STACS 2025
2025/3/6

Online Matching with Delays and Size-based Costs

The University of Tokyo

Yasushi Kawase, Tomohiro Nakayoshi

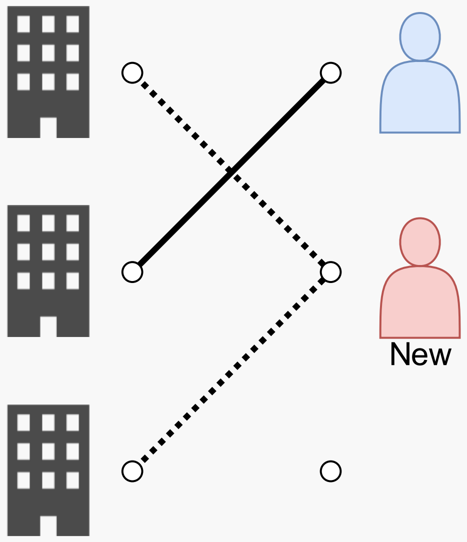
Background of Online Matching

Online bipartite matching

[Karp-UVazirani-VVazirani 1990]

e.g.) Ad allocation [Mehta 2013]

Users are
matched upon arrival

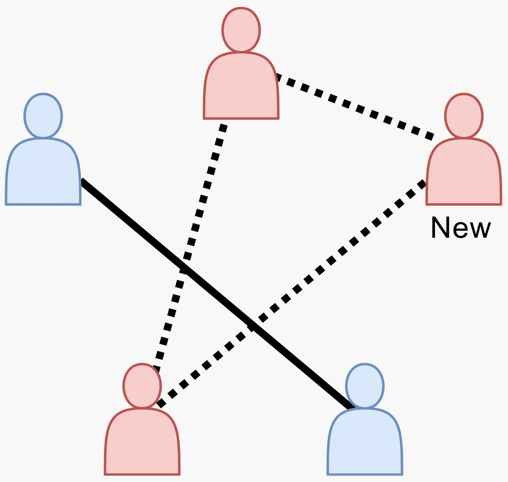


Online matching with delays

[Emek-Kutten-Wattenhofer 2016]

e.g.) Matchmaking in online games
Courier allocation in food delivery service

Users can be put on hold
at a cost after arrival



In these settings, exactly 2 requests are matched each time.

Motivation for Size-based Costs

Situations where requests can be processed with other than 2

- Matchmaking in k-player online games
  - Battle royale can start even with fewer players,
  - but players’ satisfaction decreases in a match with fewer player.
- Batch-processing API server with deep learning models
  - can process even if the capacity is not met,
  - but it is inefficient for handling many requests.

Motivation for Size-based Costs

Situations where requests can be processed with other than 2

- Matchmaking in k-player online games
  - Battle royale can start even with fewer players,
  - but players' satisfaction decreases in a match with fewer player.
- Batch-processing API server with deep learning models
  - can process even if the capacity is not met,
  - but it is inefficient for handling many requests.

Motivation for Size-based Costs

Situations where requests can be processed with other than 2

- Matchmaking in k-player online games
  - Battle royale can start even with fewer players,
  - but players' satisfaction decreases in a match with fewer player.
- Batch-processing API server with deep learning models
  - can process even if the capacity is not met,
  - but it is inefficient for handling many requests.

Introduce penalty with size-based costs

Example: 4-Player Game (e.g., CATAN)

Platform fills missing players with AI players to start the game

- **Size cost:** incurred when a match has fewer than 4 players (cost = 1)
  - No size cost if there are 4 players.
  - Players prefer matches with only human players.
- **Waiting cost:** incurred per waiting player per unit time (cost = 1)

Goal: Minimize the sum of size cost and waiting cost.

Example: 4-Player Game

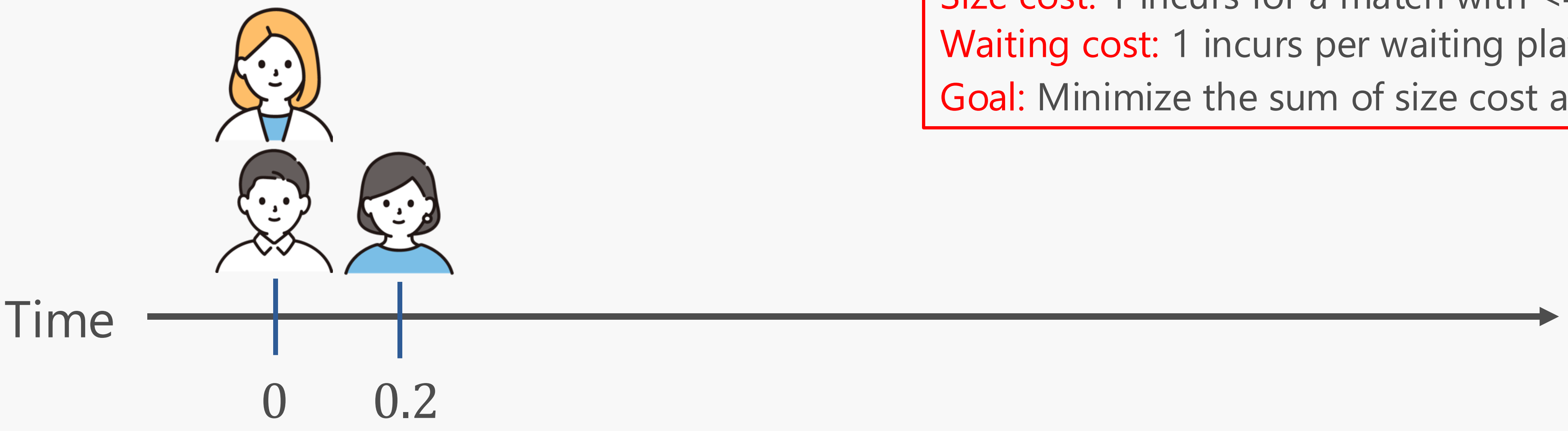
Size cost: 1 incurs for a match with <4 players
Waiting cost: 1 incurs per waiting player per unit time
Goal: Minimize the sum of size cost and waiting cost

Size cost

Waiting cost

Example: 4-Player Game

Size cost: 1 incurs for a match with <4 players
Waiting cost: 1 incurs per waiting player per unit time
Goal: Minimize the sum of size cost and waiting cost

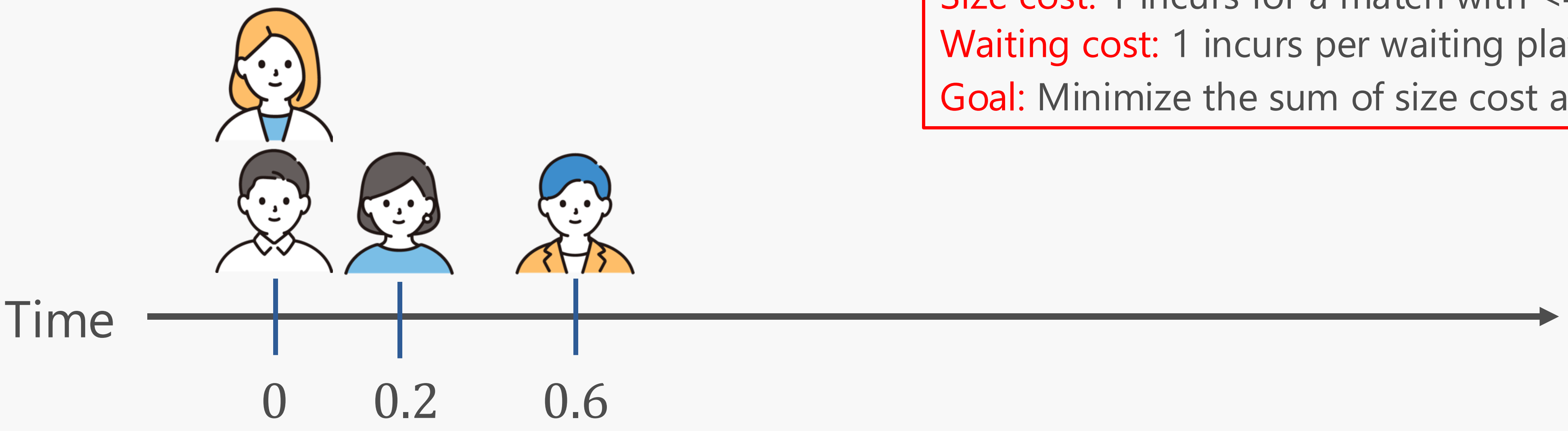


Size cost

Waiting cost

Example: 4-Player Game

Size cost: 1 incurs for a match with <4 players
Waiting cost: 1 incurs per waiting player per unit time
Goal: Minimize the sum of size cost and waiting cost

