Universal Optimization for Non-Clairvoyant Subadditive Joint Replenishment

Tomer Ezra [1] Stefano Leonardi [2] **Michał Pawłowski** [2, 3, 4]
Matteo Russo [2] Seeun William Umboh [5]

[1]Harvard University

[2]Sapienza University of Rome

[3]University of Warsaw

[4]IDEAS NCBR

[5]University of Melbourne

Problem Statement

Joint Replenishment Problem (JRP) and its generalizations

Problem Statement

Joint Replenishment Problem (JRP) and its generalizations

- a **sequence** of requests that arrive over time

Problem Statement

Joint Replenishment Problem (JRP) and its generalizations

- a **sequence** of requests that arrive over time
- each request can be one of n request **types** U

Problem Statement

Joint Replenishment Problem (JRP) and its generalizations

- a **sequence** of requests that arrive over time
- each request can be one of n request **types** U
- cost of serving a set of requests is a **subadditive** function of their types, i.e., $f(A) + f(B) \geq f(A \cup B)$ for $A, B \subseteq U$

Problem Statement

Joint Replenishment Problem (JRP) and its generalizations

- a **sequence** of requests that arrive over time
- each request can be one of n request **types** U
- cost of serving a set of requests is a **subadditive** function of their types, i.e., $f(A) + f(B) \geq f(A \cup B)$ for $A, B \subseteq U$
- requests do not need to be served on arrival

Problem Statement

Joint Replenishment Problem (JRP) and its generalizations

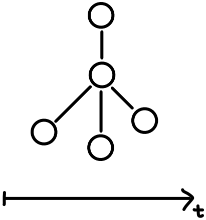
- a **sequence** of requests that arrive over time
- each request can be one of n request **types** U
- cost of serving a set of requests is a **subadditive** function of their types, i.e., $f(A) + f(B) \geq f(A \cup B)$ for $A, B \subseteq U$
- requests do not need to be served on arrival
- each request **accumulates a delay cost** while unserved

Problem Statement

Joint Replenishment Problem (JRP) and its generalizations

- a **sequence** of requests that arrive over time
- each request can be one of n request **types** U
- cost of serving a set of requests is a **subadditive** function of their types, i.e., $f(A) + f(B) \geq f(A \cup B)$ for $A, B \subseteq U$
- requests do not need to be served on arrival
- each request **accumulates a delay cost** while unserved

Goal: serve all requests **minimizing** the total service cost and delay cost

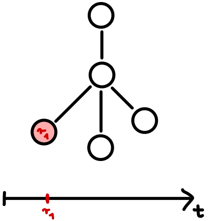


Problem Statement

Joint Replenishment Problem (JRP) and its generalizations

- a **sequence** of requests that arrive over time
- each request can be one of n request **types** U
- cost of serving a set of requests is a **subadditive** function of their types, i.e., $f(A) + f(B) \geq f(A \cup B)$ for $A, B \subseteq U$
- requests do not need to be served on arrival
- each request **accumulates a delay cost** while unserved

Goal: serve all requests **minimizing** the total service cost and delay cost

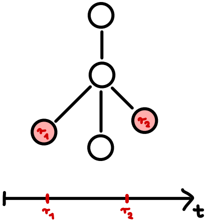


Problem Statement

Joint Replenishment Problem (JRP) and its generalizations

- a **sequence** of requests that arrive over time
- each request can be one of n request **types** U
- cost of serving a set of requests is a **subadditive** function of their types, i.e., $f(A) + f(B) \geq f(A \cup B)$ for $A, B \subseteq U$
- requests do not need to be served on arrival
- each request **accumulates a delay cost** while unserved

Goal: serve all requests **minimizing** the total service cost and delay cost

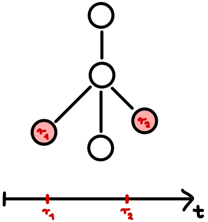


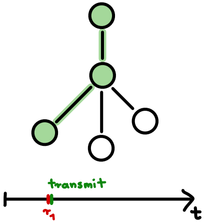
Problem Statement

Joint Replenishment Problem (JRP) and its generalizations

- a **sequence** of requests that arrive over time
- each request can be one of n request **types** U
- cost of serving a set of requests is a **subadditive** function of their types, i.e., $f(A) + f(B) \geq f(A \cup B)$ for $A, B \subseteq U$
- requests do not need to be served on arrival
- each request **accumulates a delay cost** while unserved

Goal: serve all requests **minimizing** the total service cost and delay cost



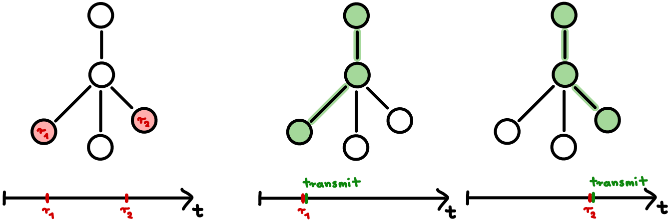


Problem Statement

Joint Replenishment Problem (JRP) and its generalizations

- a **sequence** of requests that arrive over time
- each request can be one of n request **types** U
- cost of serving a set of requests is a **subadditive** function of their types, i.e., $f(A) + f(B) \geq f(A \cup B)$ for $A, B \subseteq U$
- requests do not need to be served on arrival
- each request **accumulates a delay cost** while unserved

