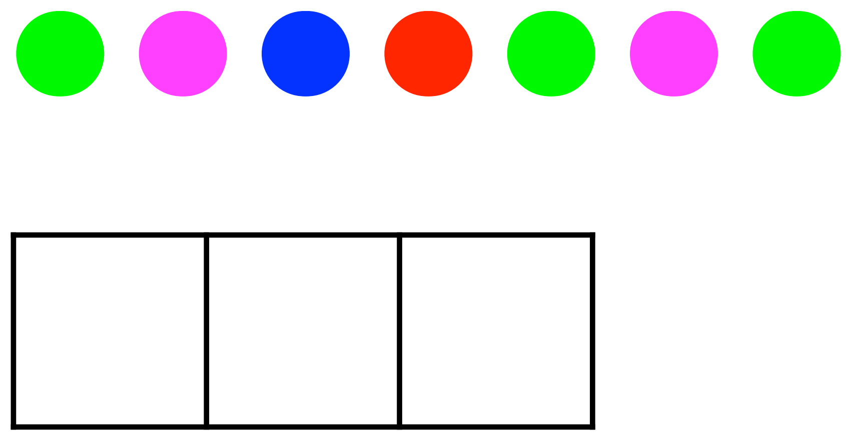
LP Relaxations for Reordering Buffer Management

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:



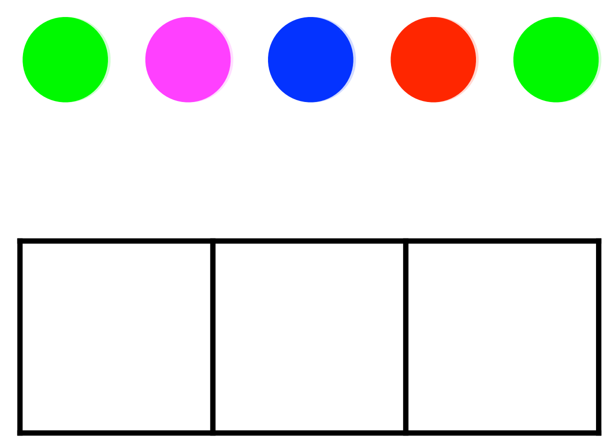
Buffer of size k=3

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:



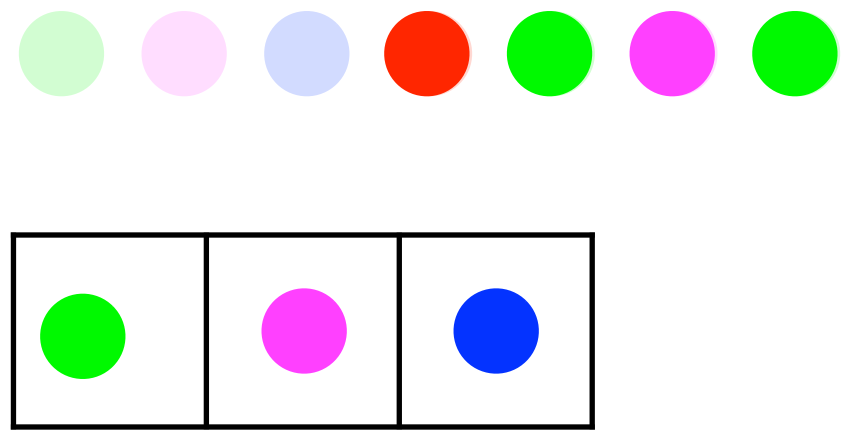
Buffer of size k=3

output:

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

Reordering Buffer Management [RSW '02]

input:

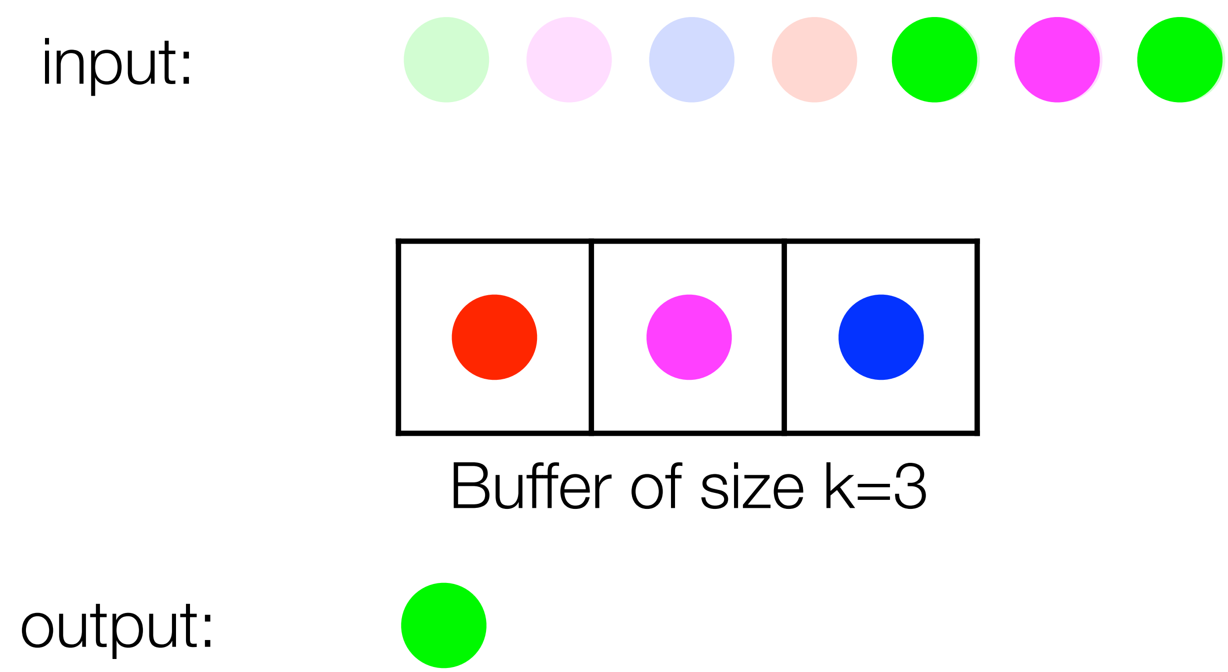


Buffer of size k=3

output:

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

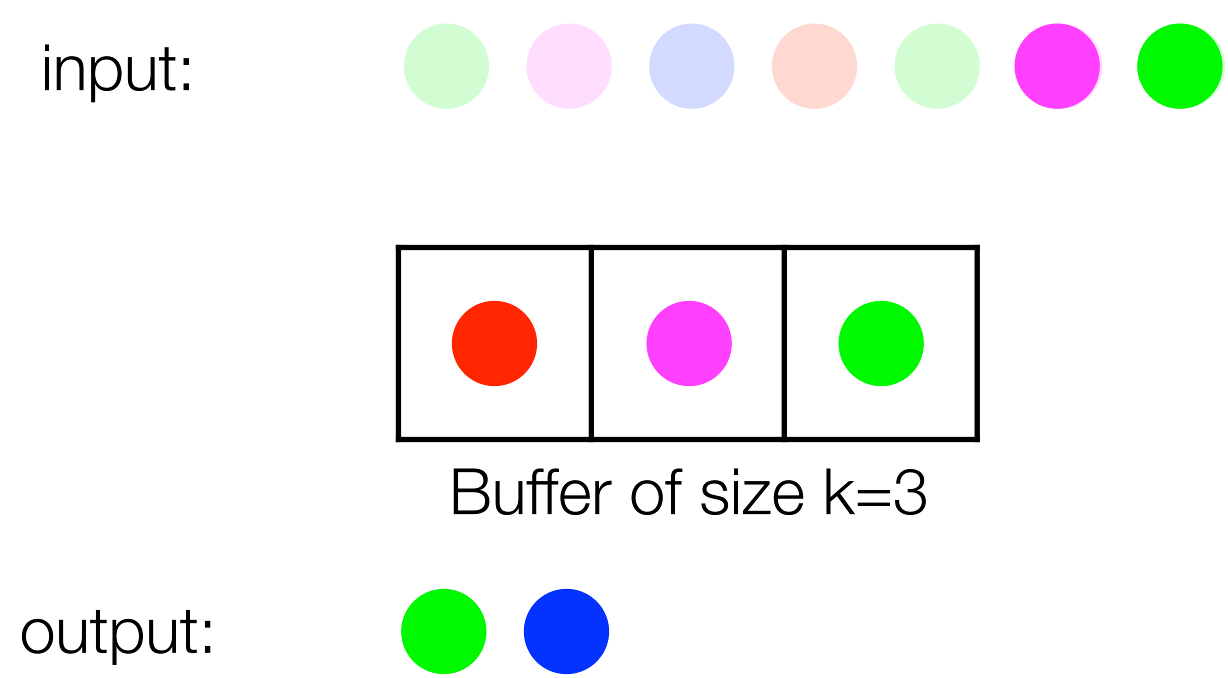
Reordering Buffer Management [RSW '02]



cost: 1

- 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

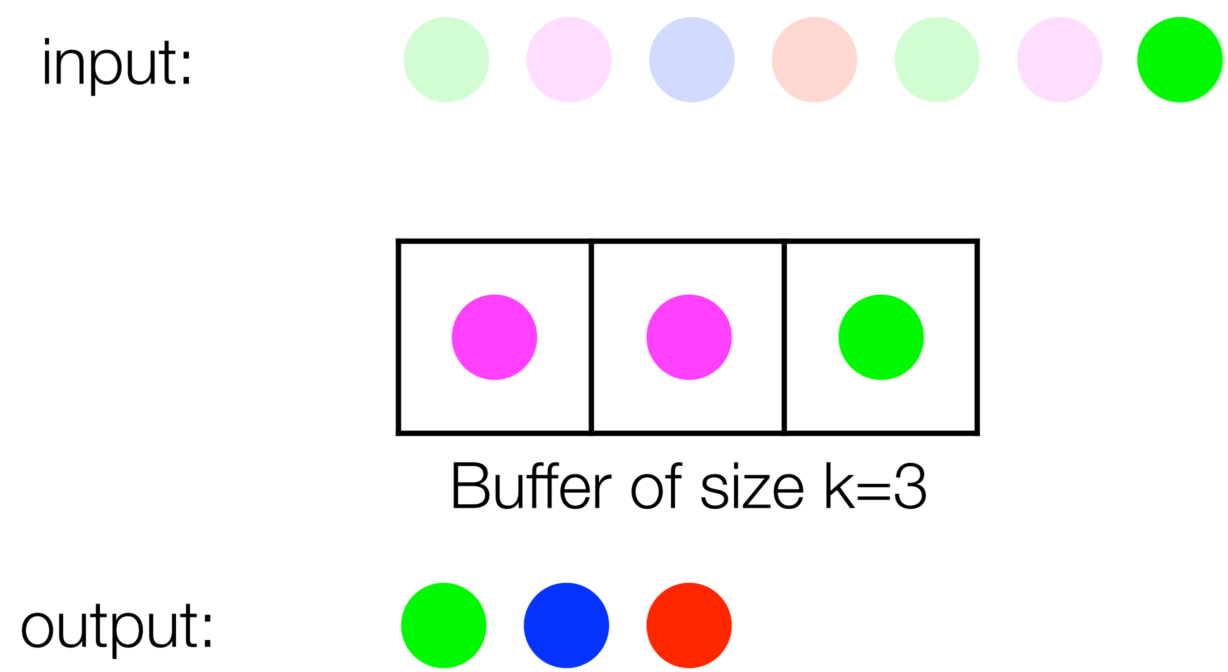
Reordering Buffer Management [RSW ’02]