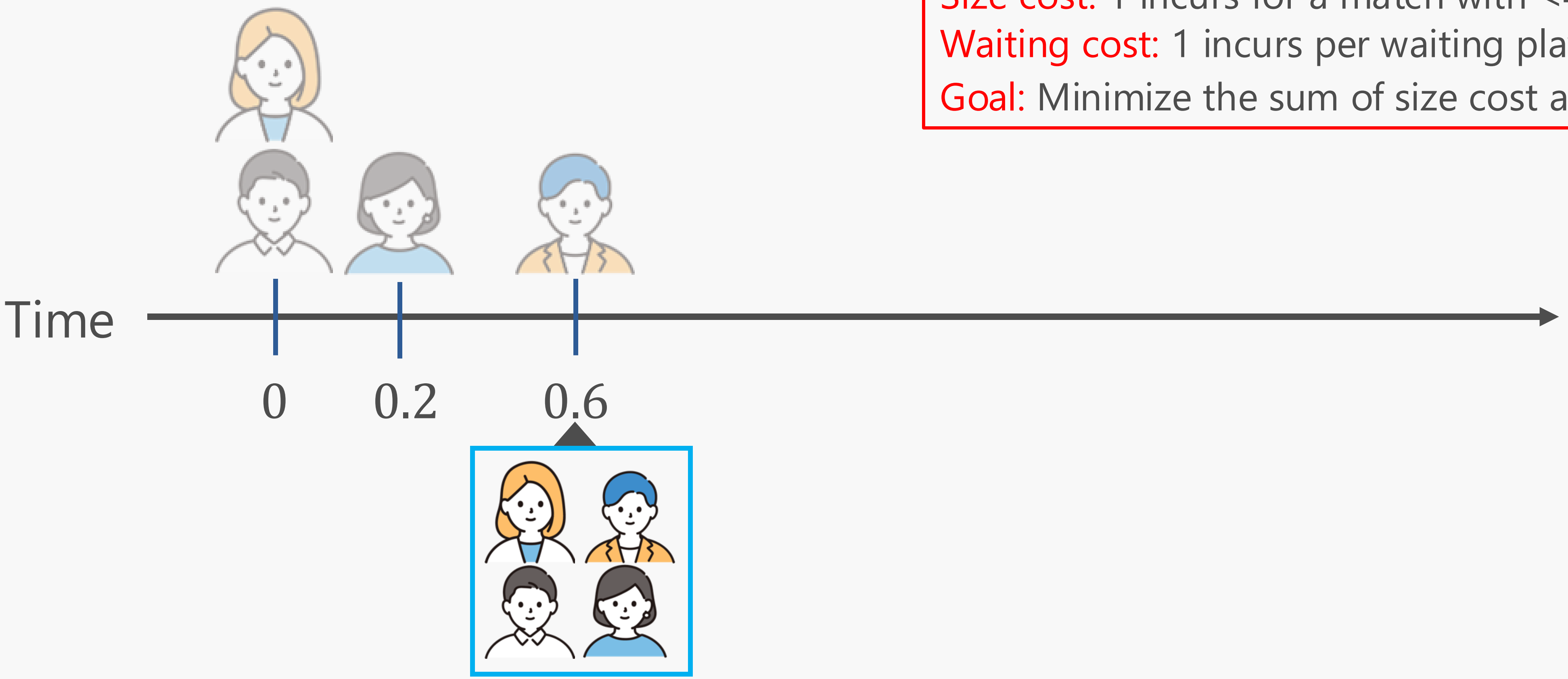


Size cost

Waiting cost

Example: 4-Player Game

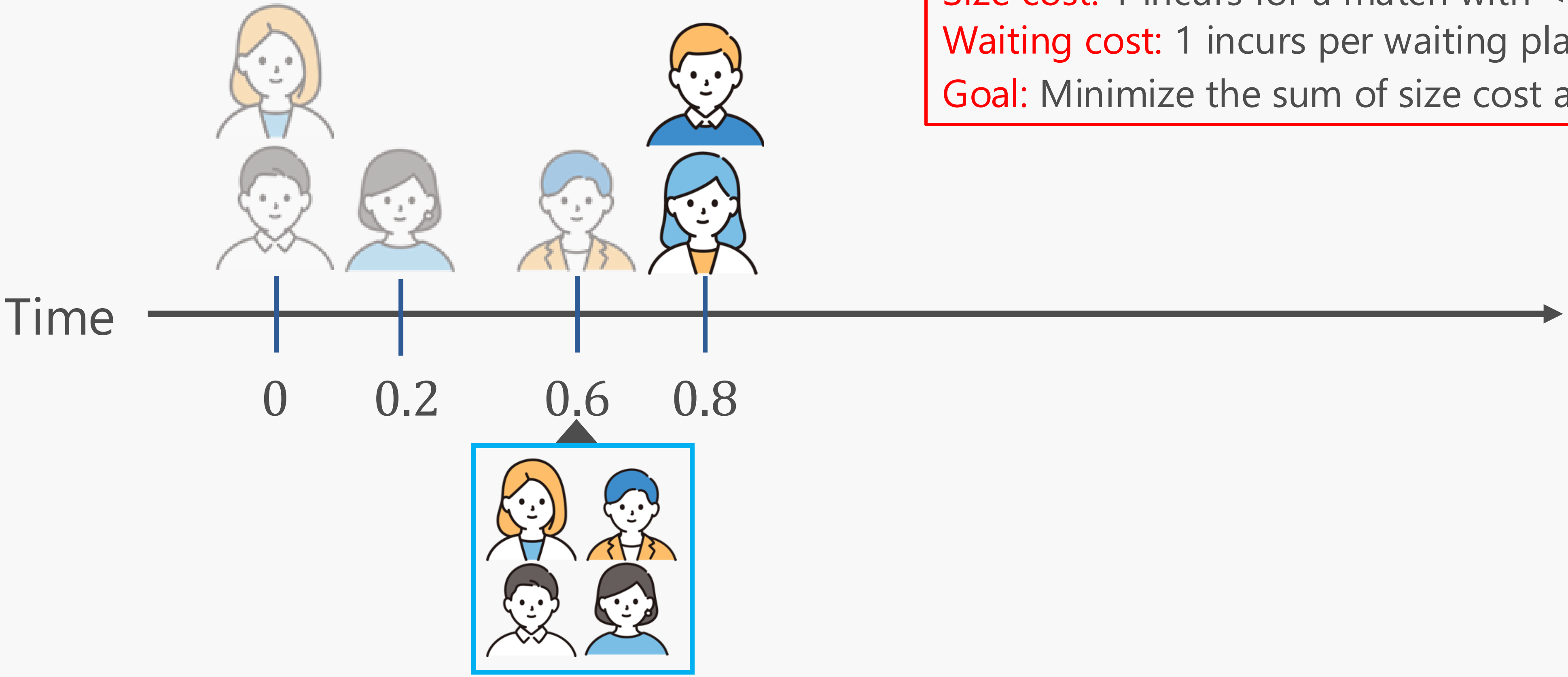
Size cost: 1 incurs for a match with <4 players
Waiting cost: 1 incurs per waiting player per unit time
Goal: Minimize the sum of size cost and waiting cost



Size cost		0
Waiting cost	0.6 + 0.6 +0.4	+0

Example: 4-Player Game

Size cost: 1 incurs for a match with <4 players
Waiting cost: 1 incurs per waiting player per unit time
Goal: Minimize the sum of size cost and waiting cost

