Modern Optimization Methods for Big Data Problems
Exercises for Lecture 10
(With Solutions)

Dominik Csiba, Jakub Konečný and Peter Richtárik

February 28, 2017

Contents

1 Comparison of NSync and dfSDCA

We want to solve the regularized empirical risk minimization problem, i.e.,

$$\min_{w\in\mathbb{R}^d}\left[P(w) := \frac{1}{n}\sum_{i=1}^{n}\phi_i(\langle x^i, w\rangle) + \frac{\lambda}{2}\|w\|^2,\right] \tag{1}$$

where the example $x^i \in \mathbb{R}^d$ and $\phi_i : \mathbb{R} \to \mathbb{R}$ are β-smooth, i.e., for all $s, t \in \mathbb{R}$ we have

$$\phi_i(s+t) \le \phi_i(s) + \phi_i'(s)t + \frac{\beta}{2}t^2. \tag{2}$$

We define $\mathbf{X}$ as the $d \times n$ matrix defined as the concatenation of the column example vectors

$$\mathbf{X} \overset{\text{def}}{=} (x^1, \ldots, x^n) = \begin{pmatrix} x_1^1 & \cdots & x_1^n \\ \vdots & \ddots & \vdots \\ x_d^1 & \cdots & x_d^n \end{pmatrix}.$$

We will use two different methods for solving (1): NSync with uniform serial sampling and dfSDCA with uniform serial sampling. The two methods are defined as Algorithm 1 and Algorithm 2 below. The main question to solve is the following: when is NSync with uniform serial sampling faster than dfSDCA with uniform serial sampling on problem (1)? We will have to solve a sequence of smaller problems to get the correct answer.

Algorithm 1 NSync with uniform serial sampling

Input: initial iterate $w^0 \in \mathbb{R}^d$, stepsize parameters $v_1, \ldots, v_d > 0$
for $k = 0, \ldots$ **do**
Sample i uniformly at random from $[d]$
Update

$$w_i^{k+1} = w_i^k - \frac{1}{v_i}\nabla_i P(w^k)$$

end for

Algorithm 2 dfSDCA with uniform serial sampling

Input: initial dual variables $\alpha^0 = (\alpha_1^0, \ldots, \alpha_n^0)$, stepsize $\theta > 0$
Initialize: set $w^0 = \frac{1}{\lambda n}\mathbf{X}\alpha^0$
for $k = 0, \ldots$ **do**
Sample i uniformly at random from $[n]$
Update α

$$\alpha_i^{k+1} = \alpha_i^k - n\theta(\phi_i'(\langle x^i, w^k\rangle) + \alpha_i^k)$$

Update w

$$w^{k+1} = w^k - \frac{\theta}{\lambda}(\phi_i'(\langle x^i, w^k\rangle) + \alpha_i^k)x^i$$

end for

Remark: This question was raised by Dominik Csiba last year during the course. This resulted in the paper

D. Csiba and P. Richtárik, Coordinate Descent Face-Off: Primal or Dual?, arXiv:1605.08982, 2016

Feel free to read the above paper for more insights.

2 Computing Stepsizes

2.1 NSync

Find the stepsize parameters v_i in terms of x, β, n, d, λ for the problem (1) using NSync with uniform serial sampling.

Solution:
As we know from theory, the stepsize parameters for uniform serial sampling can be computed as $v_i = \mathbf{M}_{ii}$. But what is $\mathbf{M}$? We need to look into the definition of $C^1(\mathbf{M})$ functions. Recall that $P \in C^1(\mathbf{M})$ if there exists a symmetric positive semindefimite matrix $\mathbf{M}$ such that $\forall w, h \in \mathbb{R}^d$ we have

$$P(w+h) \leq P(w) + (\nabla P(w))^\top h + \frac{1}{2} h^\top \mathbf{M} h. \tag{3}$$

