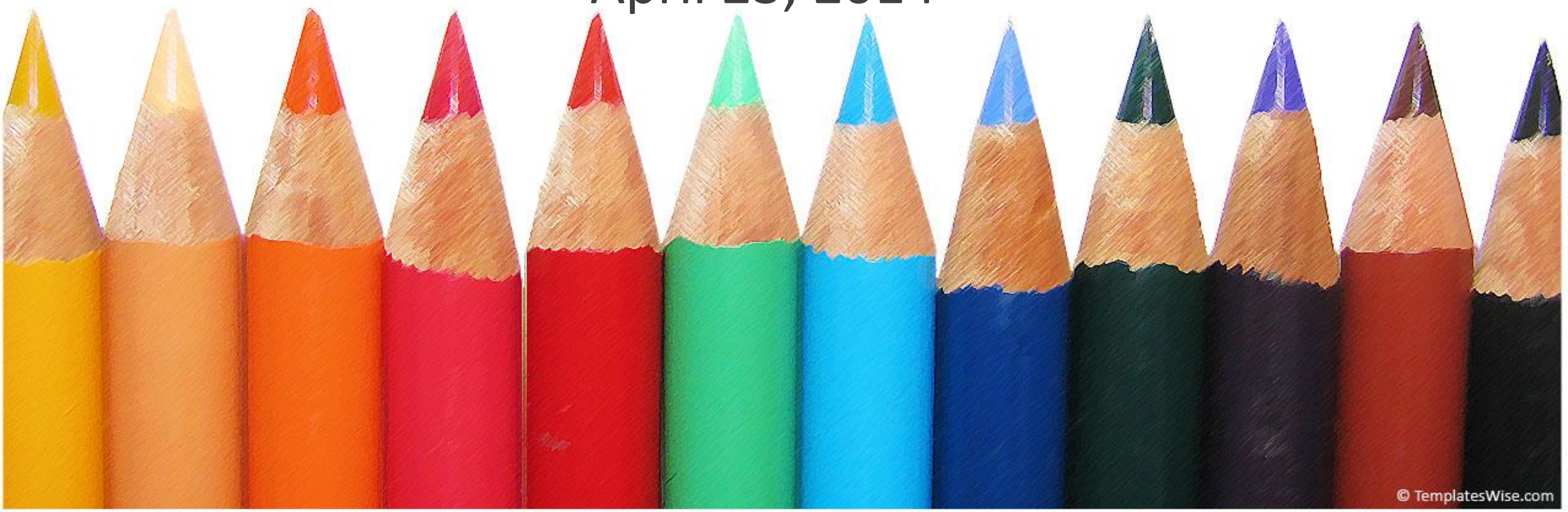
Reordering Buffer Management

Xiaoxi Zhang

April 23, 2014



Layout

- Problem Definition
- Motivation
- Related Work
- Algorithm:

Almost Tight Bounds for Reordering Buffer Management,

Anna Adamaszek and Artur Czumaj, STOC 2011

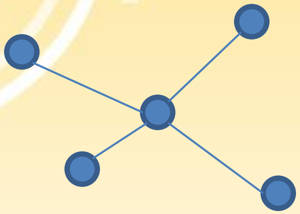
Layout

- Problem Definition
- Motivation
- Related Work
- Algorithm:

Almost Tight Bounds for Reordering Buffer Management,

Anna Adamaszek and Artur Czumaj, STOC 2011

Problem Definition



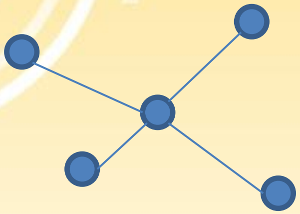
--Based on Star Metric

Input:
A sequence θ of n colored items;
Color set C
Buffer size k;
Cost metric w_c (the cost for switching **to** a color $c \in C$)

Output:
A permutation sequence σ of the input sequence θ

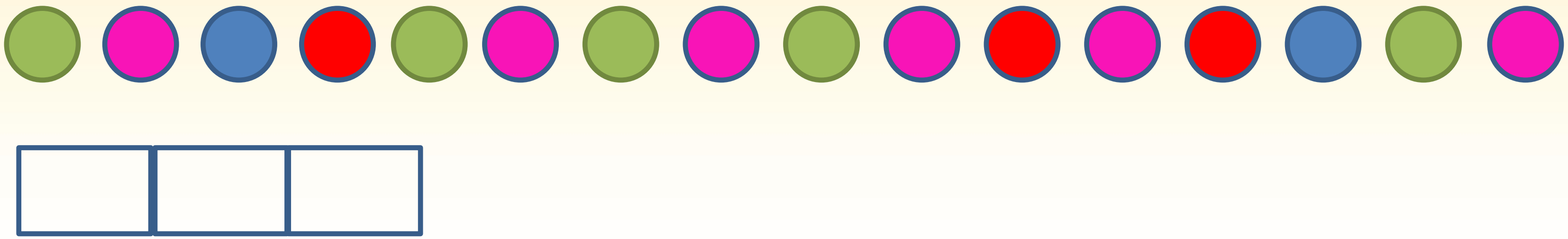
Objective:
Minimize the total cost switching cost