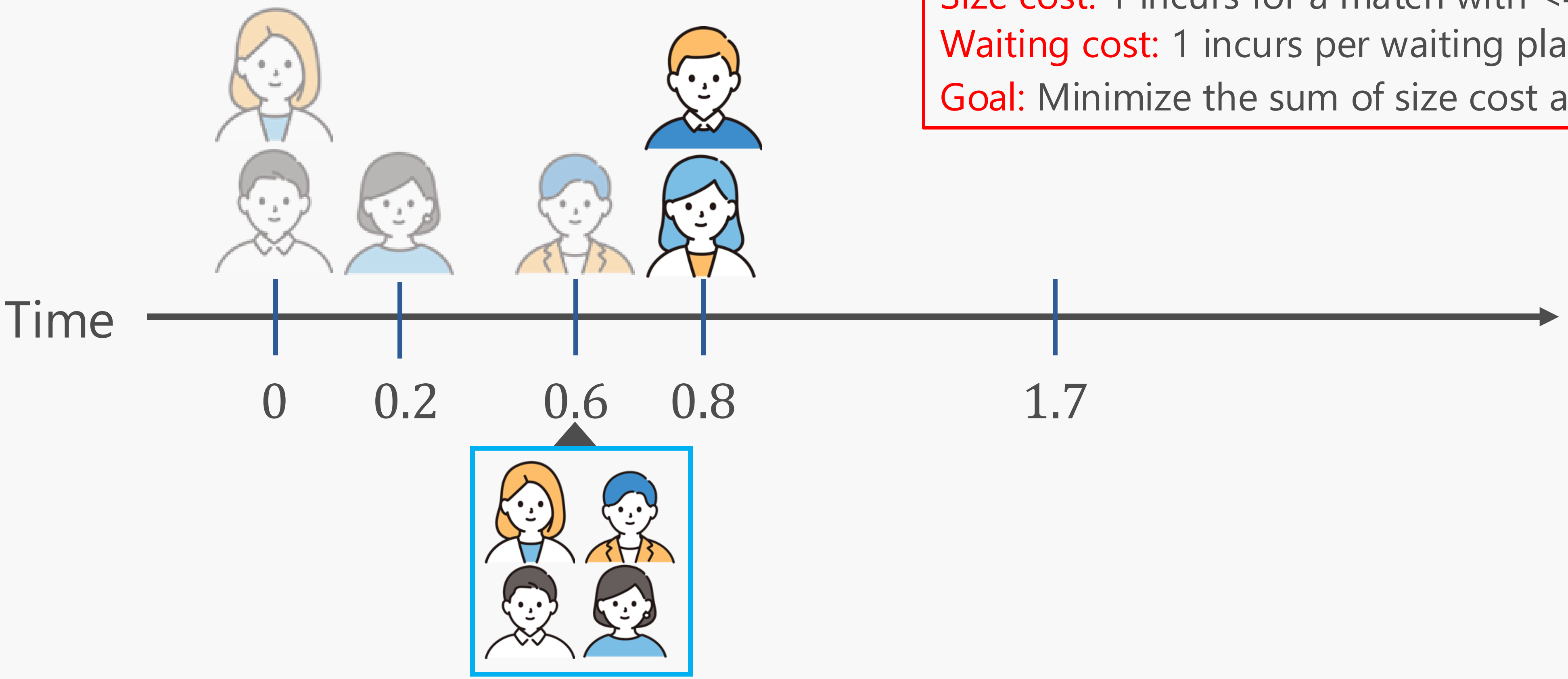


Size cost 0

Waiting cost $0.6 + 0.6 + 0.4$ $+0$

Example: 4-Player Game

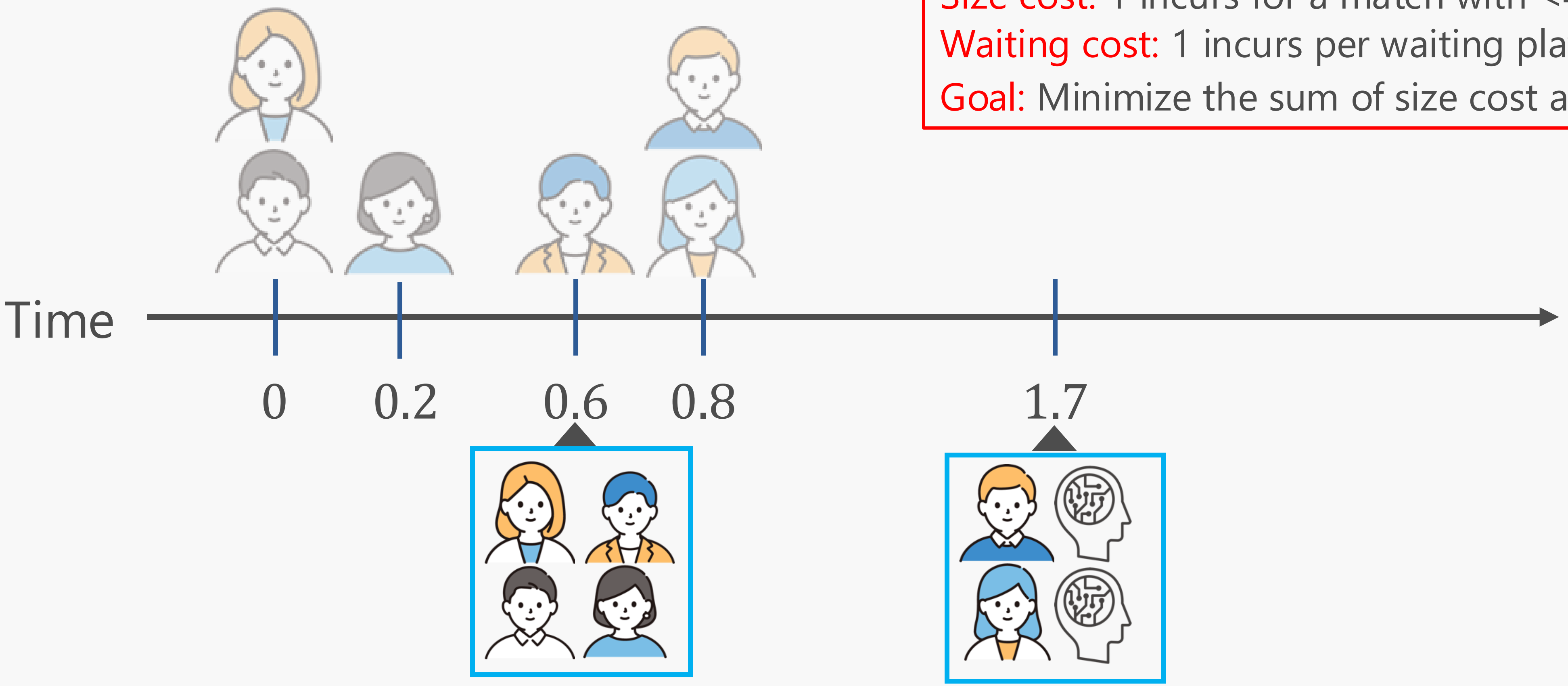
Size cost: 1 incurs for a match with <4 players
Waiting cost: 1 incurs per waiting player per unit time
Goal: Minimize the sum of size cost and waiting cost



Size cost		0
Waiting cost	$0.6 + 0.6 + 0.4$	$+0$

Example: 4-Player Game

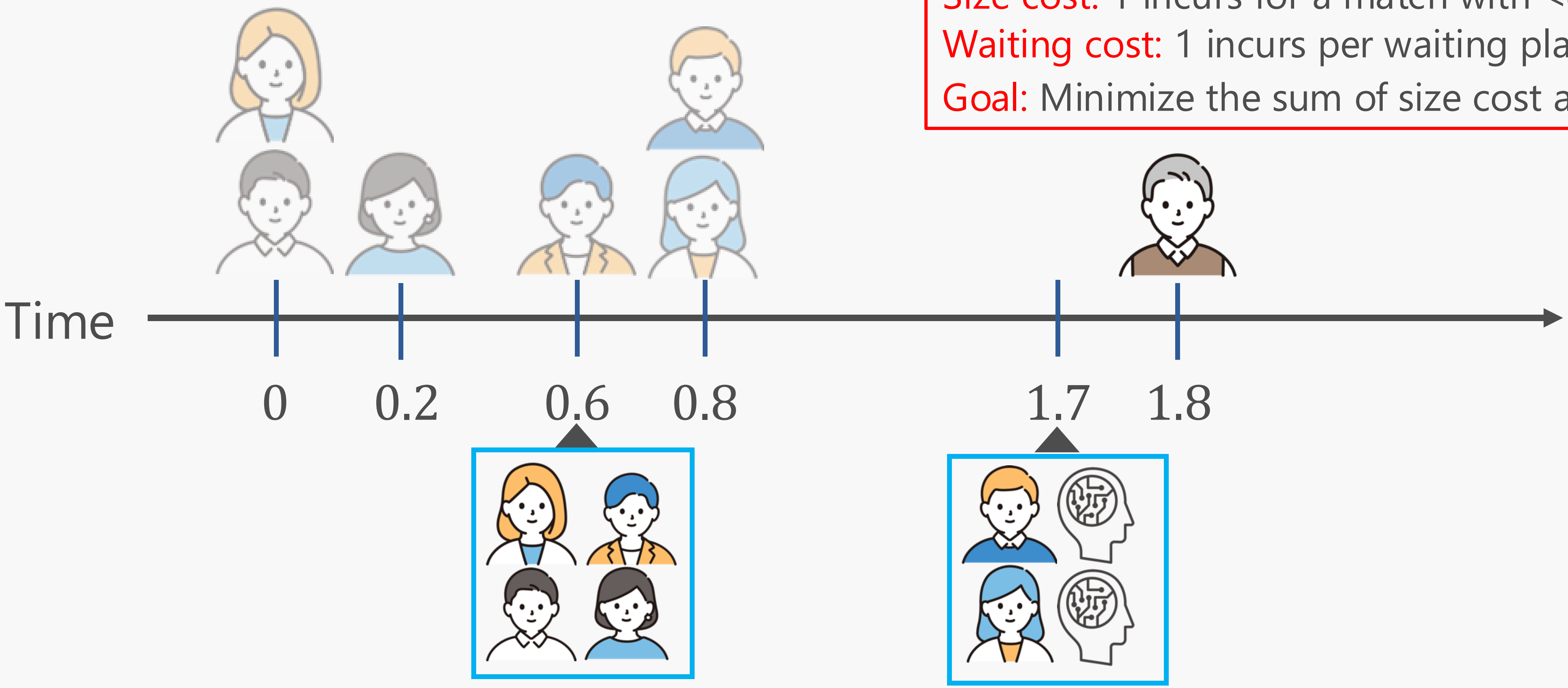
Size cost: 1 incurs for a match with <4 players
Waiting cost: 1 incurs per waiting player per unit time
Goal: Minimize the sum of size cost and waiting cost



Size cost		0		+1
Waiting cost	0.6 + 0.6 +0.4	+0	+0.9 + 0.9	

Example: 4-Player Game

Size cost: 1 incurs for a match with <4 players
Waiting cost: 1 incurs per waiting player per unit time
Goal: Minimize the sum of size cost and waiting cost