Using (2) we can do the following straightforward computation:

$$\begin{aligned}
P(w+h) &\overset{(1)}{=} \frac{1}{n}\sum_{i=1}^n \phi_i(\langle x^i, w\rangle + \langle x^i, h\rangle) + \frac{\lambda}{2}\|w+h\|^2 \\
&\overset{(2)}{\leq} \frac{1}{n}\sum_{i=1}^n \left[\phi_i(\langle x^i, w\rangle) + \phi_i'(\langle x^i, w\rangle)\cdot\langle x^i, h\rangle + \frac{\beta}{2}\langle x^i, h\rangle^2\right] + \frac{\lambda}{2}\|w\|^2 + \lambda\langle w, h\rangle + \frac{\lambda}{2}\|h\|^2 \\
&= \frac{1}{n}\sum_{i=1}^n \phi_i(\langle x^i, w\rangle) + \frac{\lambda}{2}\|w\|^2 + \left\langle \frac{1}{n}\sum_{i=1}^n \phi_i'(\langle x^i, w\rangle) x^i + \lambda w \ , \ h \right\rangle \\
&\quad + \frac{1}{2} h^\top \left(\frac{\beta}{n}\sum_{i=1}^n x^i (x^i)^\top + \lambda \mathbf{I}\right) h \\
&= P(w) + \langle \nabla P(w), h\rangle + \frac{1}{2} h^\top \mathbf{M} h
\end{aligned}$$

where

$$\mathbf{M} = \frac{\beta}{n}\sum_{i=1}^n x^i (x^i)^\top + \lambda \mathbf{I} = \frac{\beta}{n}\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I}.$$

The same result could be proven using Theorem 10 (Lecture 7) and arguing that adding a regularizing term just adds λ. For the computation of v_i the only thing that matters is the diagonal of $\mathbf{M}$. Hence

$$v_i = \frac{\beta}{n}\|\mathbf{X}_{i:}\|^2 + \lambda, \quad i \in [d]. \tag{4}$$

2.2 dfSDCA

Find the stepsize parameter θ x, β, n, d, λ for the problem (1) using dfSDCA with uniform serial sampling.

Solution:
The stepsize for dfSDCA is defined by

$$\min_{i\in[n]} \frac{p_i n\lambda}{\beta v_i + n\lambda}. \tag{5}$$

In this case we have v_i defined directly through the original problem as

$$v_i = \|x^i\|^2 = \|\mathbf{X}_{:i}\|^2, \quad i \in [n]. \tag{6}$$

Observe that the v_i for dfSDCA are computed on the same matrix as for NSync, but transposed. The stepsize θ is therefore

$$\theta = \min_{i\in[n]} \frac{\lambda}{\beta\|\mathbf{X}_{:i}\|^2 + n\lambda}. \tag{7}$$

3 (A Bound on the) Number of Iterations

3.1 NSync

Find the leading term in the number of iterations needed to get NSync with uniform serial sampling to ϵ-accuracy.

Solution:
The leading term is defined as Ω/λ, where λ is the strong convexity parameter and Ω is defined as

$$\Omega \overset{\text{def}}{=} \max_{i\in[d]} \frac{v_i}{p_i}. \tag{8}$$

We computed the v_i in the last exercise (4). Note that $p_i = 1/d$, because we are in $\mathbb{R}^d$. The leading term is therefore

$$\frac{\Omega}{\lambda} = \frac{\max_{i\in[d]} dv_i}{\lambda} = \frac{d\left(\frac{\beta}{n}\max_{i\in[d]}\|\mathbf{X}_{i:}\|^2 + \lambda\right)}{\lambda} = d + \frac{d\beta\max_{i\in[d]}\|\mathbf{X}_{i:}\|^2}{\lambda n}.$$

3.2 dfSDCA

Find the leading term in the number of iterations needed to get dfSDCA with uniform sampling to ϵ-accuracy.

Solution:
The leading term is defined as

$$\frac{1}{\theta} = \max_{i\in[n]} \left(\frac{1}{p_i} + \frac{\beta v_i}{p_i \lambda n} \right).$$

We already computed v_i before (see (6)) and we know $p_i = 1/n$ (because we are sampling through the examples). Hence

$$\frac{1}{\theta} = \max_{i\in[n]} \left(\frac{1}{p_i} + \frac{\beta v_i}{p_i \lambda n} \right) = n + \frac{n\beta \max_{i\in[n]} \|\mathbf{X}_{:i}\|^2}{\lambda n}.$$

4 Average Cost of 1 Iteration

4.1 NSync

Find the average iteration cost of performing one step of NSync with uniform serial sampling.