Problem Definition



--Based on Uniform Metric

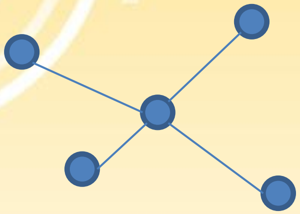
θ



Cost: 0

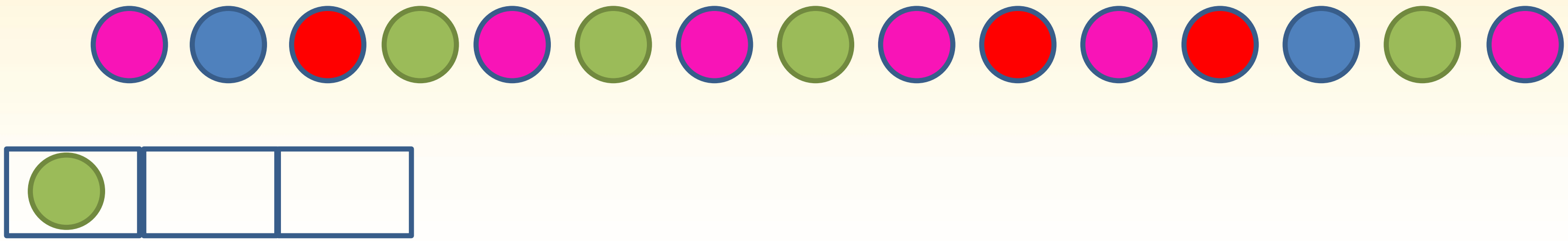
w_g w_p w_b w_r

Problem Definition



--Based on Uniform Metric

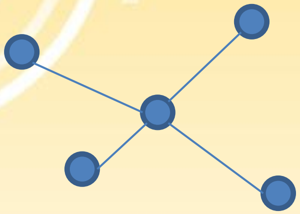
θ



Cost: 0

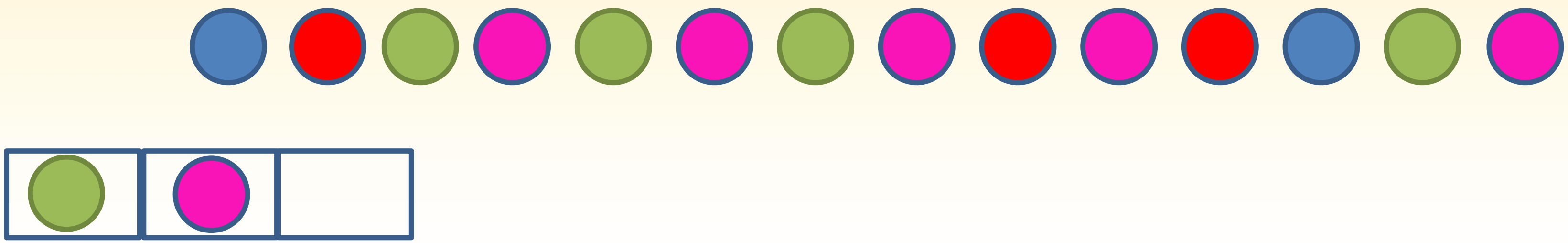
w_g w_p w_b w_r

Problem Definition



--Based on Uniform Metric

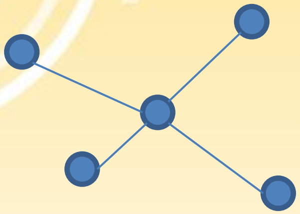
θ



Cost: 0

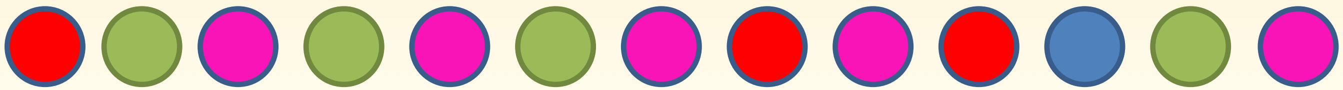
w_g w_p w_b w_r

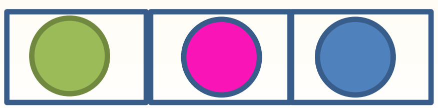
Problem Definition



--Based on Uniform Metric

θ

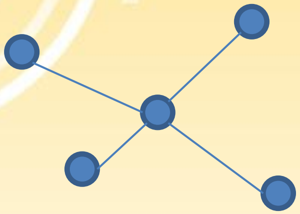




Cost: 0

w_g w_p w_b w_r

Problem Definition



--Based on Uniform Metric

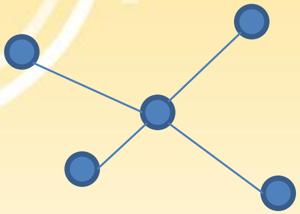
θ

σ

Cost: 1

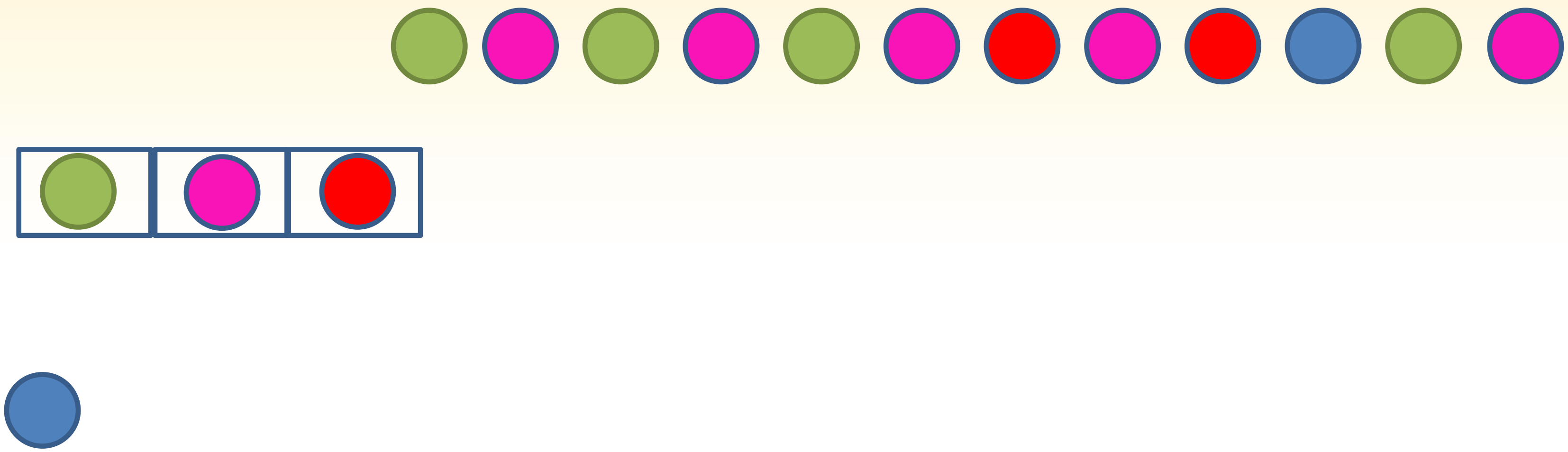
w_g w_p w_b w_r

Problem Definition



--Based on Uniform Metric

θ

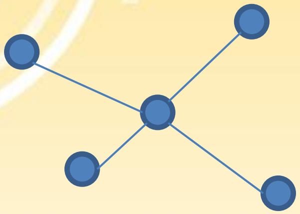


σ

Cost: 1

w_g w_p w_b w_r

Problem Definition



--Based on Uniform Metric

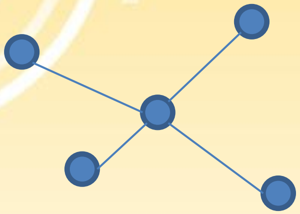
θ

σ

Cost: 2

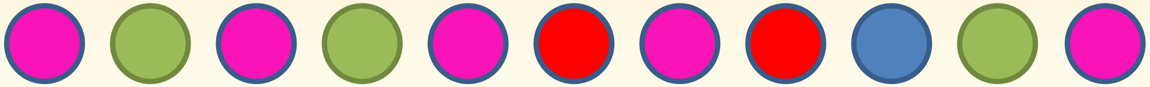
w_g w_p w_b w_r

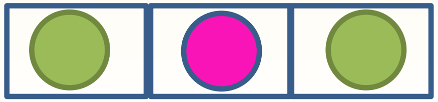
Problem Definition



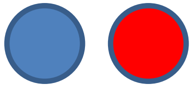
--Based on Uniform Metric

θ





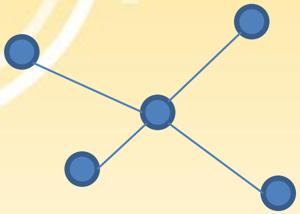
σ



Cost: 2

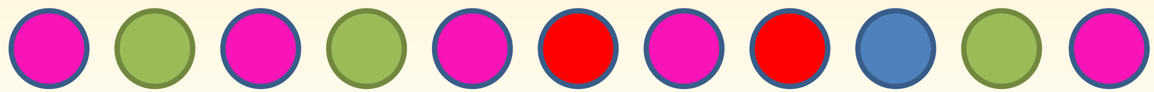
w_g w_p w_b w_r

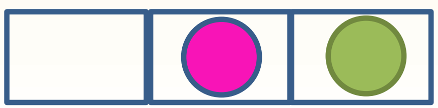
Problem Definition



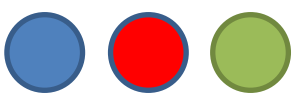
--Based on Uniform Metric

θ





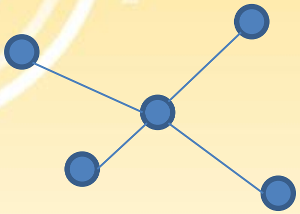
σ



Cost: 3

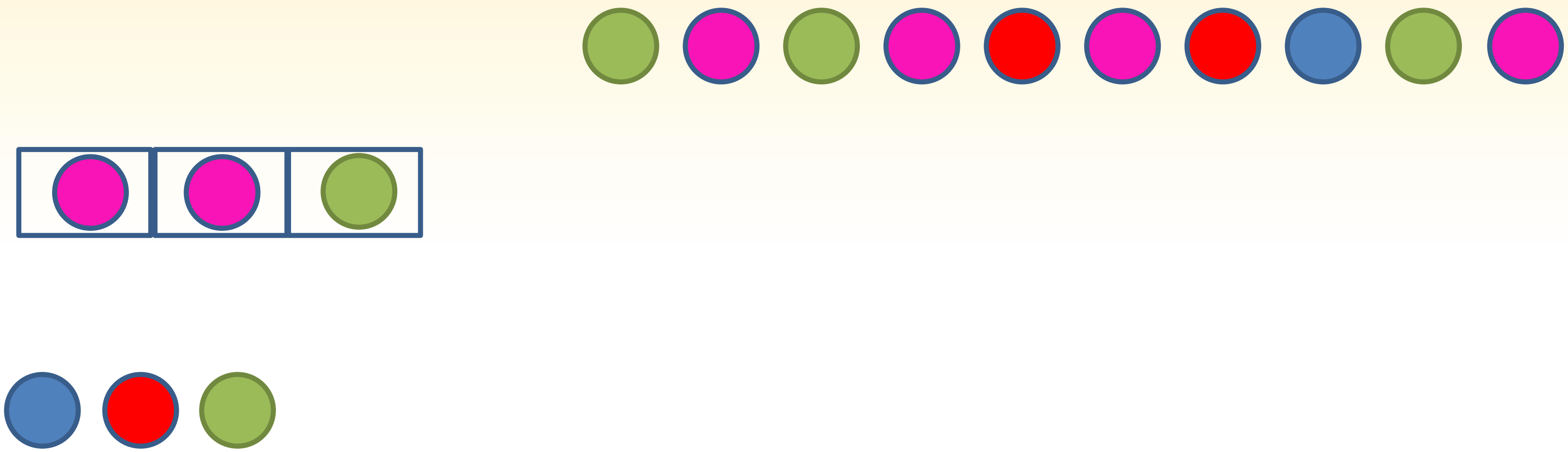
w_g w_p w_b w_r

Problem Definition



--Based on Uniform Metric

θ

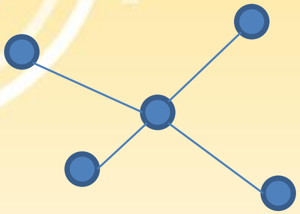


σ

Cost: 3

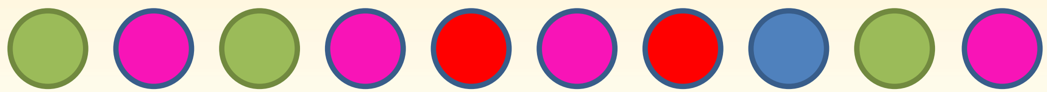
w_g w_p w_b w_r