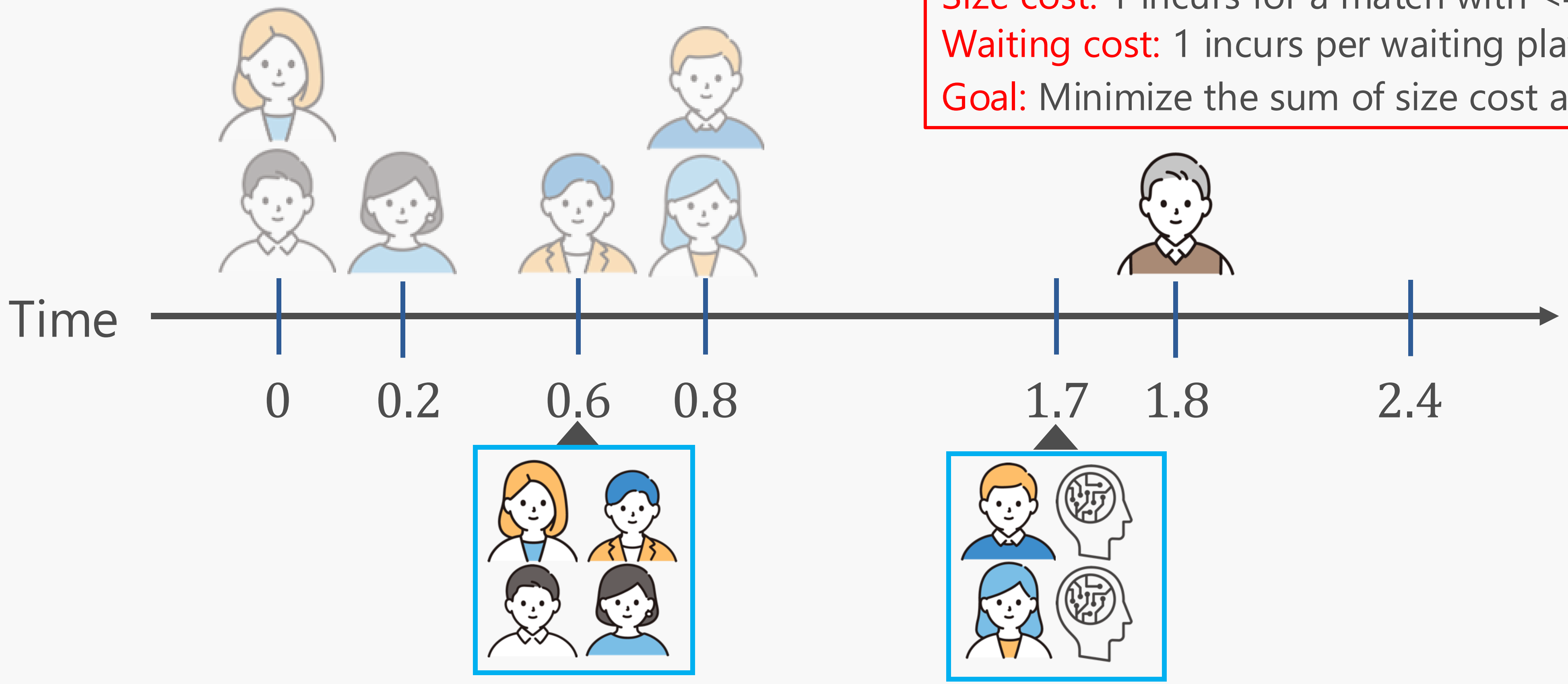


Size cost 0 +1

Waiting cost $0.6 + 0.6 + 0.4$ $+0$ $+0.9 + 0.9$

Example: 4-Player Game

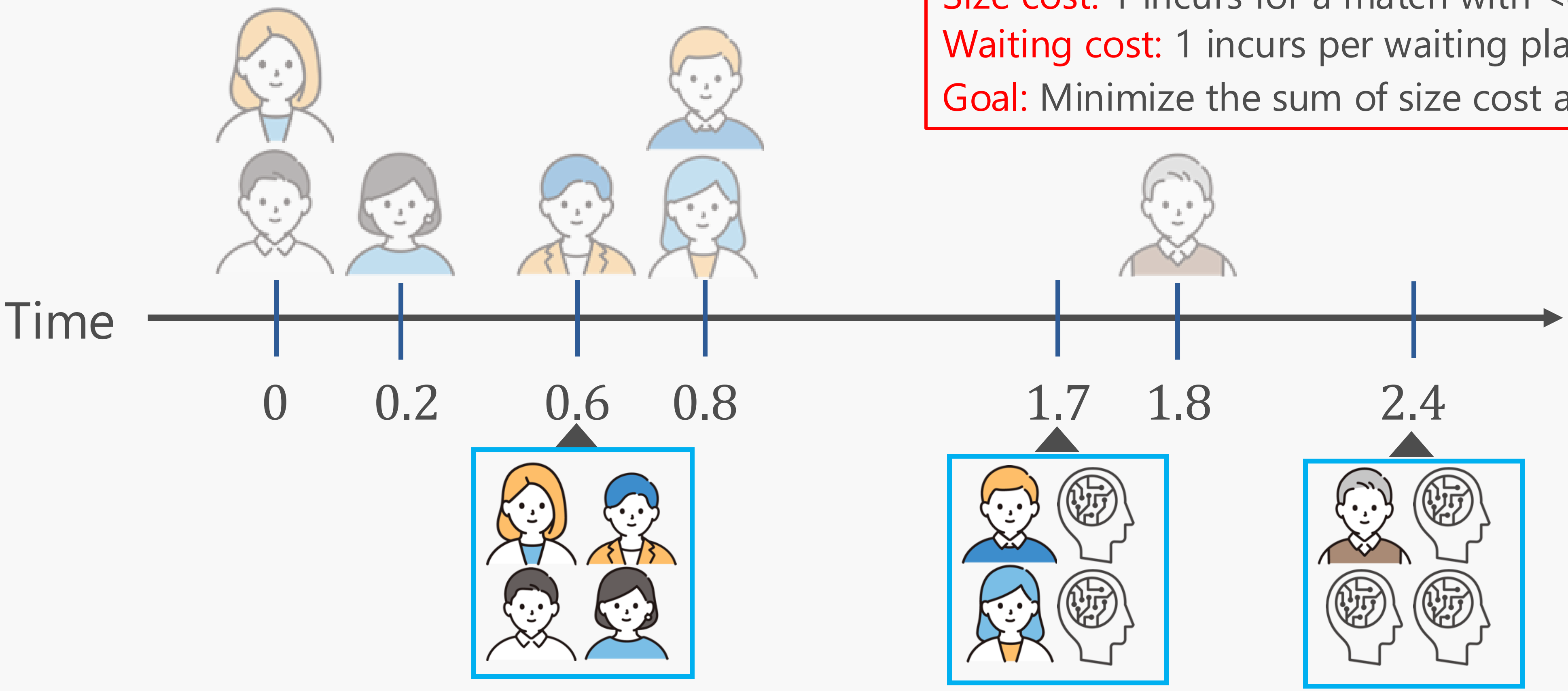
Size cost: 1 incurs for a match with <4 players
Waiting cost: 1 incurs per waiting player per unit time
Goal: Minimize the sum of size cost and waiting cost



Size cost		0		+1
Waiting cost	0.6 + 0.6 +0.4	+0	+0.9 + 0.9	

Example: 4-Player Game

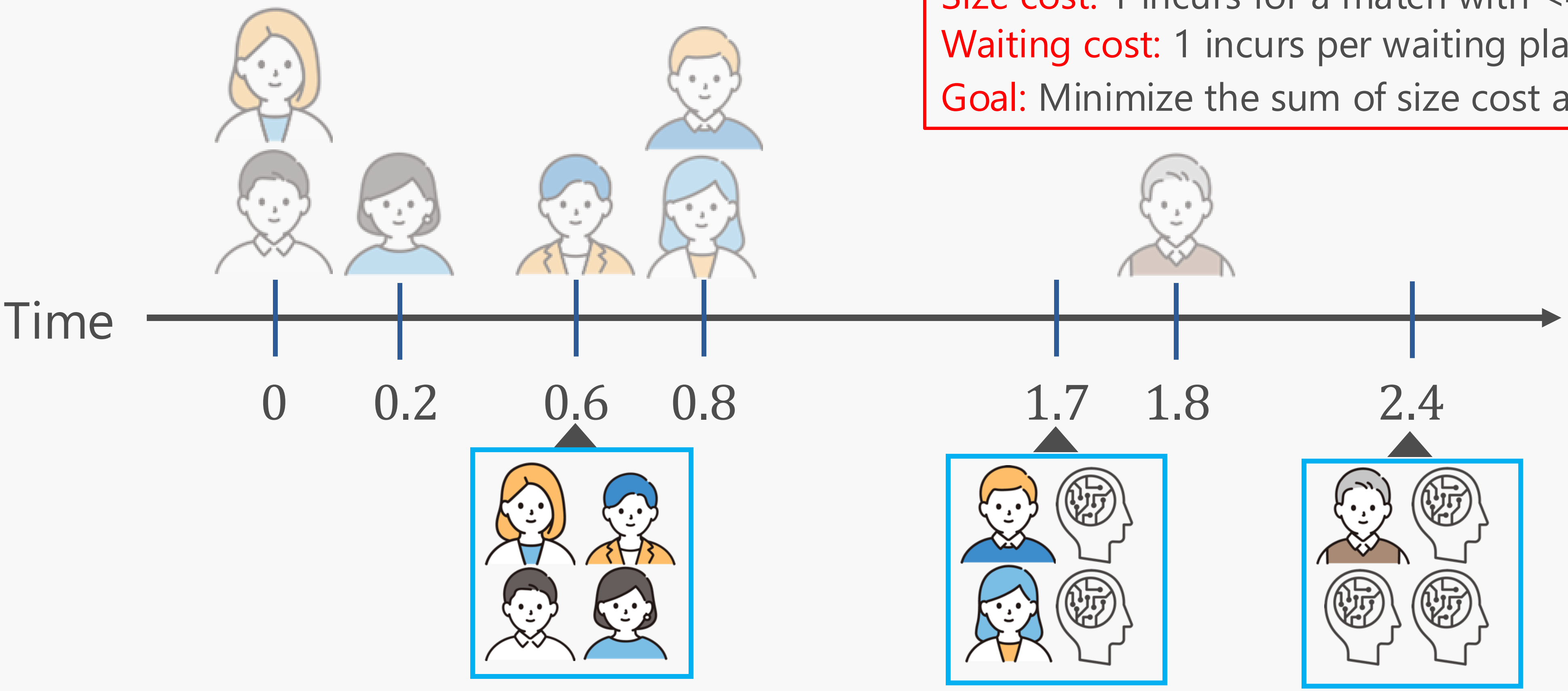
Size cost: 1 incurs for a match with <4 players
Waiting cost: 1 incurs per waiting player per unit time
Goal: Minimize the sum of size cost and waiting cost



Size cost		0		+1		+1
Waiting cost	0.6 + 0.6 +0.4	+0	+0.9 + 0.9		+0.6	

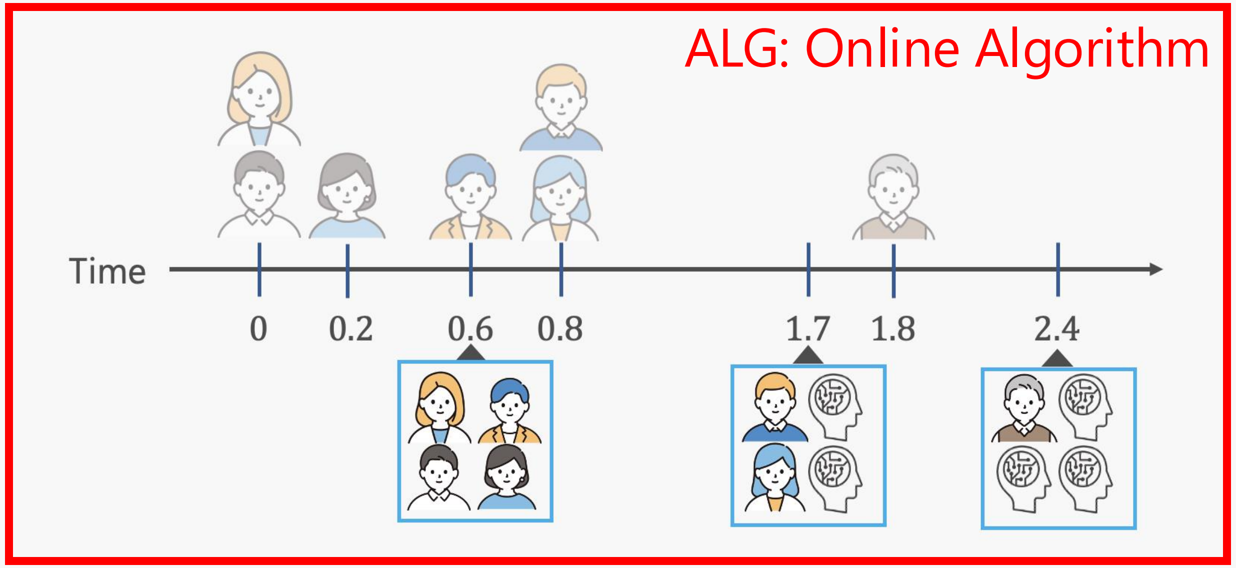
Example: 4-Player Game

Size cost: 1 incurs for a match with <4 players
Waiting cost: 1 incurs per waiting player per unit time
Goal: Minimize the sum of size cost and waiting cost