Solution:
First, we need to compute the actual update for w^{k+1}. That involves a gradient of P. We can get it by the following computation

$$P(w) = \frac{1}{n} \sum_{i=1}^{n} \phi_i(\langle x^i, w\rangle) + \frac{\lambda}{2}\|w\|^2$$

$$\nabla P(w) = \frac{1}{n} \sum_{i=1}^{n} x^i \phi_i'(\langle x^i, w\rangle) + \lambda w$$

$$\nabla_j P(w) = \frac{1}{n} \sum_{i=1}^{n} x_j^i \phi_i'(\langle x^i, w\rangle) + \lambda w_j$$

One partial derivative therefore includes the product $\mathbf{X}^\top w$, as we need all the values $\langle x^i, w\rangle$. That's the same cost as gradient descent and therefore infeasible. We can use a similar trick as shown in Lecture 7 (pages 46–47). Let us call $\alpha = \mathbf{X}^\top w$, with $\alpha \in \mathbb{R}^n$. An efficient algorithm would therefore look as in Algorithm 3

Algorithm 3 NSync for (1) with uniform serial sampling - efficient

Input: initial iterate $w^0 \in \mathbb{R}^d$, stepsize parameters $v_1, \ldots, v_d > 0$
Initialize: Let $\alpha^0 = \mathbf{X}^\top w^0$
for $k = 0, \ldots$ **do**
  Sample j uniformly at random from $[d]$
  Compute update

$$\Delta = -\frac{1}{v_j}\left(\frac{1}{n}\sum_{i=1}^{n} x_j^i \phi_i'(\alpha_i^k) + \lambda w_j^k\right)$$

  Update w: $\quad w_j^{k+1} = w_j^k + \Delta$
  Update α: $\quad \alpha^{k+1} = \alpha^k + \Delta \mathbf{X}_{j:}^\top$
end for

We can analyse the steps in the iteration:

1. Computing Δ is divided into several substeps:
   (a) Computing $\phi_i'(\alpha_i^k)$ takes $\mathcal{O}(1)$ operations, as we already know α_i^k.
   (b) Multiplying by x_j^i and summing up through i depends on the number of nonzero elements x_j^i, so it takes $\mathcal{O}(\|\mathbf{X}_{j:}\|_0)$ operations.
   (c) Adding λw_j^k takes $\mathcal{O}(1)$ operations as we add up to numbers.
   (d) Multiplying by $-\frac{1}{v_j}$ takes also $\mathcal{O}(1)$ operations, as we already computed v_j.
2. Adding Δ to one coordinate of w takes $\mathcal{O}(1)$ operations.
3. Adding two vectors takes the number of nonzero elements of the vector being added, therefore it takes $\mathcal{O}(\|\mathbf{X}_{j:}\|_0)$ operations

In total the whole iteration cost is $\|\mathbf{X}_{j:}\|_0$ depending on the sampled j. As we are sampling j uniformly at random from $[d]$, the cost of one update on average is $\mathcal{O}(\|\mathbf{X}\|_0/d)$.

4.2 dfSDCA

Find the average iteration cost of performing one step of dfSDCA with uniform serial sampling.

Solution:
We can rewrite the dfSDCA method in similar fashion as NSync:

Algorithm 4 dfSDCA with uniform serial sampling

Input: initial dual variables $\alpha^0 = (\alpha_1^0, \ldots, \alpha_n^0)$, stepsize θ
Initialize: set $w^0 = \frac{1}{\lambda n} x \alpha^0$
for $k = 0, 1, 2, \ldots$ **do**
Sample i uniformly at random from $[n]$
Compute update

$$\Delta = -n\theta(\phi_i'(\langle x^i, w^k \rangle) + \alpha_i^k)$$

Update α: $\alpha_i^{k+1} = \alpha_i^k + \Delta$
Update w: $w^{k+1} = w^k - \frac{\Delta}{\lambda n} x^i$
end for

Note how very similar are the efficient implementations for NSync and dfSDCA. They perform very similar updates, but one of them works on the original matrix and the second one works on the transpose of the matrix. As was mentioned in the lectures, dfSDCA can be reinterpreted as a Dual Coordinate Ascent algorithm, which means applying NSync to the dual (in this case transposed) problem.

Back to the exercise. By a similar process as in the last exercise we can observe,that computing Δ takes $\mathcal{O}(\|\mathbf{X}_{:i}\|_0)$ operations, as the inner product takes the most time to compute. Updating α takes $\mathcal{O}(1)$ operations and updating w takes $\mathcal{O}(\|\mathbf{X}_{:i}\|_0)$ operations as we are again summing up two vectors. The total computational cost is therefore $\mathcal{O}(\|\mathbf{X}_{:i}\|_0)$ depending on which i was sampled. As we are sampling uniformly at random, the average cost per iteration is $\mathcal{O}(\|\mathbf{X}\|_0/n)$.