cost: 2

- 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

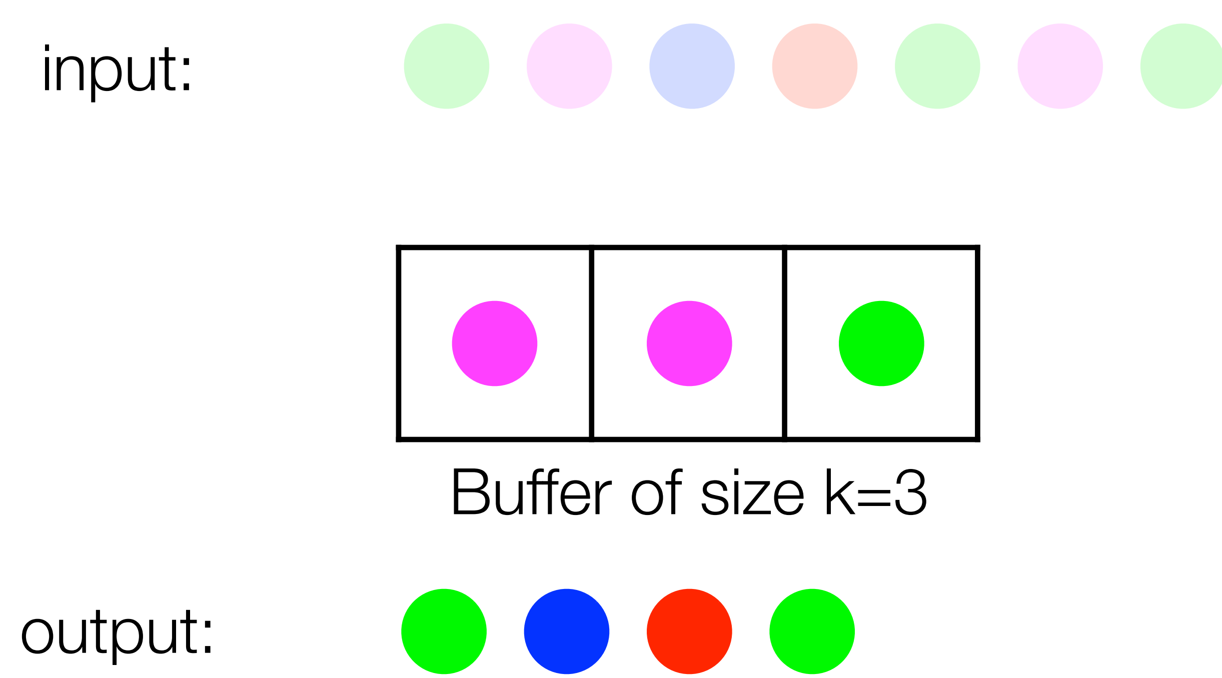
Reordering Buffer Management [RSW ’02]



cost: 3

- 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

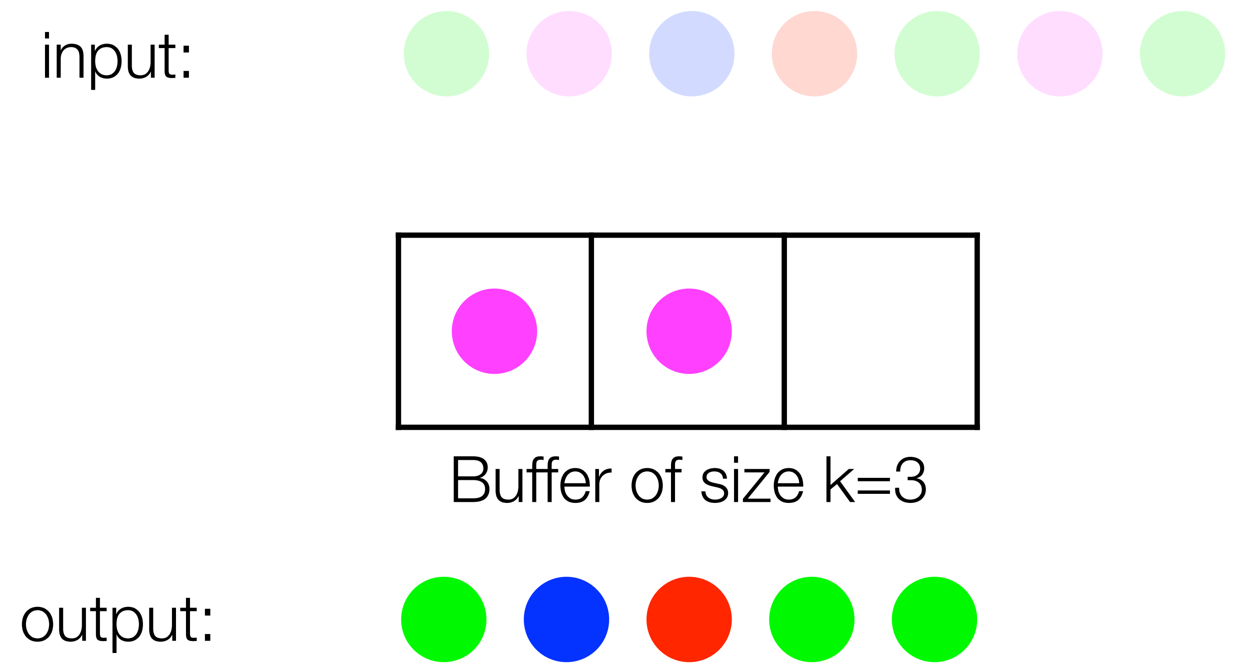
Reordering Buffer Management [RSW ’02]



