LP Relaxations for Reordering Buffer Management

Yuval Rabani - Hebrew University of Jerusalem

based largely on joint papers with Noa Avigdor-Elgrabli, Sungjin Im, Benjamin Moseley
Reordering Buffer Management  [RSW ’02]

- **Input**: a sequence of \( n \) colored items
- **Output**: the same sequence permuted, using a buffer of capacity \( k \)
- **Objective**: minimize the number of color changes in the output sequence

Buffer of size \( k=3 \)
Reordering Buffer Management  [RSW ’02]

input:  

Buffer of size k=3

output:

- **Input**: a sequence of $n$ colored items
- **Output**: the same sequence permuted, using a buffer of capacity $k$
- **Objective**: minimize the number of color changes in the output sequence
Reordering Buffer Management  [RSW ’02]

input:

• Input: a sequence of $n$ colored items

output:

• Output: the same sequence permuted, using a buffer of capacity $k$

• Objective: minimize the number of color changes in the output sequence
Reordering Buffer Management  [RSW ’02]

- **Input**: a sequence of $n$ colored items
- **Output**: the same sequence permuted, using a buffer of capacity $k$
- **Objective**: minimize the number of color changes in the output sequence

**Input**

- Green
- Purple
- Blue
- Red

**Buffer of size $k=3$**

**Output**

- Green

**Cost**: 1
Reordering Buffer Management  [RSW ’02]

- **Input**: a sequence of $n$ colored items
- **Output**: the same sequence permuted, using a buffer of capacity $k$
- **Objective**: minimize the number of color changes in the output sequence

**cost: 2**
Reordering Buffer Management [RSW ’02]

Input: a sequence of \( n \) colored items

Output: the same sequence permuted, using a buffer of capacity \( k \)

Objective: minimize the number of color changes in the output sequence

Cost: 3
Reordering Buffer Management  [RSW ’02]

- **Input:** a sequence of $n$ colored items
- **Output:** the same sequence permuted, using a buffer of capacity $k$
- **Objective:** minimize the number of color changes in the output sequence

Cost: 4
Reordering Buffer Management  [RSW ’02]

- **Input**: a sequence of \( n \) colored items
- **Output**: the same sequence permuted, using a buffer of capacity \( k \)
- **Objective**: minimize the number of color changes in the output sequence

**cost**: 4
Reordering Buffer Management  [RSW ’02]

- **Input:** a sequence of \( n \) colored items
- **Output:** the same sequence permuted, using a buffer of capacity \( k \)
- **Objective:** minimize the number of color changes in the output sequence

**Cost:** 5
Reordering Buffer Management  [RSW ’02]

- **Input**: a sequence of $n$ colored items
- **Output**: the same sequence permuted, using a buffer of capacity $k$
- **Objective**: minimize the number of color changes in the output sequence

**Cost**: 5
Motivation

• Numerous applications:
  - Automotive assembly paint shop
  - Graphics rendering processors, storage systems, network optimization
  - Inverted index compression

• Buffers are pervasive in computer and production systems

• Simple, elegant, natural, non-trivial, and thus appealing model
What’s Known

- **Offline setting:**
  - \( \text{NP}\)-hard [AKM ’10, CMSS ’10]
  - \(O(1)\)-approximation [AR ’13, IM ’14]

- **Online setting:**
  - \(O(\sqrt{\log k})\) (det.) [RSW ’02, EW ’05, AR ’10, ACER ’11]
  - \(O(\log \log k)\) (rand.) [AR ’13]
  - \(\Omega(\sqrt{\log k / \log \log k})\) (det.) \(\Omega(\log \log k)\) (rand.) [ACER ’11]

- **Non-uniform costs (star metric):**
  - offline: \(O(\log \log \log \gamma k)\) approximation [IM ’14, IM ’15]
  - online (det.): \(O(\log k / \log \log k)\) [AR ’10], \(O(\sqrt{\log \gamma k})\) [ACER ’11]
  - online (rand.): \(O(\log^2 \log \gamma k)\) [AIMR ’15] \((\gamma = \text{max costs ratio})\)
Related Work

• Other metrics:
  - line metric: $O(\log |C|)$ (discrete) $O(\log n \log \log n)$ (cont.) [GS ’07]
  - trees: $O(\log k)$ (HSTs) $O(\log D + \log k)$ (gen.) [ERW ’07, ER ’17]
  - general: $O(\log \gamma + \min\{\log k, \log |C|\})$ [KR ’17] \( (D = \text{hop diameter}) \)
  - output = input always costs at most \( 2k-1 \) [EW ’05]