5 Comparison of Runtime

5.1 Uniform serial sampling

Compare the total runtime cost of NSync with uniform serial sampling and dfSDCA with uniform serial sampling. Assume λ was chosen according to statistical learning theory as $\lambda = \beta/n$.

Solution:
The total runtime of NSync is given by multiplying the average iteration cost by the number of iterations, that is

$$\mathcal{R}_{\text{NSync}} = \mathcal{O}(\|\mathbf{X}\|_0/d) \times \left(d + \frac{d\beta \max_{i\in[d]} \|\mathbf{X}_{i:}\|^2}{\lambda n} \right) = \mathcal{O}(\|\mathbf{X}\|_0) \left(1 + \max_{i\in[d]} \|\mathbf{X}_{i:}\|^2 \right)$$

Similarly for dfSDCA we can compute

$$\mathcal{R}_{\text{dfSDCA}} = \mathcal{O}(\|\mathbf{X}\|_0/n) \times \left(n + \frac{n\beta \max_{i\in[n]} \|\mathbf{X}_{:i}\|^2}{\lambda n} \right) = \mathcal{O}(\|\mathbf{X}\|_0) \left(1 + \max_{i\in[n]} \|\mathbf{X}_{:i}\|^2 \right)$$

We can do a simple side-by-side comparison:

- if $\max_{i\in[d]} \|\mathbf{X}_{i:}\|^2 \ll 1$ and also $\max_{i\in[n]} \|\mathbf{X}_{:i}\|^2 \ll 1$, then the two methods perform equally well,

- otherwise the convergence properties directly depend on the relationship between $\max_{i\in[d]} \|\mathbf{X}_{i:}\|^2$ and $\max_{i\in[n]} \|\mathbf{X}_{:i}\|^2$.
- For a matrix with all elements squared roughly equal to some c, we have $\mathcal{R}_{\text{NSync}} \approx \mathcal{O}(\|\mathbf{X}\|_0)(1+cn)$ and $\mathcal{R}_{\text{dfSDCA}} \approx \mathcal{O}(\|\mathbf{X}\|_0)(1+cd)$, which is close to the intuition that we should compare d vs. n
- **Summary: In general, the pronblem does *not* merely reduce down to d vs. n**

5.2 Importance sampling

What is going to happen if we compare the best serial samplings, i.e., the importance samplings, for NSync and dfSDCA?

Solution:
There are not going to be that many changes. Let us analyse both cases:

NSync: The stepsize parameters v_i stay the same, as we still have a serial sampling. However, the probabilities are going to change. Instead of $p_i = 1/d$ we have

$$p_i = \frac{v_i}{\sum_{j=1}^d v_j} = \frac{\frac{\beta}{n}\|\mathbf{X}_{i:}\|^2 + \lambda}{\sum_j \left(\frac{\beta}{n}\|\mathbf{X}_{j:}\|^2 + \lambda\right)}.$$

The number of iterations changes to

$$\frac{\Omega}{\lambda} = \frac{\max_{i\in[d]} \frac{v_i}{p_i}}{\lambda} = \frac{\sum_{j=1}^d v_j}{\lambda} = \frac{\sum_{j=1}^d \left(\frac{\beta}{n}\|\mathbf{X}_{j:}\|^2 + \lambda\right)}{\lambda}.$$

As for the computational cost of one iteration, the only thing which changes is the probability to sample $i \in [d]$. The average computational cost per iteration is therefore

$$\mathcal{O}\left(\sum_{i=1}^d p_i \|\mathbf{X}_{i:}\|_0\right) = \mathcal{O}\left(\frac{\sum_{i=1}^d \|\mathbf{X}_{i:}\|_0 \left(\frac{\beta}{n}\|\mathbf{X}_{i:}\|^2 + \lambda\right)}{\sum_{j=1}^d \left(\frac{\beta}{n}\|\mathbf{X}_{j:}\|^2 + \lambda\right)}\right)$$