cost: 4

- 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

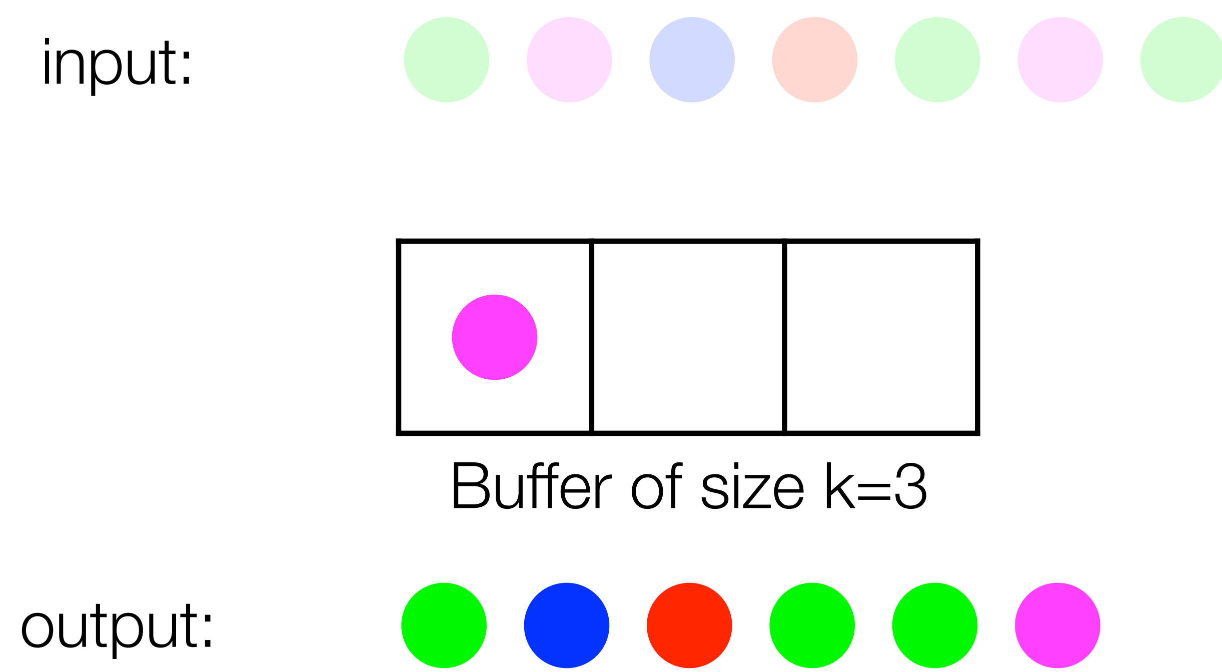
Reordering Buffer Management [RSW '02]



cost: 4

- 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

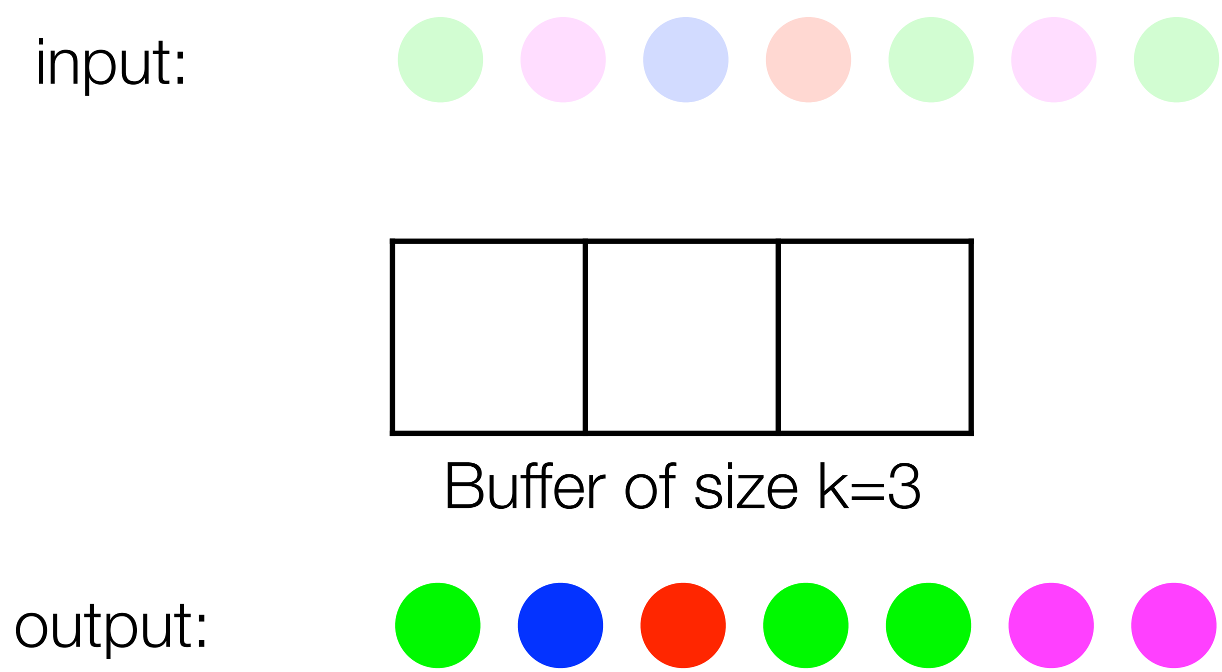
Reordering Buffer Management [RSW '02]



- Input: 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]



cost: 5

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

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:
  - 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]

(γ = 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]
  - output = input always costs at most $2k-1$ [EW ’05]

(D = hop diameter)
(γ = aspect ratio)

- 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]