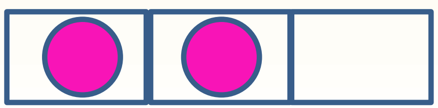
Problem Definition



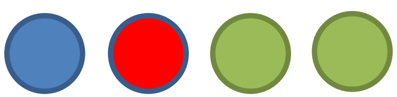
--Based on Uniform Metric

θ





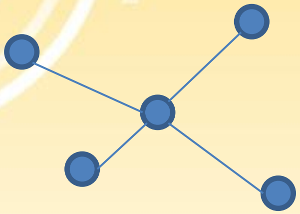
σ



Cost: 3

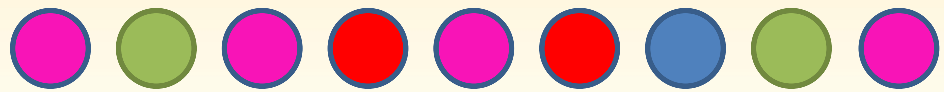
w_g w_p w_b w_r

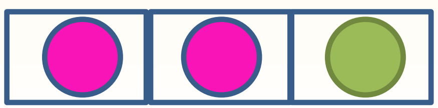
Problem Definition



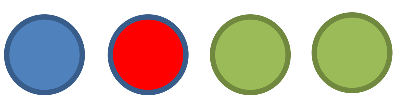
--Based on Uniform Metric

θ





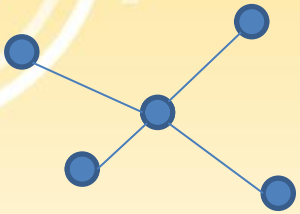
σ



Cost: 3

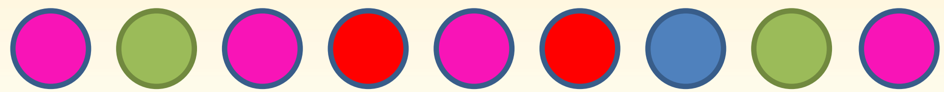
w_g w_p w_b w_r

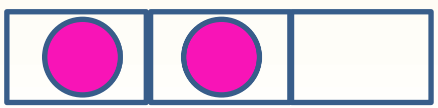
Problem Definition



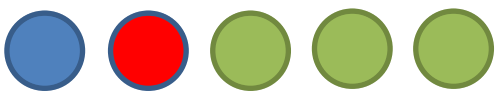
--Based on Uniform Metric

θ





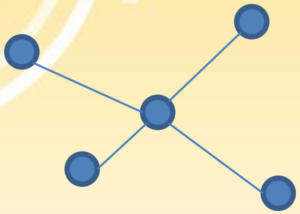
σ



Cost: 3

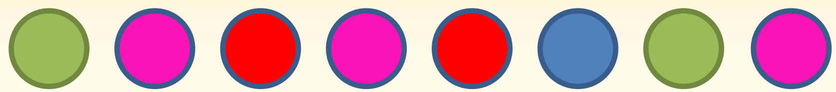
w_g w_p w_b w_r

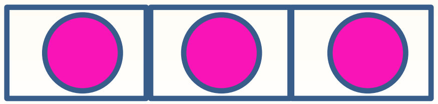
Problem Definition



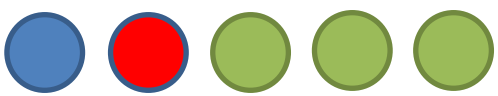
--Based on Uniform Metric

θ





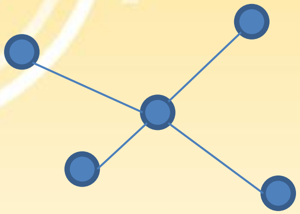
σ



Cost: 3

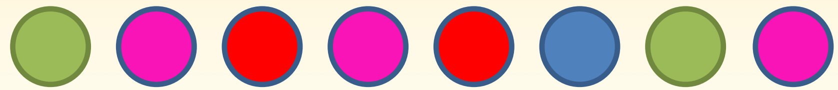
w_g w_p w_b w_r

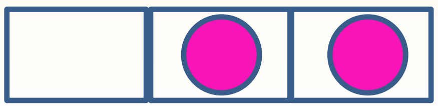
Problem Definition



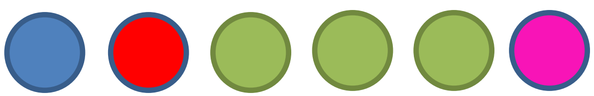
--Based on Uniform Metric

θ





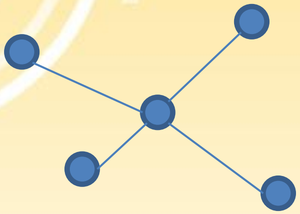
σ



Cost: 4

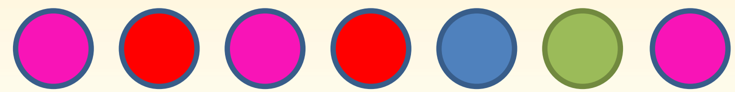
w_g w_p w_b w_r

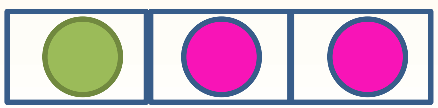
Problem Definition



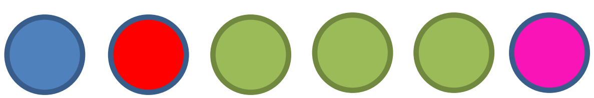
--Based on Uniform Metric

θ





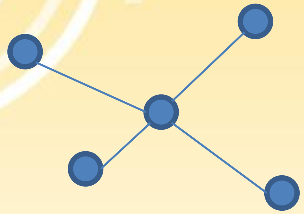
σ



Cost: 4

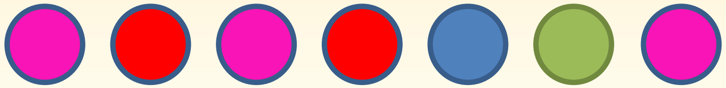
w_g w_p w_b w_r

Problem Definition

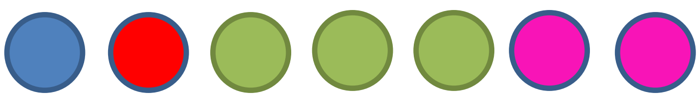


--Based on Uniform Metric

θ



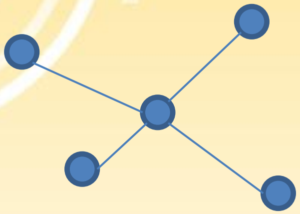
σ



Cost: 4

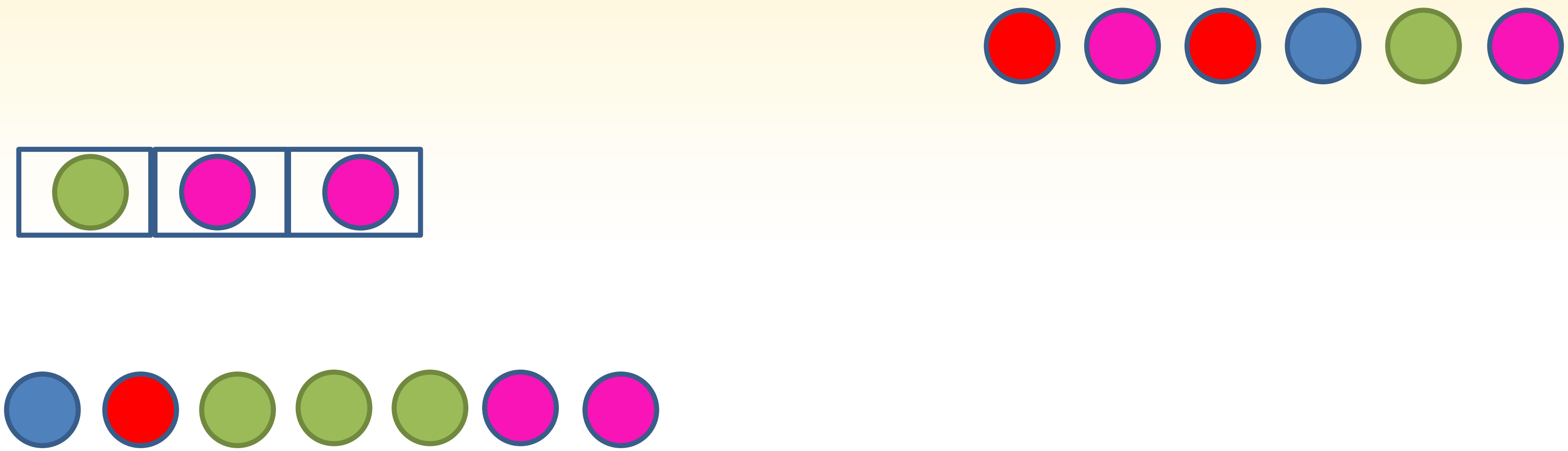
w_g w_p w_b w_r

Problem Definition



--Based on Uniform Metric

θ

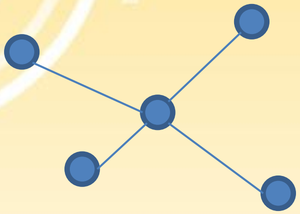