Multiplying these two quantities we get the runtime

$$\begin{aligned}
\mathcal{R}_{\text{NSync-imp}} &= \mathcal{O}\left(\frac{\sum_{i=1}^d \|\mathbf{X}_{i:}\|_0 \left(\frac{\beta}{n}\|\mathbf{X}_{i:}\|^2 + \lambda\right)}{\sum_{j=1}^d \left(\frac{\beta}{n}\|\mathbf{X}_{j:}\|^2 + \lambda\right)}\right) \times \left(\frac{\sum_{j=1}^d \left(\frac{\beta}{n}\|\mathbf{X}_{j:}\|^2 + \lambda\right)}{\lambda}\right) \\
&= \mathcal{O}\left(\frac{\sum_{i=1}^d \|\mathbf{X}_{i:}\|_0 \left(\frac{\beta}{n}\|\mathbf{X}_{i:}\|^2 + \lambda\right)}{\lambda}\right) \\
&= \mathcal{O}\left(\|\mathbf{X}\|_0 + \frac{\frac{\beta}{n}\sum_{i=1}^d \|\mathbf{X}_{i:}\|_0 \|\mathbf{X}_{i:}\|^2}{\lambda}\right)
\end{aligned}$$

and again using the fact that we chose $\lambda = \beta/n$ we get

$$\mathcal{R}_{\text{NSync-imp}} = \mathcal{O}\left(\|\mathbf{X}\|_0 + \sum_{i=1}^{d} \|\mathbf{X}_{i:}\|_0 \|\mathbf{X}_{i:}\|^2\right)$$

dfSDCA: It is a very similar process with dfSDCA. The v_i stay the same, while the probabilities change to

$$p_i = \frac{\beta v_i + n\lambda}{\sum_{j=1}^{n}(\beta v_j + n\lambda)} = \frac{\beta\|\mathbf{X}_{:i}\|^2 + n\lambda}{\sum_{j=1}^{n}(\beta\|\mathbf{X}_{:j}\|^2 + n\lambda)}.$$

The number of iterations changes to

$$\frac{1}{\theta} = \max_{i\in[n]} \frac{\beta v_i \lambda n}{p_i n\lambda} = \frac{\sum_{j=1}^{n}(\beta v_j + n\lambda)}{\lambda n} = \frac{\sum_{j=1}^{n}\left(\beta\|\mathbf{X}_{:j}\|^2 + n\lambda\right)}{\lambda n}.$$

In a very similar fashion the computational cost changes to the expected cost

$$\mathcal{O}\left(\sum_{i=1}^{n} p_i \|\mathbf{X}_{:i}\|_0\right) = \mathcal{O}\left(\frac{\sum_{i=1}^{n}\|\mathbf{X}_{:i}\|_0\left(\beta\|\mathbf{X}_{:i}\|^2 + n\lambda\right)}{\sum_{j=1}^{n}(\beta\|\mathbf{X}_{:j}\|^2 + n\lambda)}\right).$$

It follows that the total runtime cost is

$$\begin{aligned}
\mathcal{R}_{\text{dfSDCA-imp}} &= \mathcal{O}\left(\frac{\sum_{i=1}^{n}\|\mathbf{X}_{:i}\|_0\left(\beta\|\mathbf{X}_{:i}\|^2 + n\lambda\right)}{\sum_{j=1}^{n}(\beta\|\mathbf{X}_{:j}\|^2 + n\lambda)}\right) \times \left(\frac{\sum_{j=1}^{n}\left(\beta\|\mathbf{X}_{:j}\|^2 + n\lambda\right)}{\lambda n}\right) \\
&= \mathcal{O}\left(\frac{\sum_{i=1}^{n}\|\mathbf{X}_{:i}\|_0\left(\beta\|\mathbf{X}_{:i}\|^2 + n\lambda\right)}{\lambda n}\right) \\
&= \mathcal{O}\left(\|\mathbf{X}\|_0 + \frac{\beta\sum_{i=1}^{n}\|\mathbf{X}_{:i}\|_0\|\mathbf{X}_{:i}\|^2}{\lambda n}\right)
\end{aligned}$$

and again using the setup with $\lambda = \beta/n$ we get

$$\mathcal{R}_{\text{dfSDCA-imp}} = \mathcal{O}\left(\|\mathbf{X}\|_0 + \sum_{i=1}^{n} \|\mathbf{X}_{:i}\|_0 \|\mathbf{X}_{:i}\|^2\right)$$

The runtime of importance serial sampling for both NSync and dfSDCA depends on both the sparsity pattern and the squared norms of the examples/features. One can compute both $\mathcal{R}_{\text{NSync-imp}}$ and $\mathcal{R}_{\text{dfSDCA-imp}}$ before starting the algorithm and decide which to use depending on the result of the comparison. One needs to compute the squared norms anyhow for importance sampling, so the computational cost rises just by a factor of 2. There is no immediate connection between d,n and the runtime of NSync and dfSDCA. All depends on the specific structure of the matrix.