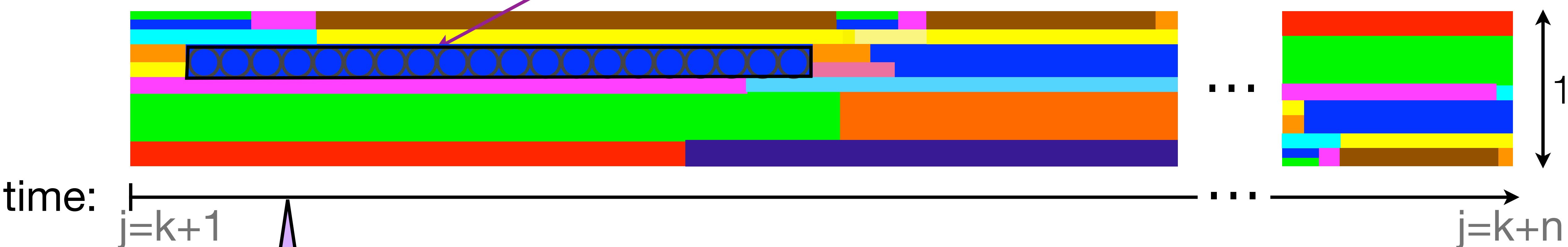
$$\textit{minimize} \quad \sum_{i} \sum_{j=i}^{n(i)-2} x_{i,j}$$

$$\begin{aligned}
\text{s.t.} \quad & \textstyle\sum_{j:j\geq i} x_{i,j} \geq 1 && \forall i = 1, 2, \ldots, n \\
& \textstyle\sum_{i:i\leq j} x_{i,j} \leq 1 && \forall j = k+1, k+2, \ldots, k+n \\
& x_{n(i),j} - x_{i,j-1} \geq 0 && \forall i = 1, 2, \ldots, n, \forall j \geq n(i) \\
& x \geq 0
\end{aligned}$$

- $x_{i,j}$- 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

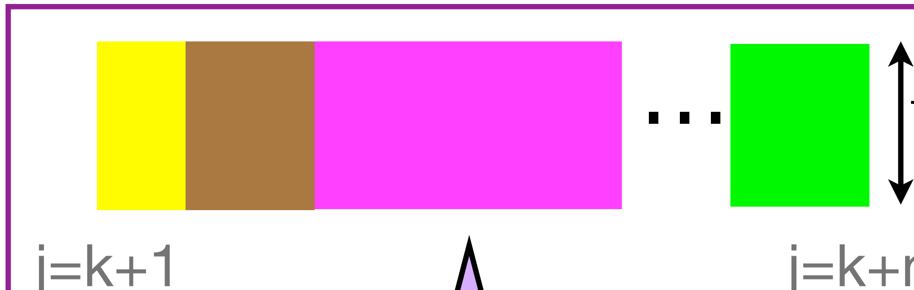
A blue-batch (I,j) =
a sequence of consecutive blue items *I* that are removed starting at output slot *j*



Feasible LP solution:

Fractional packing λ of color batches:

- $\sum_{I,j:\, j \geq t-|I|} \lambda_{I,j} = 1$
- $\sum_{I,j:\, i \in I} \lambda_{I,j} = 1$
- The cost is $\sum_{I,j} \lambda_{I,j}$



Feasible RBM solution:

An **integral** packing λ

Linear Programming Relaxation [AR '10]

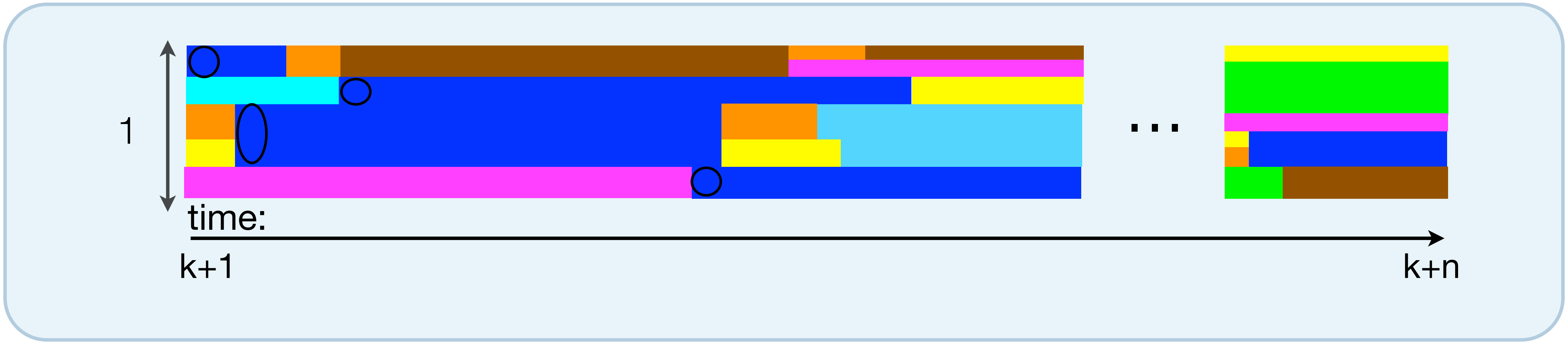
primal

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

$x_{I,j}$ = fraction of I that is removed starting at time j

dual

$$\begin{array}{lll} \text{maximize} & \sum_{i=1}^{n} y_i - \sum_{j=k+1}^{k+n} z_j & \\ \text{s.t.} & \sum_{i\in I} y_i - \sum_{t=j}^{j+|I|-1} z_t \leq 1 & \forall (I,j) \\ & y, z \geq 0 & \end{array}$$



Warmup

- 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

- $OPT(k) = O(\log k) \cdot OPT(4k)$ [EW '05]

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

- Integrality gap upper bound: $O(1)$ [AR '13, IM '14]

  (for non-uniform costs: $\min\{\log k / \log \log k, \log \log \gamma k\}$ [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 LP has both covering and packing constraints.
- Raising $x_{I,j}$ consumes space beyond slot j.