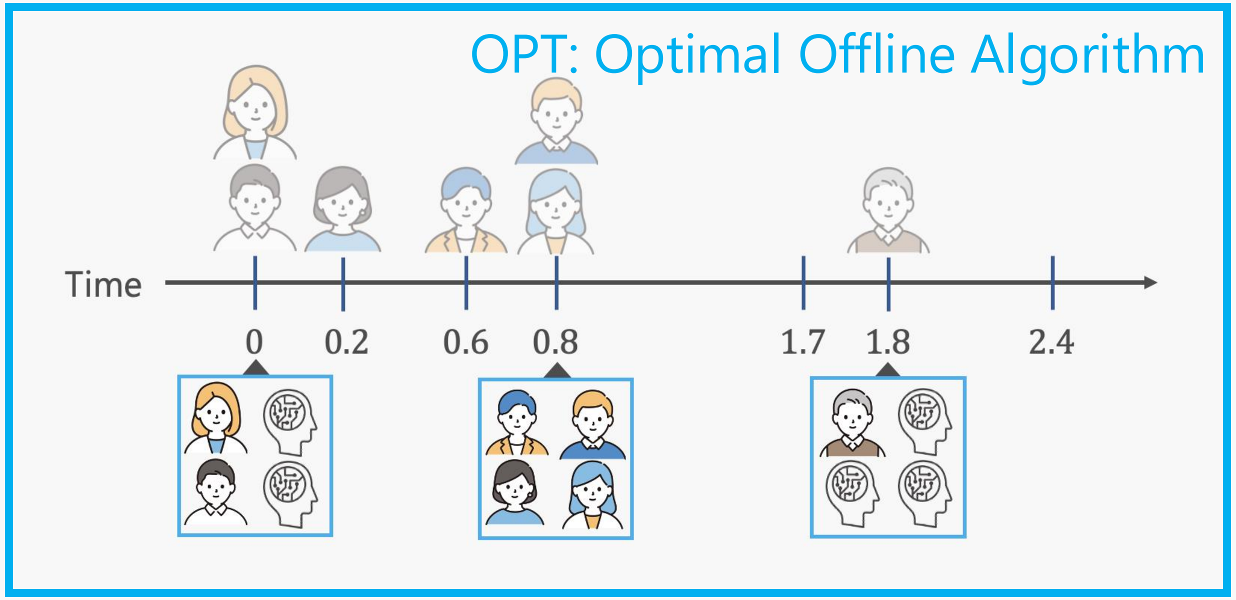


Size cost		0		+1	+1	Total: 6
Waiting cost	0.6 + 0.6 +0.4	+0	+0.9 + 0.9		+0.6	

Comparison with Optimal Offline Solution

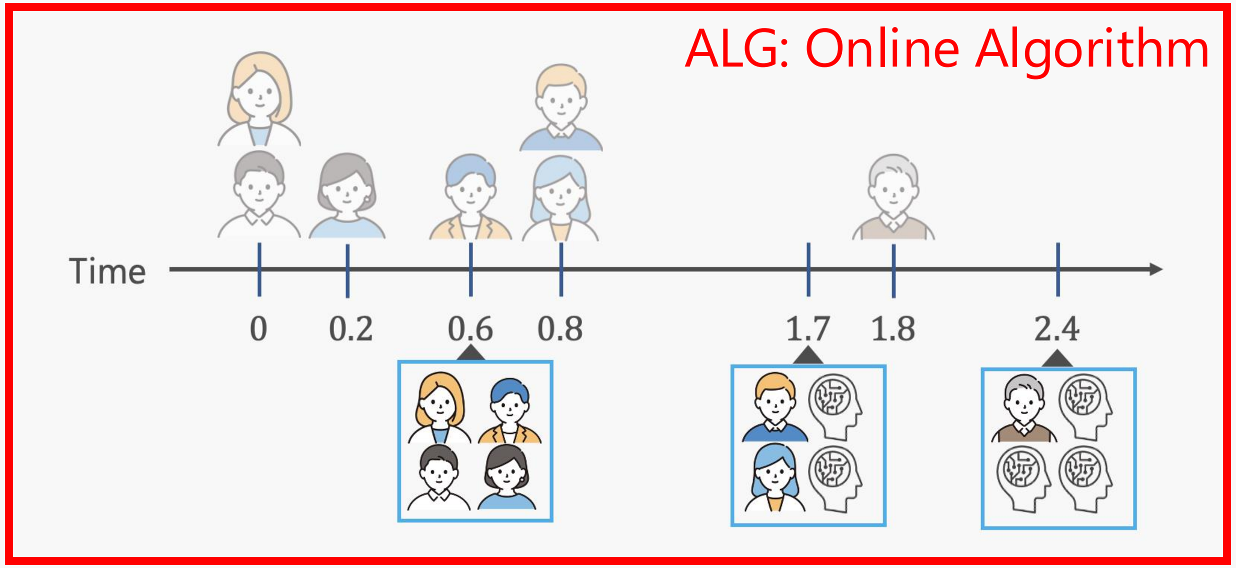


Total cost: 6

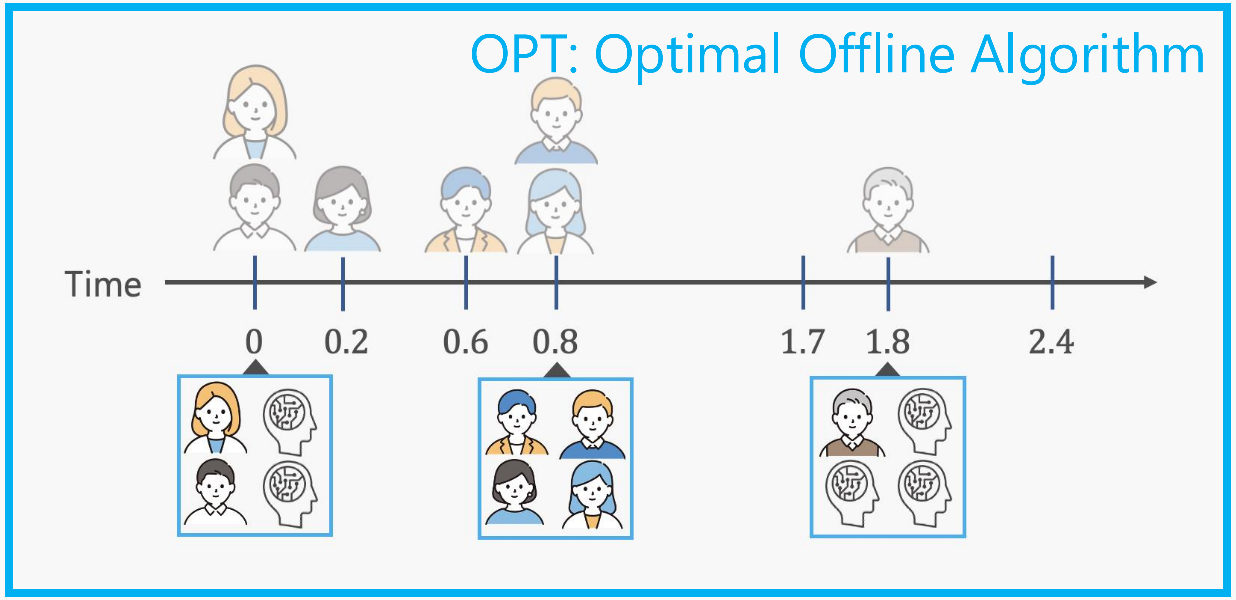


Total cost: 2.8

Comparison with Optimal Offline Solution



Total cost: 6



Total cost: 2.8

Cost Ratio: $\frac{6}{2.8}$

Competitive ratio

Worst case cost ratio of ALG to OPT

$$\sup_{\sigma \in \mathcal{I}} \frac{\mathrm{ALG}(\sigma)}{\mathrm{OPT}(\sigma)}$$

Our Problem: Online Matching with Delays and Size-based Costs

- Requests arrive sequentially in real-time.
- The algorithm performs matching sequentially in real-time.
- All requests must be matched.

$\Big\rangle$ Min. total match cost

Match cost for a subset S:

$$\underbrace{f(|S|)}_{\text{size cost}} + \underbrace{\sum_{v \in S} (\text{waiting time of } v)}_{\text{waiting cost}}$$

Our Problem: Online Matching with Delays and Size-based Costs

- Requests arrive sequentially in real-time.
- The algorithm performs matching sequentially in real-time.
- All requests must be matched.

Min. total match cost

Match cost for a subset S: $\underbrace{f(|S|)}_{\text{size cost}} + \underbrace{\sum_{v \in S} (\text{waiting time of } v)}_{\text{waiting cost}}$

This study focuses on binary penalty functions $f\colon \mathbb{Z}_{++} \to \{0, 1\}$.