σ

Cost: 4

w_g w_p w_b w_r

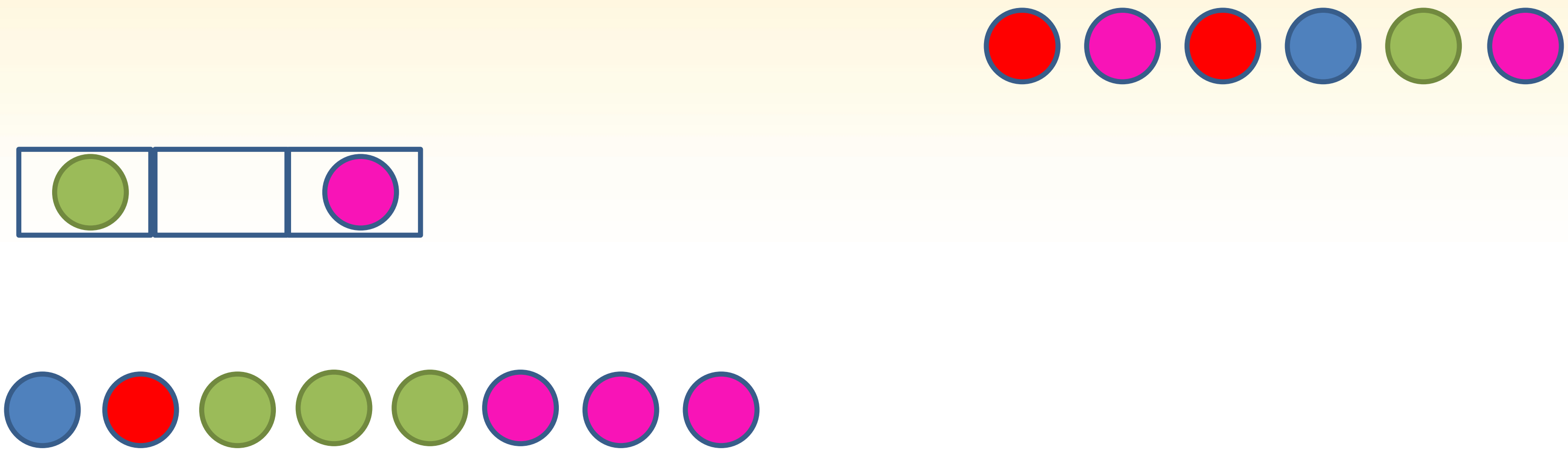
Problem Definition



--Based on Uniform Metric

θ

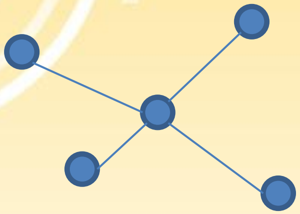
σ



Cost: 4

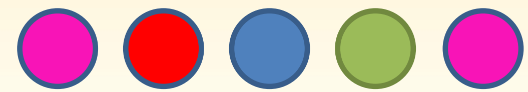
w_g w_p w_b w_r

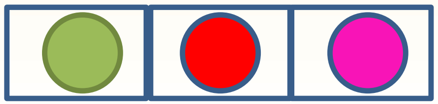
Problem Definition



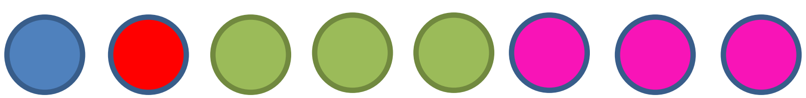
--Based on Uniform Metric

θ





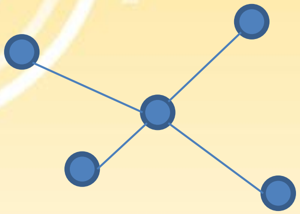
σ



Cost: 4

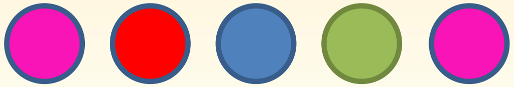
w_g w_p w_b w_r

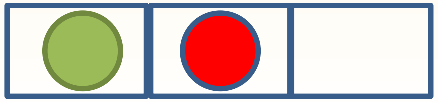
Problem Definition



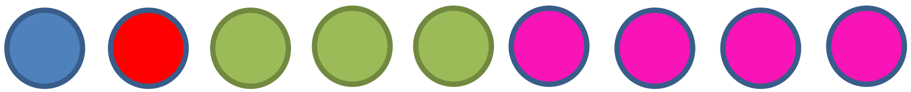
--Based on Uniform Metric

θ





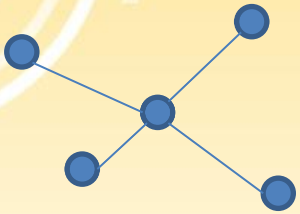
σ



Cost: 4

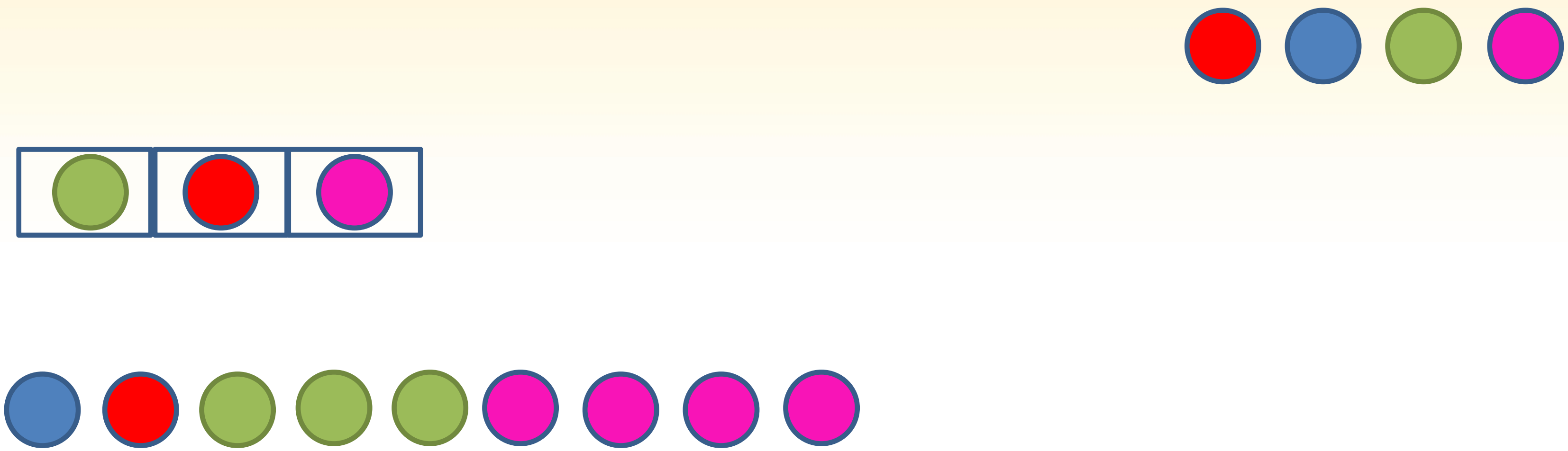
w_g w_p w_b w_r

Problem Definition



--Based on Uniform Metric

θ

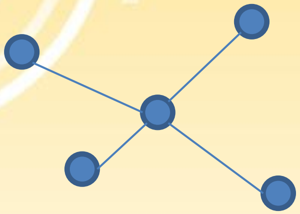


σ

Cost: 4

w_g w_p w_b w_r

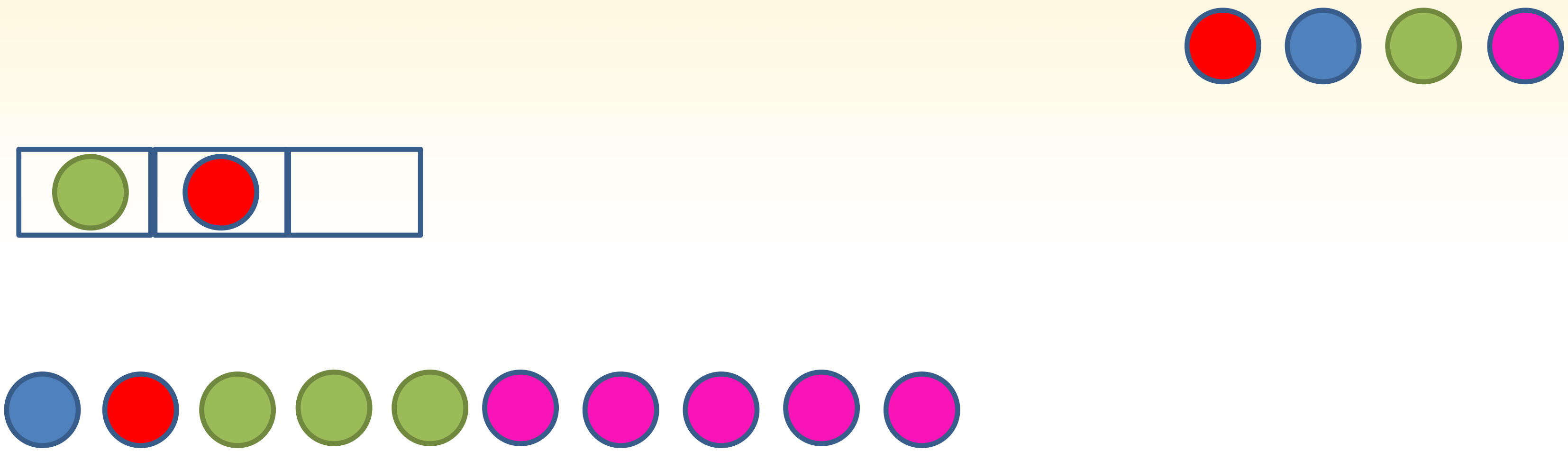
Problem Definition



--Based on Uniform Metric

θ

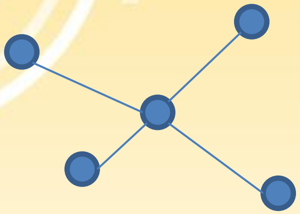
σ



Cost: 4

w_g w_p w_b w_r

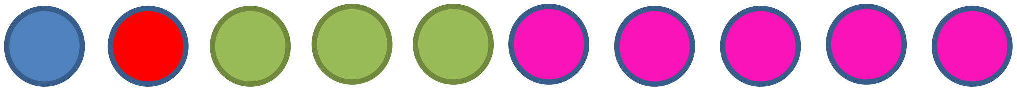
Problem Definition



--Based on Uniform Metric

θ

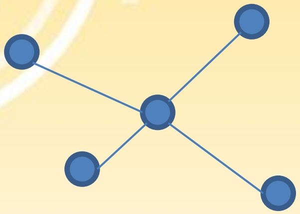
σ



Cost: 4

w_g w_p w_b w_r

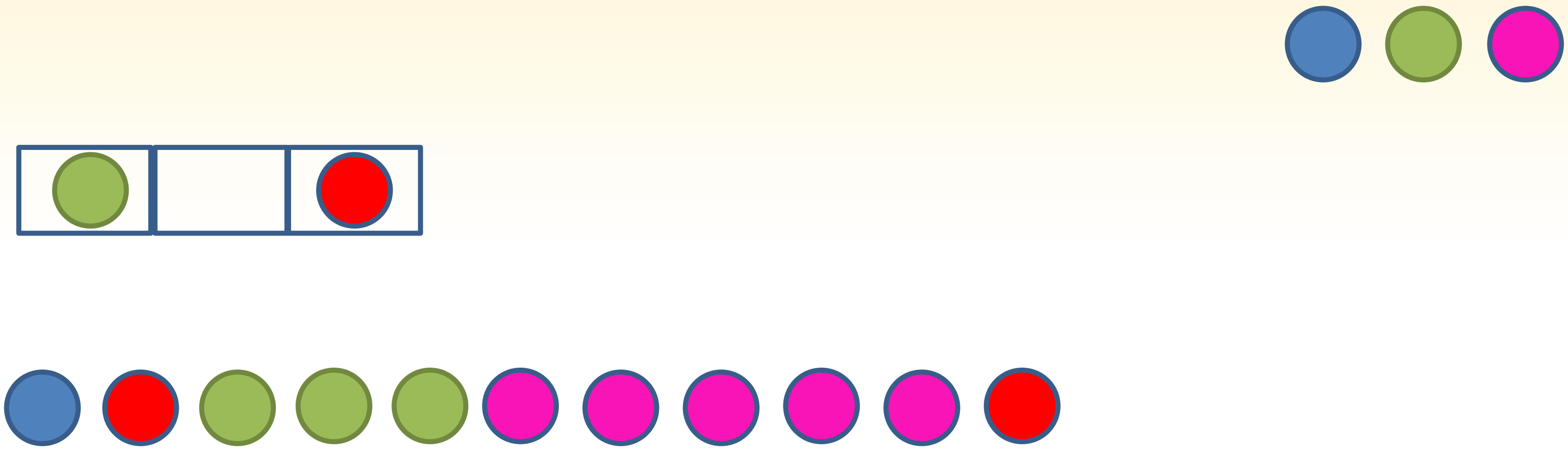
Problem Definition



--Based on Uniform Metric

θ

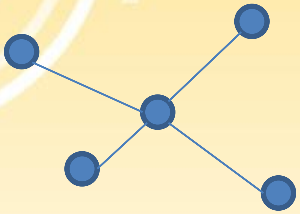
σ



Cost: 5

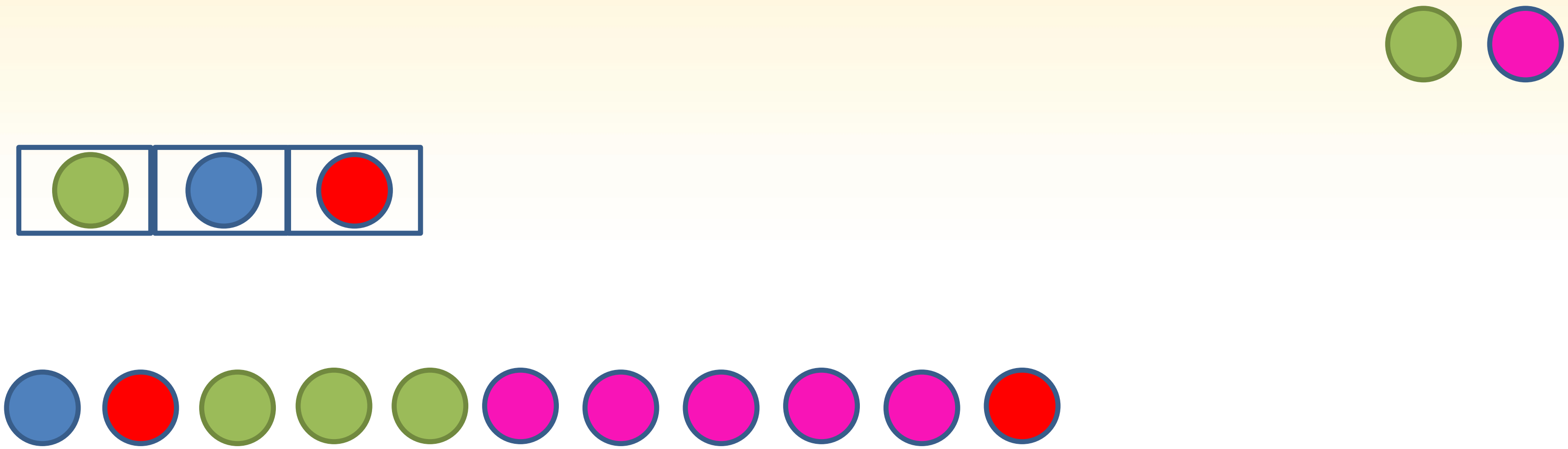
w_g w_p w_b w_r

Problem Definition



--Based on Uniform Metric

θ



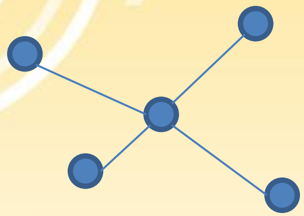
σ

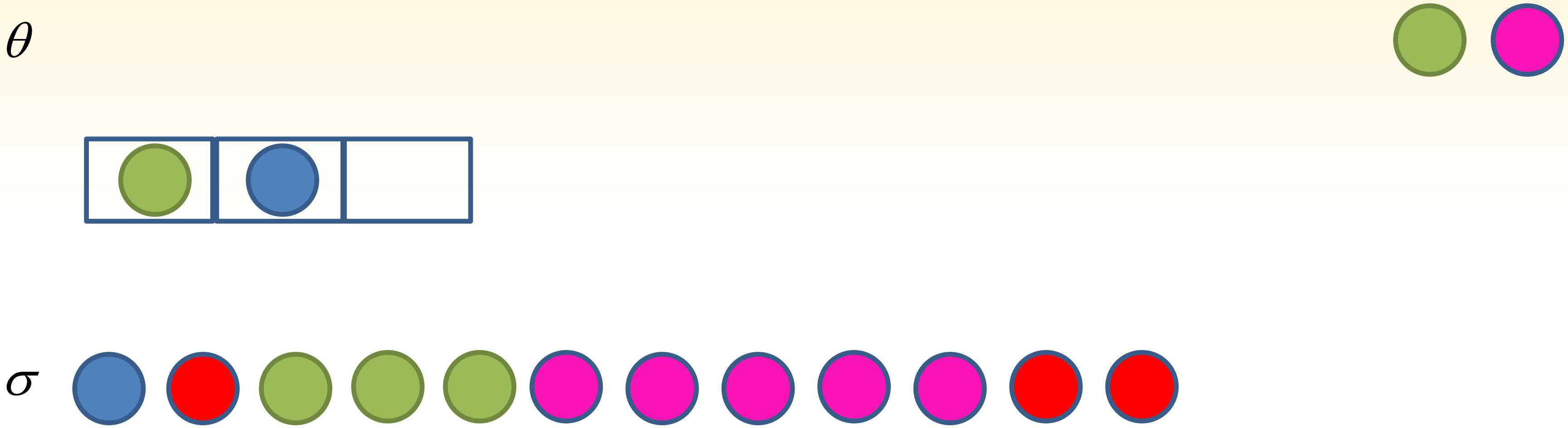
Cost: 5

w_g w_p w_b w_r

Problem Definition

--Based on Uniform Metric

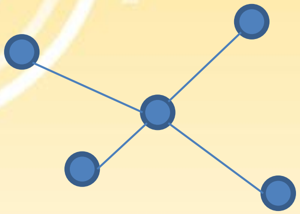




Cost: 5

w_g w_p w_b w_r

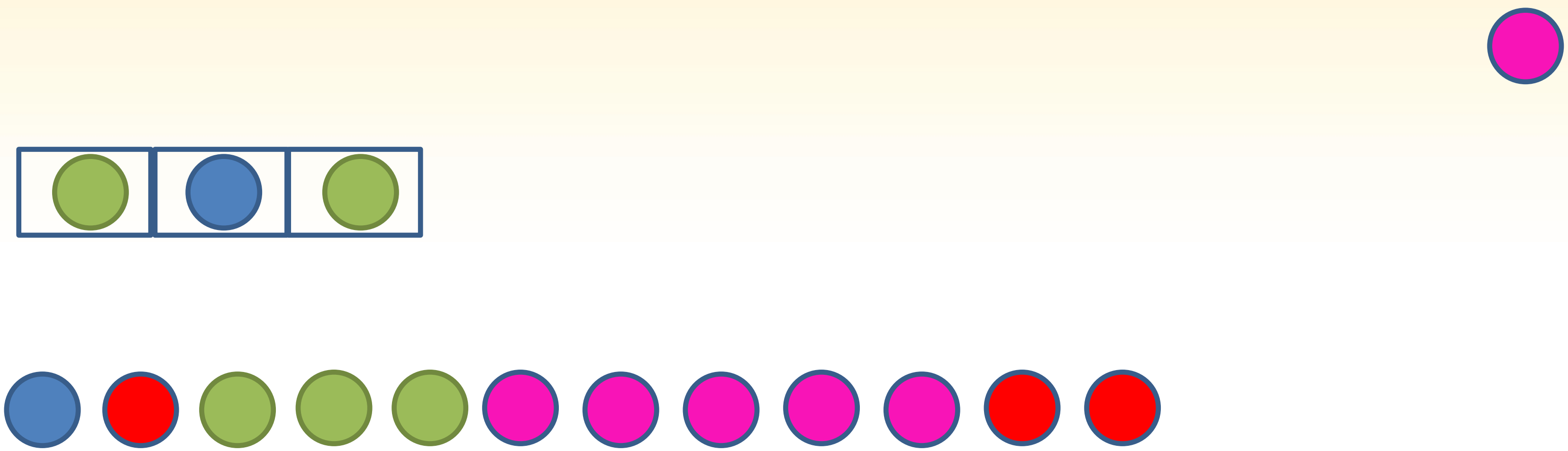
Problem Definition



--Based on Uniform Metric

θ