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)

$$\frac{dx_{J,t}}{d\mu} = \max\left\{\frac{d\hat{x}_{I,j}}{d\mu} : \ c(I) = c(J)\right\}$$

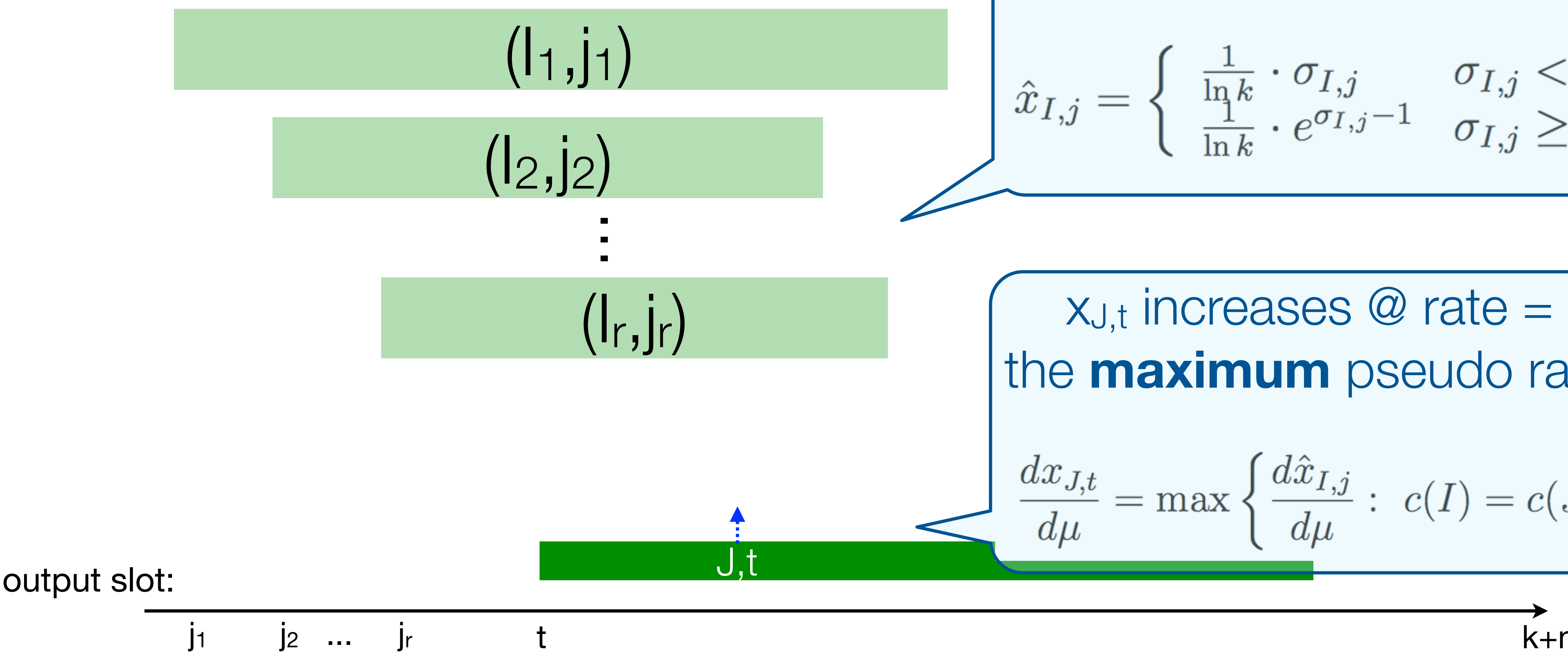
The primal program:

$$\begin{aligned} & minimize \sum_{(I,j)} x_{I,j} \\ & s.t: \sum_{(I,j):\ i \in I} x_{I,j} \geq 1 && \forall i = 1, 2, \ldots, n \\ & \sum_{(I,j'): j' \leq j < j'+|I|} x_{I,j'} \leq 1 && \forall j = k+1, \ldots, k+n \\ & x \geq 0 \end{aligned}$$

The dual program:

$$\begin{aligned} & maximize \sum_{i=1}^{n} y_i - \sum_{j=k'+1}^{k'+n} z_j \\ & s.t: \\ & \sum_{i \in I} y_i - \sum_{j'=j}^{j+|I|-1} z_{j'} \leq 1 && \forall (I,j) \\ & y, z \geq 0 \end{aligned}$$

Primal Solution Construction (sort of)



Pseudo primal variables:

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

$x_{J,t}$ increases @ rate = the **maximum** pseudo rate

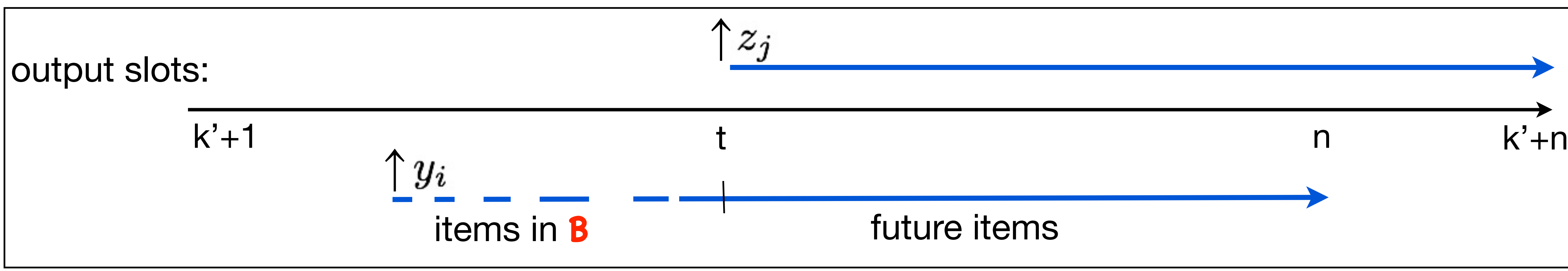
$$\frac{dx_{J,t}}{d\mu} = \max \left\{ \frac{d\hat{x}_{I,j}}{d\mu} \; : \; c(I) = c(J) \right\}$$