• Other models:
  - block devices: $O(\log \log k)$ (rand.) [ACER ’12]
  - k-client problem: lower bound $\Omega(\log k)$ (det.) [ATUW ’01]

• Other objectives:
  - maximize # color “unchanges” [KP ’04, BL ’07] $O(1)$ approx. (offline)
Linear Programming Relaxation [AR ’10]

\[
\text{minimize } \sum_{i} \sum_{j=i}^{n(i)-2} x_{i,j} \\
\text{s.t. } \sum_{j:j \geq i} x_{i,j} \geq 1 \quad \forall i = 1, 2, \ldots, n \\
\sum_{i:i \leq j} x_{i,j} \leq 1 \quad \forall j = k + 1, k + 2, \ldots, k + n \\
x_{n(i),j} - x_{i,j-1} \geq 0 \quad \forall i = 1, 2, \ldots, n, \forall j \geq n(i) \\
x \geq 0
\]

• \(x_{i,j}\) - the fraction of item \(i\) that is removed at output slot \(j\)
• \(n(i)\) - the next input item of the same color \(c(i)\) as \(i\)
The Fractional Solution

A blue-batch \((l, j)\) = a sequence of consecutive blue items \(l\) that are removed starting at output slot \(j\)

Feasible LP solution:
Fractional packing \(\lambda\) of color batches:
- \(\sum_{l, j: j \geq t-|l|} \lambda_{l, j} = 1\)
- \(\sum_{l, j: i \in l} \lambda_{l, j} = 1\)
- The cost is \(\sum_{l, j} \lambda_{l, j}\)

Feasible RBM solution:
An integral packing \(\lambda\)
Linear Programming Relaxation [AR ’10]

minimize \[ \sum_{(I,j)} x_{I,j} \]

\[ \begin{align*}
\text{s.t.} & & \\
& & \sum_{(I,j): i \in I} x_{I,j} \geq 1 & & \forall i = 1, 2, \ldots, n \\
& & \sum_{(I,t): t \leq j < t + |I|} x_{I,t} \leq 1 & & \forall j = k + 1, k + 2, \ldots, k + n \\
& & x \geq 0
\end{align*} \]

primal \[ x_{I,j} = \text{fraction of } I \text{ that is removed starting at time } j \]

maximize \[ \sum_{i=1}^{n} y_i - \sum_{j=k+1}^{k+n} z_j \]

\[ \begin{align*}
\text{s.t.} & & \\
& & \sum_{i \in I} y_i - \sum_{t = j}^{j + |I| - 1} z_t \leq 1 & & \forall (I, j) \\
& & y, z \geq 0
\end{align*} \]

dual
Consider the following rounding procedure:
- If there is an item in the buffer with LP weight \( \leq \frac{1}{2} \), evict this item’s color
- Otherwise, keep accumulating items in the buffer

The cost increases by a factor of at most 2.
A buffer of size \( 2k \) is sufficient to accommodate the non-evicted items.

\[
\text{OPT}(k) = O(\log k)\cdot\text{OPT}(4k) \quad \text{[EW '05]}
\]

There’s an instance for which \( \text{LP}(k) = \Omega(\log k)\cdot\text{OPT}(4k) \) [Aboud '08]

Integrality gap upper bound: \( O(1) \) [AR '13, IM '14]
(for non-uniform costs: \( \min\{\log k / \log \log k, \log \log \chi k\} \) [AR '10, IM '14])
Simple Online Algorithm

The algorithm:

- increase the “penalty” of each item in the buffer continuously
- if a color’s total penalty reaches 1 — remove this color

**Theorem [AR ’10]:**

The algorithm is $O(\log k / \log \log k)$-competitive (for non-uniform costs).

**Proof:** dual fitting: $z_j = $ penalty per item up to slot $j$

$$y_i - z_{j(i)} = i$’s accumulated penalty / $O(\log k / \log \log k)$$

Let $s$ be the size of the smallest removed color block.