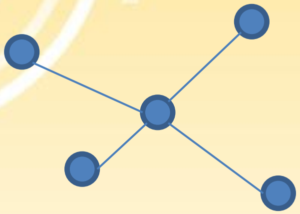
σ



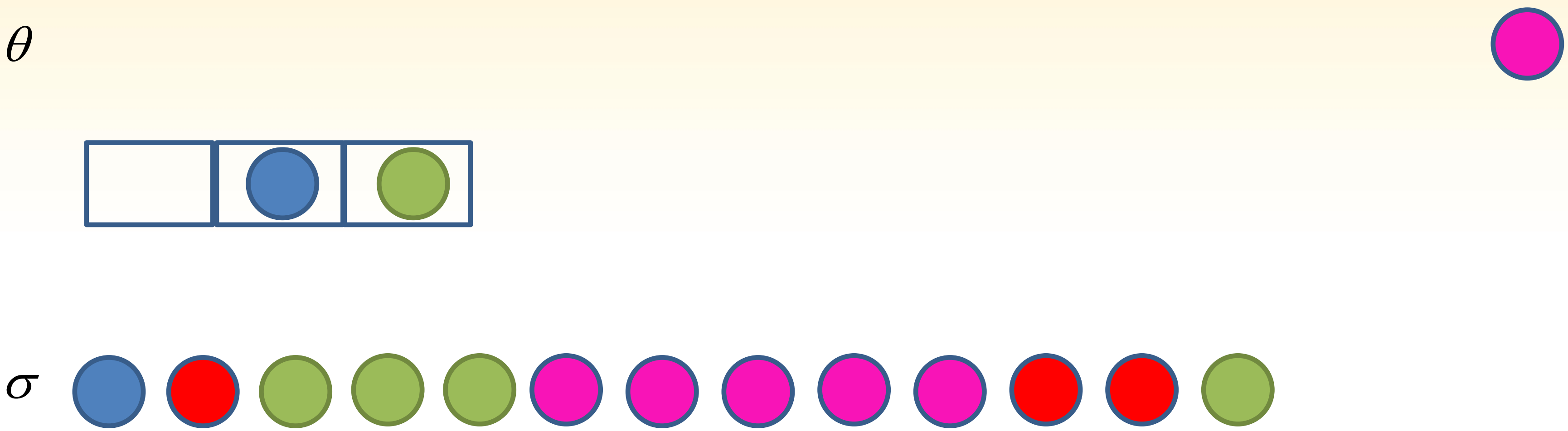
Cost: 5

w_g w_p w_b w_r

Problem Definition



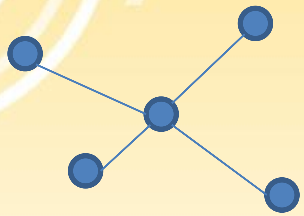
--Based on Uniform Metric



Cost: 6

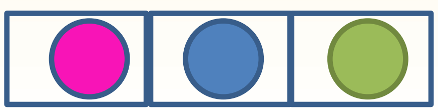
w_g w_p w_b w_r

Problem Definition

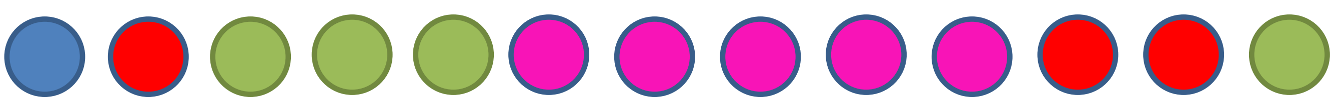


--Based on Uniform Metric

θ



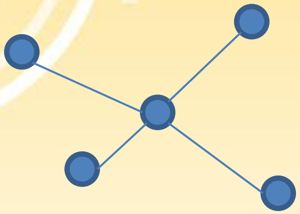
σ



Cost: 6

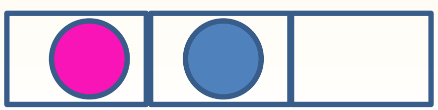
w_g w_p w_b w_r

Problem Definition

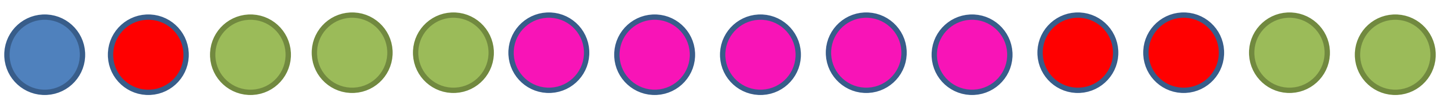


--Based on Uniform Metric

θ



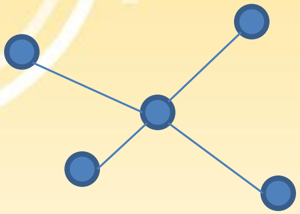
σ



Cost: 6

w_g w_p w_b w_r

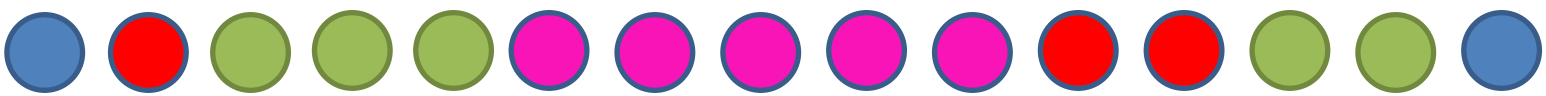
Problem Definition



--Based on Uniform Metric

θ

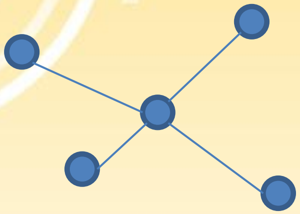
σ



Cost: 7

w_g w_p w_b w_r

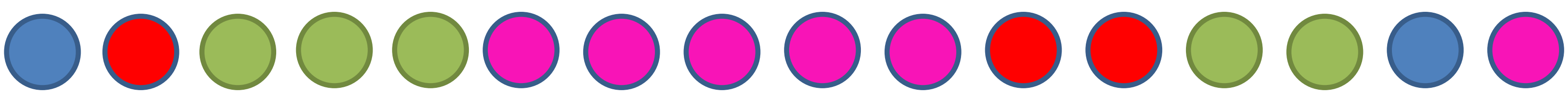
Problem Definition



--Based on Uniform Metric

θ

σ



Cost: 8

w_g w_p w_b w_r

Layout

- Problem Definition
- Motivation
- Related Work
- Algorithm:

Almost Tight Bounds for Reordering Buffer Management,

Anna Adamaszek and Artur Czumaj, STOC 2011

Motivation

- Numerous Application
  - Automotive assembly paint shop
  - Graphic rendering processors, storage systems network optimization
  - Inverted index compression
- Buffers are pervasive in computer and production systems
- Simple, elegant, natural, non-trivial, and appealing model (a sensible generalization of lookahead)

A sensible generalization of **LOOKAHEAD**

- There are different variations on the exact type of information provided to the algorithm under **lookahead** but arguably the most common one is to assume that, at every point in time, the algorithm has knowledge of the attributes of the next k tasks to arrive. This assumption is justified by the fact that, in practice, tasks may not always strictly arrive one-by-one and therefore, **a certain number of tasks are always waiting in a queue to be processed.**