J = The green color block currently in our buffer

Analysis (bluffing a bit)

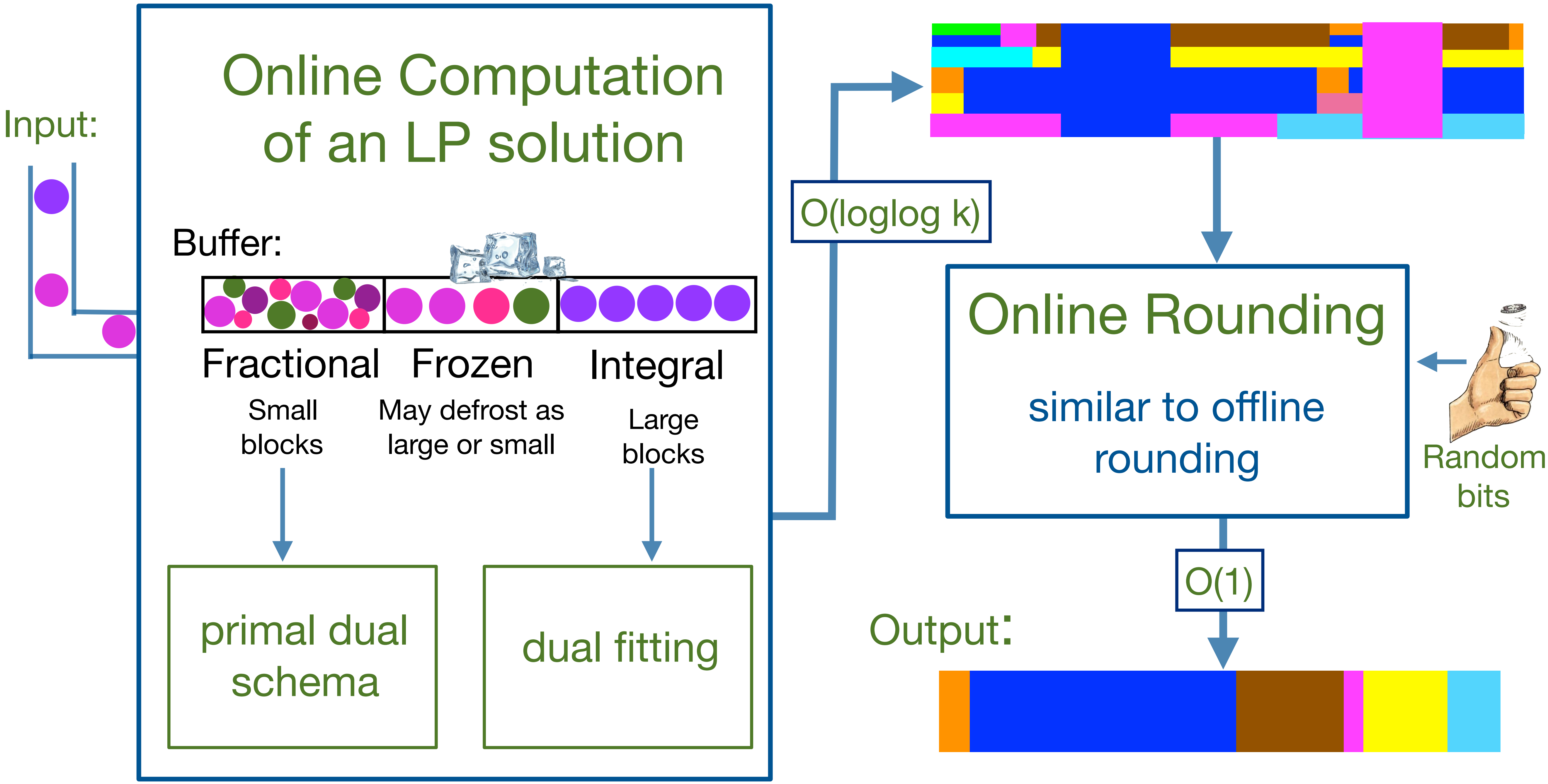
B = set of items still present (fractionally) in the buffer

- **Dual increase rate:** |B| - k’
- |B| might be k, so we compete against a dual that uses k’ = k - 2k / ln k
- OPT(k’) = O(1)·OPT(k) [ERW ’09, ACER ’12]



- **Primal increase rate:** proportional to the scheduled volume
  so we need color blocks of size ≤ O(k - k’) = O(k / log k)