6 (*) Which Serial Sampling Leads to Optimal Runtime?

Find the importance sampling minimizing the *total (expected) runtime* of NSync.

Solution:
This task leads to the following optimization problem:

$$p^* \quad \leftarrow \quad \operatorname*{arg\,min}_{p\in\mathbb{R}^d_+ \,:\, \sum_i p_i=1} \quad \left[S(p) \stackrel{\text{def}}{=} \left(\max_{i\in d} \frac{v_i}{p_i}\right) \cdot \frac{\sum_{j=1}^d p_i \|\mathbf{X}_{j:}\|_0}{\lambda} \right] \tag{9}$$

First observe, that the problem is homogeneous in p, i.e., if p is optimal, also cp will be optimal for $c \in \mathbb{R}_+$, as the solution will be the same. Using this argument, we can remove the constraint $\sum_i p_i = 1$. Also, we can remove the λ from the denominator as it does not change the $\arg\min$. Hence we get the simpler but still complicated problem

$$p^* \quad \leftarrow \quad \operatorname*{arg\,min}_{p\in\mathbb{R}^d_+} \quad \left[T(p) \stackrel{\text{def}}{=} \left(\max_{i\in d} \frac{v_i}{p_i}\right) \cdot \sum_{j=1}^d p_i \|\mathbf{X}_{j:}\|_0 \right]. \tag{10}$$

Take an arbitrary $p \in \mathbb{R}^d_+$ and let M_p be the set of maximizers of $\frac{v_i}{p_i}$, i.e.,

$$M_p \stackrel{\text{def}}{=}= \left\{ i \in [d] : \frac{v_i}{p_i} = \max_{j\in d} \frac{v_j}{p_j} \right\}. \tag{11}$$

As $\frac{v_i}{p_i}$ is a continuous function on $p_i \in \mathbb{R}_+$, there exists such a constant $c > 1$ that if we define a new vector $q \in \mathbb{R}^d$ such that

$$q_i = \begin{cases} cp_i & \text{if } i \in M_p \\ p_i & \text{if } i \notin M_p \end{cases} \tag{12}$$

then $M_p = M_q$. Let $k \in M_p$. It follows that

$$
\begin{aligned}
T(q) &\overset{(10)}{=} \max_{i \in d} \left(\frac{v_i}{q_i} \right) \sum_{j=1}^{d} q_j \|\mathbf{X}_{j:}\|_0 \\
&\overset{(11)}{=} \frac{v_k}{q_k} \sum_{j=1}^{d} q_j \|\mathbf{X}_{j:}\|_0 \\
&= \frac{v_k}{q_k} \left(\sum_{j \in M_p} q_j \|\mathbf{X}_{j:}\|_0 + \sum_{i \in ([d] \setminus M_p)} q_i \|\mathbf{X}_{i:}\|_0 \right) \\
&\overset{(12)}{=} \frac{v_k}{cp_k} \left(\sum_{j \in M_p} cp_j \|\mathbf{X}_{j:}\|_0 + \sum_{i \in ([d] \setminus M_p)} p_i \|\mathbf{X}_{i:}\|_0 \right) \\
&\leq \frac{v_k}{p_k} \left(\sum_{j \in M_p} p_j \|\mathbf{X}_{j:}\|_0 + \sum_{i \in ([d] \setminus M_p)} p_i \|\mathbf{X}_{i:}\|_0 \right) \\
&= \frac{v_k}{p_k} \sum_{j=1}^{d} p_j \|\mathbf{X}_{j:}\|_0 \\
&\overset{(11)}{=} \max_{i \in d} \left(\frac{v_i}{p_i} \right) \sum_{j=1}^{d} p_j \|\mathbf{X}_{j:}\|_0 \\
&\overset{(10)}{=} T(p)
\end{aligned}
$$

We can observe that $T(q) \leq T(p)$, while the equality holds if and only if $M_p = [d]$, i.e., if $v_i \sim p_i$ for all $i \in d$ (assuming that $\forall i \in d,\ \|\mathbf{X}_{i:}\|_0 \neq 0$). In other words, if the probabilities are not proportional to v_i, we do not have the optimal probabilities.

To conclude, optimal runtime probabilities are the standard importance sampling probabilities.