- In recent years, so-called reordering buffers have been studied as a sensible generalization of **lookahead.** The basic idea is that, in problem settings where the **order** in which the tasks are processed is not important, we can permit a scheduling algorithm to **choose to process any task waiting in the queue**. This stands in contrast to look-ahead, where the algorithm has knowledge of all the tasks in the queue but still has to process them in the order they arrived.

Layout

- Problem Definition
- Motivation
- Related Work
- Algorithm:

Almost Tight Bounds for Reordering Buffer Management,

Anna Adamaszek and Artur Czumaj, STOC 2011

Related Work

- Mostly in the online setting (competitive analysis)
  - Upper bounds[RSW'02, EW'05, AR'10, STOC'11]—STOC'11($\Omega(\log\log k)$)
  - Lower bounds[STOC'11] $\Omega(\sqrt{\log k / \log\log k})$(det.) $O(\sqrt{\log k})$ (rand.)
- The algorithms balance between removing large color blocks and removing older color blocks
- E.g. the following algorithm gives $\mathrm{O}(\log k)$
  - While the buffer contains an item of the current color,
  - Switch to a color with maximum total penalty and
  - Penalize each item the buffer by 1/k.
- The problem is NP-hard[AKM'10]

lower bound VS upper bound

- Lower bound (all the algorithms)
  - ➢ As for online, it aims at the competitive ratio of all the online algorithms (NP-hard problems: offline algorithms should also be bounded)
- Upper bound (all the sequences)
  - ➢ As for online, it aims at designing new online algorithms to get smaller competitive ratio

- People attempt to make lower bound and upper bound more and more tight