Putting It All Together

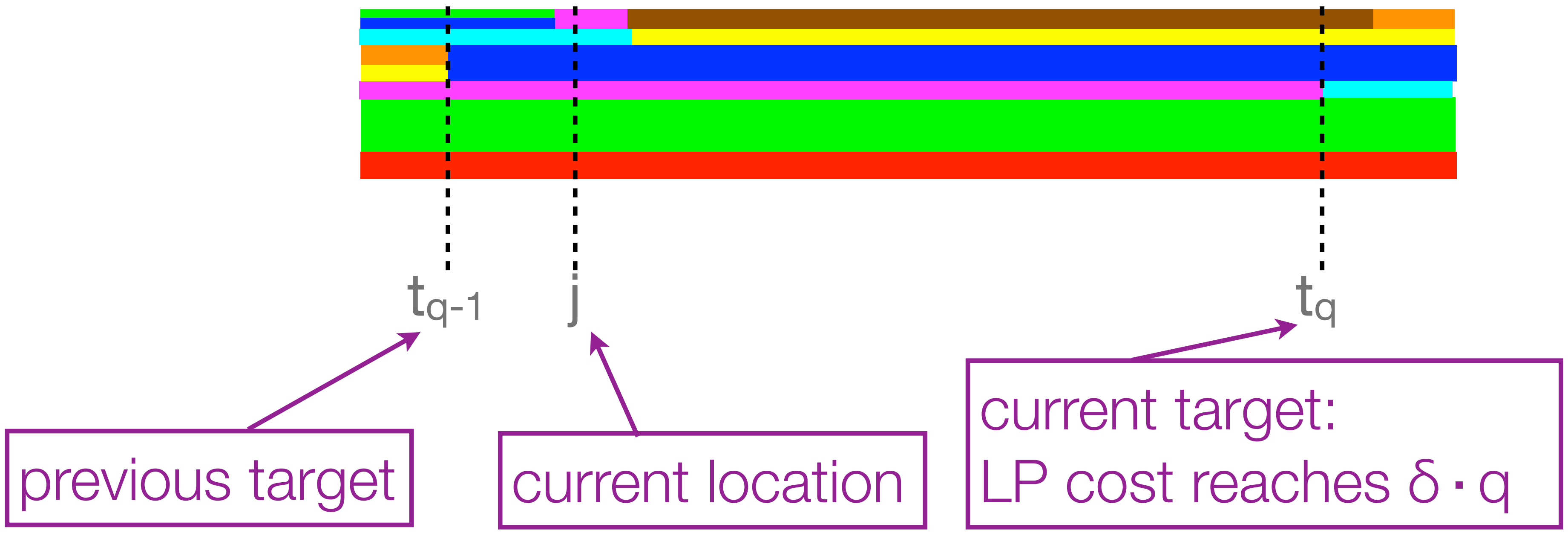


Open Problems

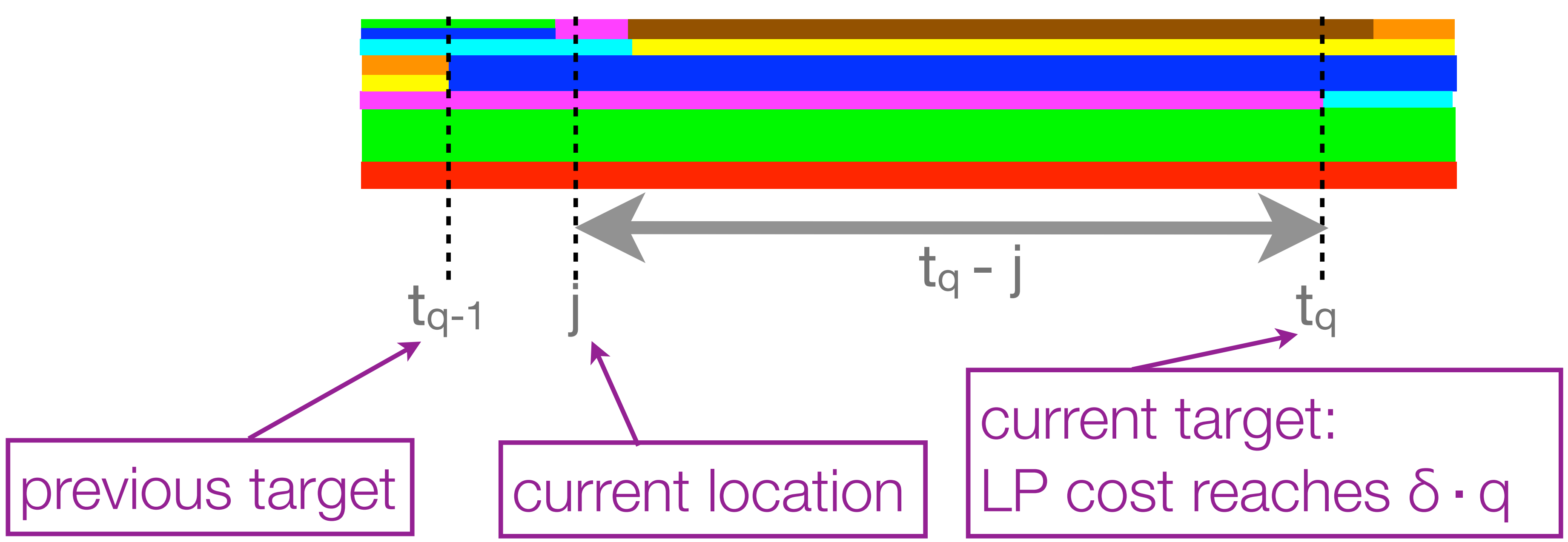
- Uniform costs:
  - Small constant offline approximation guarantees? PTAS?
  - Limited extra memory?
- Non-uniform costs:
  - $O(\log \log \log \gamma k)$-approx. alg. [IM ’15] (uses knapsack constraints)
  - $O(\log^2 \log \gamma k)$-competitive rand. online alg. [AIMR ’15]
- Other metrics:
  - $o(k)$ guarantees? (independent of other parameters)
  - LP relaxations? LP-based algorithms?
  - Better offline approximation algorithms?

Thank you!

Rounding


t_{q-1}
j
t_q
previous target
current location
current target:
LP cost reaches $\delta \cdot q$

Rounding



t_{q-1}
j
$t_q - j$
t_q
previous target
current location
current target:
LP cost reaches δ · q

Rounding

If the difference between our buffer and fractional buffer (Δ_j) is large