Related Work

Online Matching with Delays and Size-based Costs (This study)

cost $f(|S|) + \sum_{v \in S} (\text{waiting time of } v)$

TCP Acknowledgment Problem

[Dooly-Goldman-Scott 2001]

cost $1 + \sum_{v \in S} (\text{waiting time of } v)$

MPMDfp for single source

[Emek-Kutten-Wattenhofer 2016] [Emek-Shapiro-Wang 2019]

cost $f(|S|) + \sum_{v \in S} (\text{waiting time of } v)$

$f(|S|) = |S| \bmod 2$ (i.e., $f = (1, 0, 1, 0, 1, 0, \dots)$)

Online k-way Matching with Delays

[Melnyk-Wang-Wattenhofer 2021]

cost $d(S) + \sum_{v \in S} (\text{waiting time of } v)$

$d(S)$: distance function

$|S|$ must be k

Online Weighted Cardinality Joint Replenishment Problem with Delay

[Chen-Khatkar-Umboh 2022]

cost $f\left(\sum_{i \in \{t(v) \mid v \in S\}} w_i\right) + \sum_{v \in S} (\text{waiting time of } v)$

Online Matching with Delays and Size-based Costs (This study)

cost $f(|S|) + \sum_{v \in S} (\text{waiting time of } v)$

TCP Acknowledgment Problem

[Dooly-Goldman-Scott 2001]

cost $1 + \sum_{v \in S} (\text{waiting time of } v)$

MPMDfp for single source

[Emek-Kutten-Wattenhofer 2016] [Emek-Shapiro-Wang 2019]

cost $f(|S|) + \sum_{v \in S} (\text{waiting time of } v)$

$f(|S|) = |S| \bmod 2$ (i.e., $f = (1, 0, 1, 0, 1, 0, \dots)$)

Online k-way Matching with Delays

[Melnyk-Wang-Wattenhofer 2021]

cost $d(S) + \sum_{v \in S} (\text{waiting time of } v)$

$d(S)$: distance function

$|S|$ must be k

Online Weighted Cardinality Joint Replenishment Problem with Delay

[Chen-Khatkar-Umboh 2022]

can express weighted cardinality

cost $f\left(\sum_{i \in \{t(v) \mid v \in S\}} w_i \right) + \sum_{v \in S} (\text{waiting time of } v)$

Online Matching with Delays and Size-based Costs (This study)

cost $f(|S|) + \sum_{v \in S} (\text{waiting time of } v)$

TCP Acknowledgment Problem

[Dooly-Goldman-Scott 2001]

cost $1 + \sum_{v \in S} (\text{waiting time of } v)$

MPMDfp for single source

[Emek-Kutten-Wattenhofer 2016] [Emek-Shapiro-Wang 2019]

cost $f(|S|) + \sum_{v \in S} (\text{waiting time of } v)$

$f(|S|) = |S| \bmod 2$ (i.e., $f = (1, 0, 1, 0, 1, 0, \dots)$)

Online k-way Matching with Delays

[Melnyk-Wang-Wattenhofer 2021]

cost $d(S) + \sum_{v \in S} (\text{waiting time of } v)$

$d(S)$: distance function

$|S|$ must be k

Online Weighted Cardinality Joint Replenishment Problem with Delay

[Chen-Khatkar-Umboh 2022]

can express weighted cardinality

cost $f\left(\sum_{i \in \{t(v) \mid v \in S\}} w_i\right) + \sum_{v \in S} (\text{waiting time of } v)$

restricted: monotonically non-decreasing concave

Modified Penalty Function

Match cost for a subset S: $f(|S|) + \sum_{v \in S} (\text{waiting time of } v)$

e.g.) Consider a size cost for a game playable with 3 or 4 players (Actually, CATAN is for 3-4 players).

n	1	2	3	4	5	6	7	8	9	10	…
$f(n)$	1	1	0	0	1	1	1	1	1	1	…

Modified Penalty Function

Match cost for a subset S: $f(|S|) + \sum_{v \in S} (\text{waiting time of } v)$

e.g.) Consider a size cost for a game playable with 3 or 4 players (Actually, CATAN is for 3-4 players).

n	1	2	3	4	5	6	7	8	9	10	…
$f(n)$	1	1	0	0	1	1	1	1	1	1	…

match requests separately instead of doing at once

n	1	2	3	4	5	6	7	8	9	10	…
$\bar{f}(n)$	1	1	0	0	1	0	0	0	0	0	…

Modified Penalty Function

Match cost for a subset S: $f(|S|) + \sum_{v \in S} (\text{waiting time of } v)$

e.g.) Consider a size cost for a game playable with 3 or 4 players (Actually, CATAN is for 3-4 players).

n	1	2	3	4	5	6	7	8	9	10	...
$f(n)$	1	1	0	0	1	1	1	1	1	1	...

match requests separately instead of doing at once

n	1	2	3	4	5	6	7	8	9	10	...
$\bar{f}(n)$	1	1	0	0	1	0	0	0	0	0	...

$f(3) + f(3) + f(4)$

We define a penalty function that has an optimal size cost, like the latter, as a modified penalty function.

Result for Binary Penalty Functions

Penalty function (with modification)	Competitive ratio
(i) always 1 (the number of processing is important)	2 [Dooly-Goldman-Scott 2001]
(ii) 0 if the size is a multiple of k (prefer size k)	$\Theta(\log k \,/\, \log\log k)$
(iii) other scenarios	**unbounded**

(i)

n	1	2	3	4	5	6	7	8	9	10	…
$\bar{f}$	1	1	1	1	1	1	1	1	1	1	…

(ii)

n	1	2	3	4	5	6	7	8	9	10	…
$\bar{f}$	1	1	1	0	1	1	1	0	1	1	…

(iii)

n	1	2	3	4	5	6	7	8	9	10	…
$\bar{f}$	1	1	0	0	1	0	0	0	0	0	…

Result for Binary Penalty Functions

Penalty function (with modification)	Competitive ratio
(i) always 1 (the number of processing is important) n: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, … $\bar{f}$: 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …	2 [Dooly-Goldman-Scott 2001]
(ii) 0 if the size is a multiple of k (prefer size k) n: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, … $\bar{f}$: 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, …	$\boldsymbol{\Theta(\log k \,/\, \log\log k)}$
(iii) other scenarios n: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, … $\bar{f}$: 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, …	**unbounded**

Algorithm Design Idea of Our Algorithm ALG

Bound ALG's cost by $O(\alpha)$ until OPT incurs a cost of at least 1 (phase)

a real α satisfying $\alpha^{\alpha} = k$, where $\alpha = \Theta(\log k \ / \ \log\log k)$

ALG moves to **the next phase** after ensuring that the cost of all algorithms exceeds 1.
ALG splits an instance into phases.

Algorithm Design Idea of Our Algorithm ALG

Bound ALG's cost by $O(\alpha)$ until OPT incurs a cost of at least 1 (phase)

a real α satisfying $\alpha^{\alpha} = k$, where $\alpha = \Theta(\log k \ / \ \log\log k)$

ALG moves to **the next phase** after ensuring that the cost of all algorithms exceeds 1.
ALG splits an instance into phases.

Example with 3 phases

$k = 3$

The cost of all algorithms must be evaluated per phase.

Algorithm Design Idea of Our Algorithm ALG

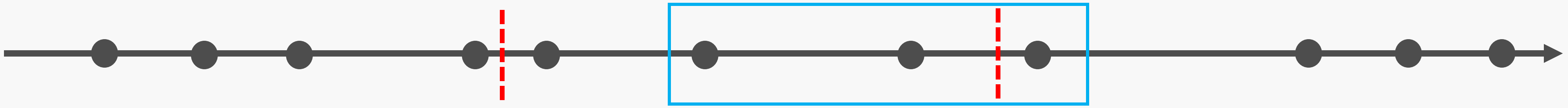
Bound ALG's cost by $O(\alpha)$ until OPT incurs a cost of at least 1 (phase)

a real α satisfying $\alpha^\alpha = k$, where $\alpha = \Theta(\log k \ / \ \log\log k)$

ALG moves to **the next phase** after ensuring that the cost of all algorithms exceeds 1.
ALG splits an instance into phases.

Example with 3 phases

$k = 3$



The cost of all algorithms must be evaluated per phase.

However, algorithms may carry over some requests from previous phases (**carry**).

The number of carries affects **waiting and size costs**, which must be considered.

Method to Bound OPT Cost in Each Phase

It is necessary to ensure that the cost of all algorithms is at least 1.

"All possible algorithms" are too many, so we narrow them down.

Matching a number of requests that is not a multiple of k incurs a size cost of 1.

Method to Bound OPT Cost in Each Phase

It is necessary to ensure that the cost of all algorithms is at least 1.

“All possible algorithms” are too many, so we narrow them down.

Matching a number of requests that is not a multiple of k incurs a size cost of 1.

→ Consider only algorithms that match exactly k requests at a time.

Method to Bound OPT Cost in Each Phase

It is necessary to ensure that the cost of all algorithms is at least 1.

"All possible algorithms" are too many, so we narrow them down.

Matching a number of requests that is not a multiple of k incurs a size cost of 1.

➡ Consider only algorithms that match exactly k requests at a time.

➡ Consider only algorithms that match immediately after k requests accumulate.

Method to Bound OPT Cost in Each Phase

It is necessary to ensure that the cost of all algorithms is at least 1.

"All possible algorithms" are too many, so we narrow them down.

Matching a number of requests that is not a multiple of k incurs a size cost of 1.

→ Consider only algorithms that match exactly k requests at a time.

→ Consider only algorithms that match immediately after k requests accumulate.

→ Algorithm's behavior depends only on carries.

Method to Bound OPT Cost in Each Phase

It is necessary to ensure that the cost of all algorithms is at least 1.

"All possible algorithms" are too many, so we narrow them down.

Matching a number of requests that is not a multiple of k incurs a size cost of 1.

→ Consider only algorithms that match exactly k requests at a time.

→ Consider only algorithms that match immediately after k requests accumulate.

→ Algorithm's behavior depends only on carries. → **The number of candidates reduces to k.**

(since the number of carries never exceeds k)

Method to Bound OPT Cost in Each Phase

It is necessary to ensure that the cost of all algorithms is at least 1.