- We focus on the upper bound of an online algorithm

Layout

- Problem Definition
- Motivation
- Related Work
- Algorithm:

Almost Tight Bounds for Reordering Buffer Management,

Anna Adamaszek and Artur Czumaj, STOC 2011

Almost Tight Bounds for Reordering Buffer Management,
Anna Adamaszek and Artur Czumaj,
STOC 2011

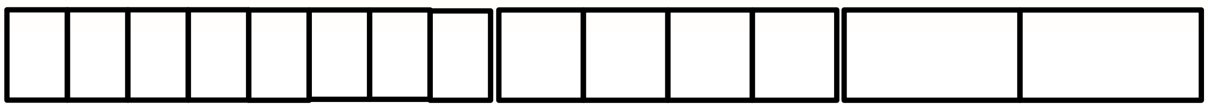
- Premise
- ➢ Class
- Lower bound of any online algorithms
- Online algorithm (upper bound)
- ➢ penalty

Almost Tight Bounds for Reordering Buffer Management,
Anna Adamaszek and Artur Czumaj,
STOC 2011

- Premise
- Class
- Lower bound of any online algorithms
- Online algorithm (upper bound)
- penalty

Partition the buffer

- Class – A color is in class i, i=1,2,…, $\log k$ at time t if the algorithm stores between 2^{i-1} to 2^{i} elements of this color at time t.



Almost Tight Bounds for Reordering Buffer Management,
Anna Adamaszek and Artur Czumaj,
STOC 2011

- Premise
  - partition of the buffer
- Lower bound of any online algorithms
- Online algorithm (upper bound)
  - penalty

lower bound of any online alg.

- Basic idea:
  - For any online algorithms of this problem, we can find a sequence (worst sequence)
  - There exists an OPT under this worst sequence consuming smaller cost than any online algorithms
  - The gap grows to $\Omega(\sqrt{\log k / \log\log k})$

- The sequence must satisfy some properties:
  - Define OPT' whose buffer has a size of $(1+\alpha)k$
  - OPT' can output a permutation σ , in which the number θ of color changes equals to that of different colors in

Almost Tight Bounds for Reordering Buffer Management, Anna Adamaszek and Artur Czumaj, STOC 2011

- Premise
  - Class
- Lower bound of any online algorithms
- Online algorithm (upper bound)
