true. Such assignments are consistent, since $E_{\text{total}} \leq 0$ assures each pair in the selection is not a pair of a variable and its negative. Then we know this is also a solution to SAT. (Variables that are not selected could be assigned with any value.)

Thus, in $O(p)$ time, a solution to SAT can be transformed to a solution to PRODES ($E_{\text{total}} \leq 0$), and vice verse. SAT is polynomial-transformable to PRODES. So PRODES $\in NP$-complete.