"All possible algorithms" are too many, so we narrow them down.

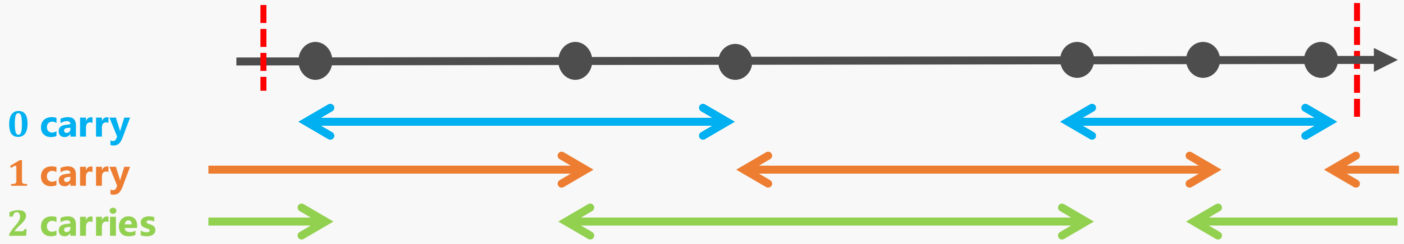
Matching a number of requests that is not a multiple of k incurs a size cost of 1.

➡ Consider only algorithms that match exactly k requests at a time.

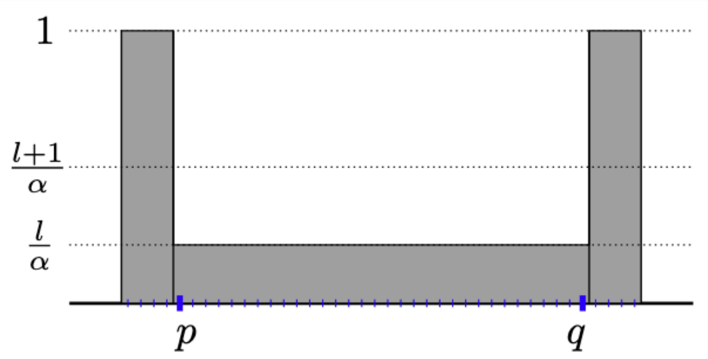
➡ Consider only algorithms that match immediately after k requests accumulate.

➡ Algorithm's behavior depends only on carries. ➡ **The number of candidates reduces to k.**
(since the number of carries never exceeds k)

The number of candidates is 3 with $k = 3$



Proposed Algorithm



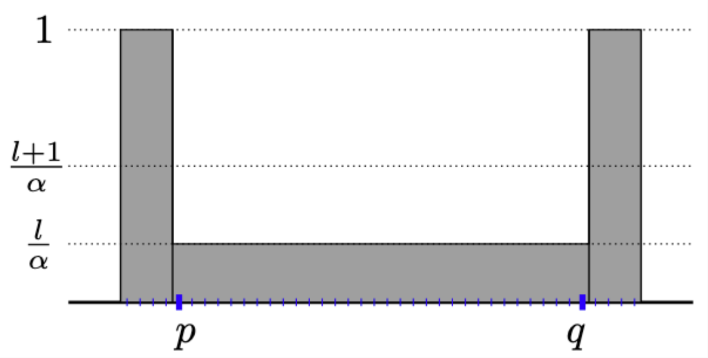
Ensure the waiting cost of the algorithm for each carry is at least 1

Manage variables $\ell, [p, q]$ for each phase:

- $\ell \in \{0, 1, \ldots, \alpha\}$: the waiting cost of any algorithm is at least ℓ/α.
- $[p, q] \subseteq \{0, 1, \ldots, k-1\}$: the waiting cost of algorithms with carries not in $[p, q]$ is at least 1.

Proposed Algorithm

Ensure the waiting cost of the algorithm for each carry is at least 1



Manage variables $\ell, [p, q]$ for each phase:

- $\ell \in \{0, 1, \ldots, \alpha\}$: the waiting cost of any algorithm is at least ℓ/α.
- $[p, q] \subseteq \{0, 1, \ldots, k-1\}$: the waiting cost of algorithms with carries not in $[p, q]$ is at least 1.

If $\ell \geq \alpha$ or $|[p, q]| = 0$, then the waiting cost of any algorithm is at least 1.

At the start of a phase, initialize:

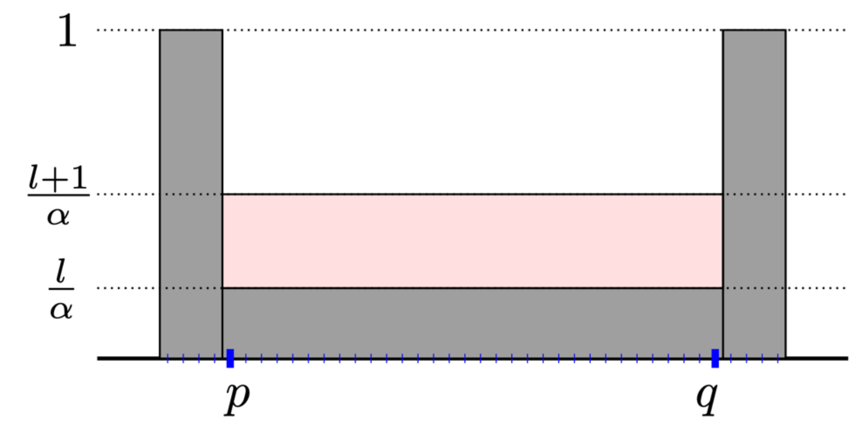
- $\ell = 0$,
- $[p, q] = [0, k-1]$.

Proposed Algorithm

Through the constant cost procedure, the variables are updated in either of the following ways:

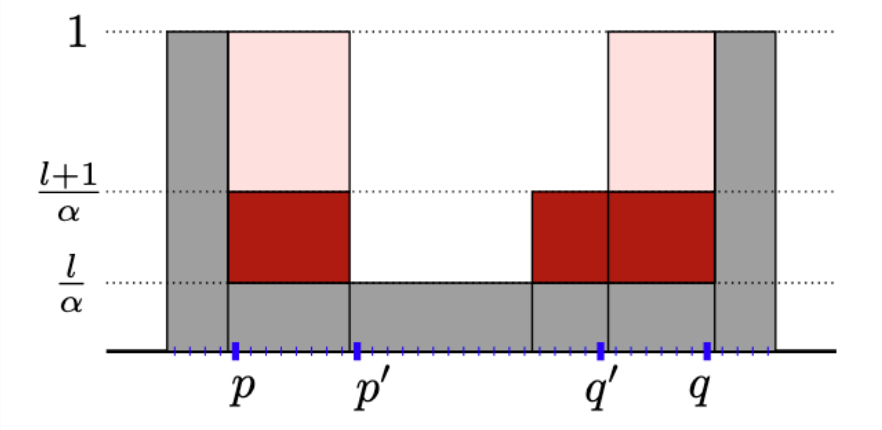
Minimum value increase：$\ell \to \ell + 1$

- The waiting cost increases by $1/\alpha$ for all carries.



Interval shrink：$[p, q] \to [p', q']$

- $[p', q'] \subset [p, q]$, $|[p', q']| \leq 2 \cdot |[p, q]|/\alpha$

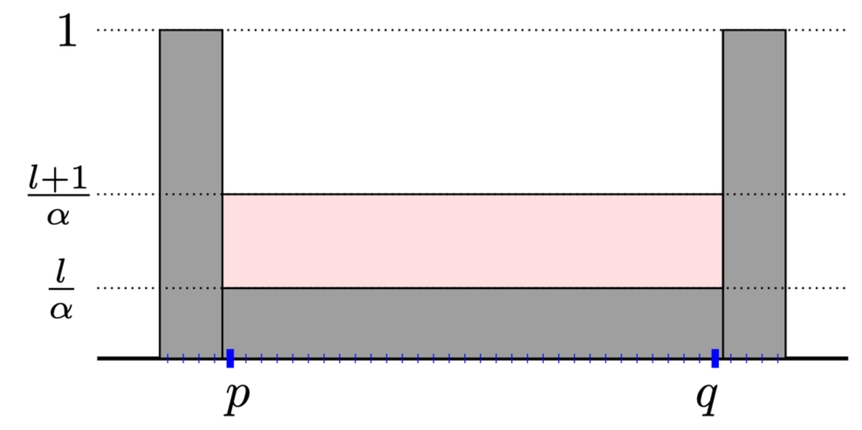


Proposed Algorithm

Through the constant cost procedure, the variables are updated in either of the following ways:

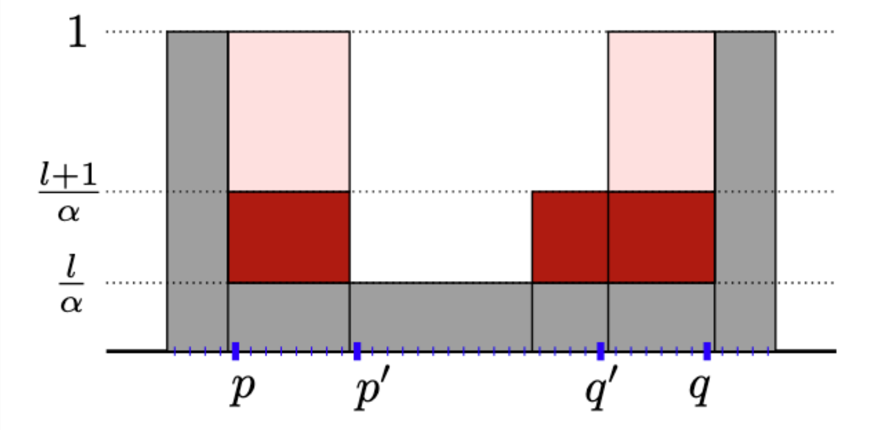
Minimum value increase：$\ell \to \ell + 1$

- The waiting cost increases by $1/\alpha$ for all carries.