  - penalty

Penalty(from a survey)

- The previous algorithms introduce a penalty counter for each color

- PENALTY:
  - Initially set penalty for each color to 0
  - Incase you need to do a color-change:
    - Increase the penalty of each color by the number of elements that are in the buffer (uniform metric).
    - Switch to an arbitrary color. Set its counter to 0

The online algorithm

Penalty distribution + Marking:

- Initially set penalty for each color Pc to 0.
- If there is no marked color choose a class and mark all colors in this class
- Switch to an arbitrary marked color. Unmark the color. Set its counter Pc to zero
- Increase the penalty Pc of every color by a value proportional to the number of elements n_c in color c stored in the buffer

Algorithm 1 Largest Color Class (LCC)

Output: a new output color
// let n_c denote the number of elements
// with color c in the buffer
$\forall$ colors $c : t_c \leftarrow \frac{w_c - P_c}{n_c/k}; \quad t \leftarrow \min(\{t_c \mid \text{color } c\} \cup \{P\});$
$P \leftarrow P - t; \quad \forall$ colors $c : P_c \leftarrow P_c + \frac{n_c}{k} \cdot t$
// the above ensures that t is small enough such that
// $P \geq 0$ and $P_c \leq w_c$ for all c
if $P = 0$ **then**
  if no marked color exists **then**
    // let $C_{\max}$ denote the class that occupies
    // the largest space in the buffer
    mark all colors in $C_{\max}$
  end if
  // let c_m denote an arbitrary marked color
  $P \leftarrow w_{c_m}$
  $P_{c_m} \leftarrow 0$
  unmark color c_m
  return color c_m as the new output color
else
  $c_a \leftarrow \arg\min_c t_c$ // pick color c_a such that $P_{c_a} = w_{c_a}$
  $P_{c_a} \leftarrow 0$
  unmark color c_a if it was marked
  return color c_a as the new output color
end if

Thanks!

Q&A