**Theorem:**

The algorithm is $O(\log (k/s))$-competitive.
Primal-Dual Schema (a la [BN ’06])

The algorithm:

while slot $t$ is not full:
- raise $y_i$ for all $i$ not removed completely
- raise $z_j$ for all $j \geq t$

\[
\sigma_{I,j} = \sum_{i \in I} y_i - \sum_{j' = j}^{j+|I|-1} z_{j'}
\]

pseudo primal solution:
\[
\hat{x}_{I,j} = \begin{cases} 
\frac{1}{\ln k} \cdot \sigma_{I,j} & \sigma_{I,j} < 1 \\
\frac{1}{\ln k} \cdot e^{\sigma_{I,j} - 1} & \sigma_{I,j} \geq 1
\end{cases}
\]

primal increment ($J =$ color block in buffer)
\[
\frac{dx_{J,t}}{d\mu} = \max \left\{ \frac{d\hat{x}_{I,j}}{d\mu} : c(I) = c(J) \right\}
\]

- The LP has both covering and packing constraints.
- Raising $x_{I,j}$ consumes space beyond slot $j$.

\[
\begin{align*}
\text{The primal program:} & \\
\text{minimize} & \sum_{(I,j)} x_{I,j} \\
\text{s.t.} & \sum_{(I,j): i \in I} x_{I,j} \geq 1 & \forall i = 1, 2, \ldots, n \\
& \sum_{(I,j') : j' \leq j} x_{I,j'} \leq 1 & \forall j = k + 1, \ldots, k + n \\
& x \geq 0
\end{align*}
\]

\[
\begin{align*}
\text{The dual program:} & \\
\text{maximize} & \sum_{t=1}^{n} y_i - \sum_{j=k'+1}^{k'+n} z_j \\
\text{s.t.} & \sum_{i \in I} y_i - \sum_{j'=j}^{j+|I|-1} z_{j'} \leq 1 & \forall (I,j') \\
y, z \geq 0
\end{align*}
\]
Primal Solution Construction (sort of)

\[ J = \text{The green color block currently in our buffer} \]

Pseudo primal variables:

\[ \hat{x}_{I,j} = \begin{cases} \frac{1}{\ln k} \cdot \sigma_{I,j} & \sigma_{I,j} < 1 \\ \frac{1}{\ln k} \cdot e^{\sigma_{I,j} - 1} & \sigma_{I,j} \geq 1 \end{cases} \]

\[ x_{J,t} \text{ increases @ rate = the maximum pseudo rate} \]

\[ \frac{dx_{J,t}}{d\mu} = \max \left\{ \frac{d\hat{x}_{I,j}}{d\mu} : c(I) = c(J) \right\} \]
**Analysis (bluffing a bit)**

\( \mathcal{B} = \) set of items still present (fractionally) in the buffer

- **Dual increase rate:** \( |\mathcal{B}| - k' \)
- \( |\mathcal{B}| \) might be \( k \), so we compete against a dual that uses \( k' = k - 2k / \ln k \)
- \( \text{OPT}(k') = O(1) \cdot \text{OPT}(k) \) [ERW '09, ACER '12]

**Primal increase rate:** proportional to the scheduled volume, so we need color blocks of size \( \leq O(k - k') = O(k / \log k) \)
Putting It All Together

Online Computation of an LP solution

Input:

Buffer:

- Fractional
  - Small blocks
- Frozen
  - May defrost as large or small
- Integral
  - Large blocks

primal dual schema

dual fitting

O(loglog k)

Online Rounding

similar to offline rounding

Random bits

Output:

O(1)
Open Problems

- Uniform costs:
  - Small constant offline approximation guarantees? PTAS?
  - Limited extra memory?

- Non-uniform costs:
  - $O(\log \log \log \gamma k)$-approx. alg. [IM ’15] (uses knapsack constraints)
  - $O(\log^2 \log \gamma k)$-competitive rand. online alg. [AIMR ’15]

- Other metrics:
  - $o(k)$ guarantees? (independent of other parameters)
  - LP relaxations? LP-based algorithms?
  - Better offline approximation algorithms?
Thank you!
Rounding

Previous target \( t_{q-1} \) \rightarrow current location \( j \) \rightarrow current target \( t_q \):

LP cost reaches \( \delta \cdot q \).
Rounding

previous target

current location

current target: LP cost reaches $\delta \cdot q$
If the difference between our buffer and fractional buffer ($\Delta_j$) is large

$\Rightarrow$ remove $\text{blue}$ and charge $\text{red}$, $\text{yellow}$, $\text{magenta}$, ... $\text{green}$