Interval shrink：$[p, q] \to [p', q']$

- $[p', q'] \subset [p, q]$, $|[p', q']| \leq 2 \cdot |[p, q]|/\alpha$



After performing $O(\alpha)$ iterations, we have either $\ell \geq \alpha$ or $|[p, q]| = 0$.

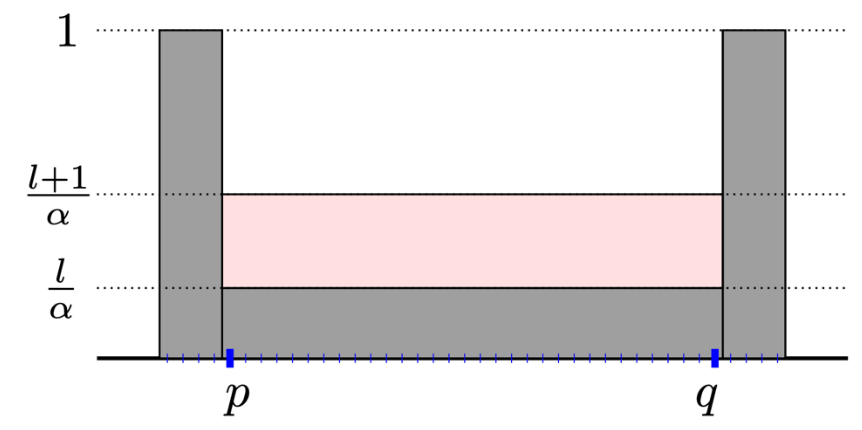
$\boxed{\alpha^\alpha = k,\ \alpha = \Theta(\log k\ /\ \log\log k)}$

Proposed Algorithm

Through the constant cost procedure, the variables are updated in either of the following ways:

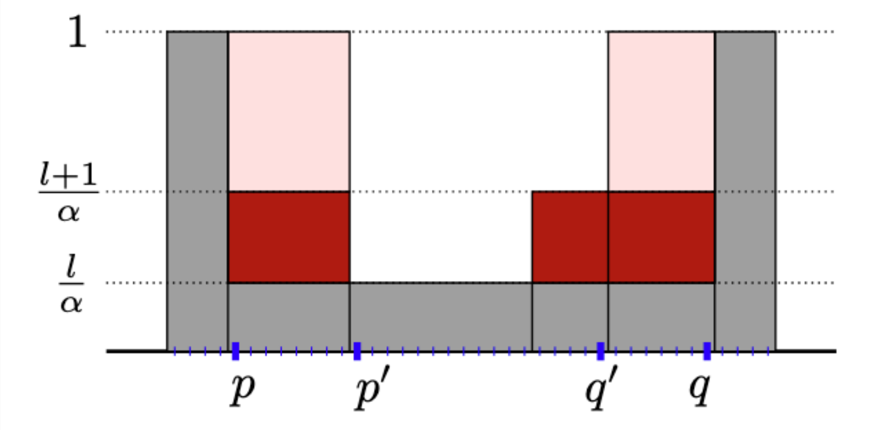
Minimum value increase：$\ell \to \ell + 1$

- The waiting cost increases by $1/\alpha$ for all carries.



Interval shrink：$[p, q] \to [p', q']$

- $[p', q'] \subset [p, q]$, $|[p', q']| \leq 2 \cdot |[p, q]|/\alpha$



After performing $O(\alpha)$ iterations, we have either $\ell \geq \alpha$ or $|[p, q]| = 0$.

$\boxed{\alpha^\alpha = k,\ \alpha = \Theta(\log k\ /\ \log\log k)}$

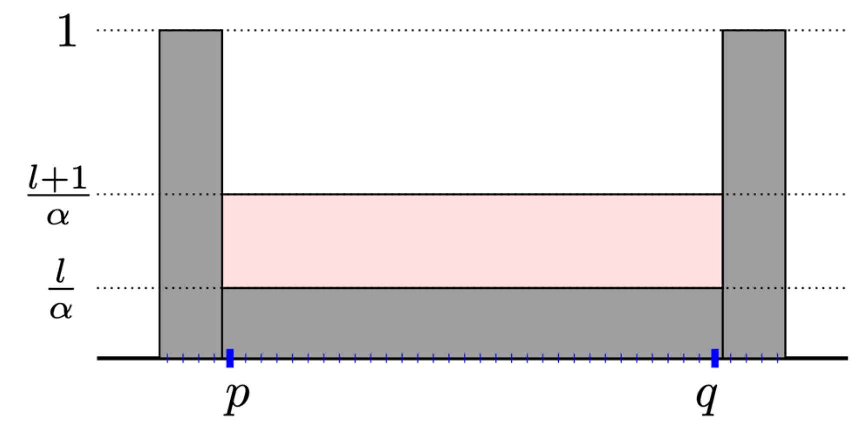
Cost per phase: $O(\alpha)$

Proposed Algorithm

Through the constant cost procedure, the variables are updated in either of the following ways:

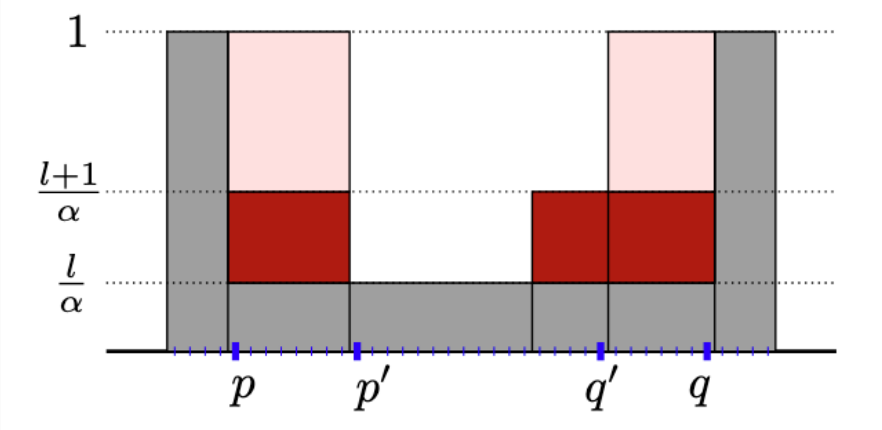
Minimum value increase：$\ell \rightarrow \ell + 1$

- The waiting cost increases by $1/\alpha$ for all carries.



Interval shrink：$[p, q] \rightarrow [p', q']$

- $[p', q'] \subset [p, q]$, $|[p', q']| \leq 2 \cdot |[p, q]|/\alpha$



After performing $O(\alpha)$ iterations, we have either $\ell \geq \alpha$ or $|[p, q]| = 0$.

$\boxed{\alpha^{\alpha} = k,\ \alpha = \Theta(\log k \ / \ \log \log k)}$

⇒ Cost per phase: $O(\alpha)$

⇒ Competitive ratio: $O(\log k / \log \log k)$

Conclusion

Penalty Function (with modification)	Competitive Ratio
(i) always 1	2 [Dooly et al. 2001]
(ii) 0 if the size is a multiple of k	$\boldsymbol{\Theta(\log k \,/\, \log\log k)}$
(iii) other scenarios	**unbounded**

- Our algorithm can be extended to penalty functions whose range is not $\{0, 1\}$.
  - For some reals $\mu < \lambda$, the range can be $\{0, \mu\}$ and $\{0\} \cup [\mu, \lambda]$.

- Future work
  - Introduce distance cost,
  - Introduce party: in the same party $\Rightarrow$ in the same match.

References

[Chen-Khatkar-Umboh 2022] Chen, Ryder, Jahanvi Khatkar, and Seeun William Umboh. 2022. “Online Weighted Cardinality Joint Replenishment Problem with Delay.” In *Proceedings of the 49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*, 229:40:1-40:18. LIPIcs.

[Dooly-Goldman-Scott 2001] Dooly, Daniel R., Sally A. Goldman, and Stephen D. Scott. 2001. “On-Line Analysis of the TCP Acknowledgment Delay Problem.” *Journal of the ACM* 48 (2): 243–73.

[Emek-Kutten-Wattenhofer 2016] Emek, Yuval, Shay Kutten, and Roger Wattenhofer. 2016. “Online Matching: Haste Makes Waste!” In *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing (STOC 2016)*, 333–44. STOC ’16.

[Mehta 2013] Mehta, Aranyak. 2013. “Online Matching and Ad Allocation.” *Foundations and Trends in Theoretical Computer Science* 8 (4): 265–368.

[Melnyk-Wang-Wattenhofer 2021] Melnyk, Darya, Yuyi Wang, and Roger Wattenhofer. 2021. “Online K-Way Matching with Delays and the H-Metric.” *arXiv [Cs.DS]*. arXiv. https://doi.org/10.48550/arXiv.2109.06640.

References

[Karp-UVazirani-VVazirani 1990] Karp, Richard M., Umesh V. Vazirani, and Vijay V. Vazirani. 1990. "An Optimal Algorithm for On-Line Bipartite Matching." In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing (STOC 1990)*, 352–58.

[Emek-Shapiro-Wang 2019] Yuval Emek, Yaacov Shapiro, and Yuyi Wang. Minimum cost perfect matching with delays for two sources. Theor. Comput. Sci., 754:122–129, 2019.

Appendix

Hardness of (ii) 0 if the size is a multiple of k

(i) When the penalty function is always 1 [Dooly-Sally-Stephen 2001]

The optimal online algorithm matches all remaining requests whenever it performs a match.

⇒ Only **the timing** of matches needs to be considered.

Hardness of (ii) 0 if the size is a multiple of k

(i) When the penalty function is always 1 [Dooly-Sally-Stephen 2001]

The optimal online algorithm matches all remaining requests whenever it performs a match.

Only **the timing** of matches needs to be considered.

(ii) When the penalty is $\mathbf{0}$ if the size is a multiple of $\boldsymbol{k}$

Any algorithm that matches all remaining requests whenever it performs a match
has a competitive ratio of $\Omega(\sqrt{k})$ (we prove this).

Both **the timing and size** of matches must be considered to obtain the competitive ratio of $O\left(\frac{\log k}{\log\log k}\right)$.