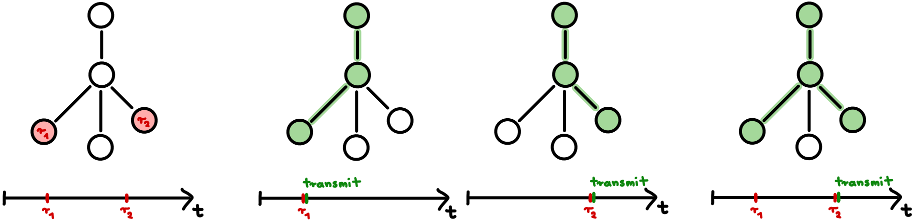
Goal: serve all requests **minimizing** the total service cost and delay cost



Joint Replenishment Problem (JRP) and its generalizations

- a **sequence** of requests that arrive over time
- each request can be one of n request **types** U
- cost of serving a set of requests is a **subadditive** function of their types, i.e., $f(A) + f(B) \geq f(A \cup B)$ for $A, B \subseteq U$
- requests do not need to be served on arrival
- each request **accumulates a delay cost** while unserved

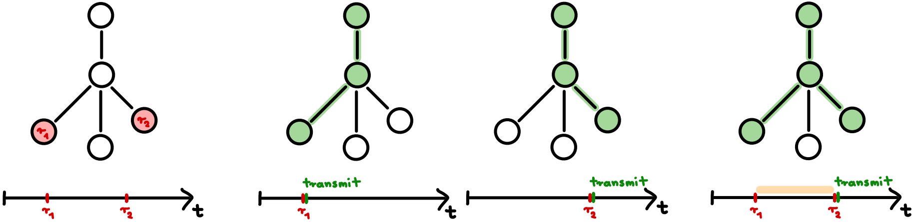
Goal: serve all requests **minimizing** the total service cost and delay cost



Joint Replenishment Problem (JRP) and its generalizations

- a **sequence** of requests that arrive over time
- each request can be one of n request **types** U
- cost of serving a set of requests is a **subadditive** function of their types, i.e., $f(A) + f(B) \geq f(A \cup B)$ for $A, B \subseteq U$
- requests do not need to be served on arrival
- each request **accumulates a delay cost** while unserved

Goal: serve all requests **minimizing** the total service cost and delay cost

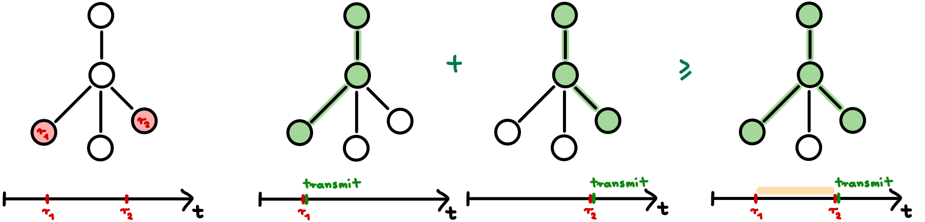


Problem Statement

Joint Replenishment Problem (JRP) and its generalizations

- a **sequence** of requests that arrive over time
- each request can be one of n request **types** U
- cost of serving a set of requests is a **subadditive** function of their types, i.e., $f(A) + f(B) \geq f(A \cup B)$ for $A, B \subseteq U$
- requests do not need to be served on arrival
- each request **accumulates a delay cost** while unserved

Goal: serve all requests **minimizing** the total service cost and delay cost



Clairvoyant vs Non-Clairvoyant:

Overview

Clairvoyant vs Non-Clairvoyant:

- most prior works on JRP, and its generalizations have focused on the **clairvoyant** setting (whole delay function known at arrival)

Overview

Clairvoyant vs Non-Clairvoyant:

- most prior works on JRP, and its generalizations have focused on the **clairvoyant** setting (whole delay function known at arrival)
- Touitou (ICALP 2023) developed a **non-clairvoyant** framework that provided an $O(\sqrt{n \log n})$ upper bound for a wide class of generalized JRP problems

Overview

Clairvoyant vs Non-Clairvoyant:

- most prior works on JRP, and its generalizations have focused on the **clairvoyant** setting (whole delay function known at arrival)
- Touitou (ICALP 2023) developed a **non-clairvoyant** framework that provided an $O(\sqrt{n \log n})$ upper bound for a wide class of generalized JRP problems

Our results:

Overview

Clairvoyant vs Non-Clairvoyant:

- most prior works on JRP, and its generalizations have focused on the **clairvoyant** setting (whole delay function known at arrival)
- Touitou (ICALP 2023) developed a **non-clairvoyant** framework that provided an $O(\sqrt{n \log n})$ upper bound for a wide class of generalized JRP problems

Our results:

- we provide a **simpler, modular framework** that matches the competitive ratio established by Touitou for the same class of generalized JRP

Overview

Clairvoyant vs Non-Clairvoyant:

- most prior works on JRP, and its generalizations have focused on the **clairvoyant** setting (whole delay function known at arrival)
- Touitou (ICALP 2023) developed a **non-clairvoyant** framework that provided an $O(\sqrt{n \log n})$ upper bound for a wide class of generalized JRP problems

Our results:

- we provide a **simpler, modular framework** that matches the competitive ratio established by Touitou for the same class of generalized JRP
- we obtain tight $O(\sqrt{n})$-competitive algorithms for two significant problems: **Multi-Level Aggregation** and **Weighted Symmetric Subadditive JRP**

Towards the Reduction to Disjoint TCP

Theorem [Jia et al., STOC 2005]

For every subadditive service function f, there is a **disjoint service function** g that $O(\sqrt{n \log n})$-**approximates** function f.

Towards the Reduction to Disjoint TCP

Theorem [Jia et al., STOC 2005]

For every subadditive service function f, there is a **disjoint service function** g that $O(\sqrt{n \log n})$-**approximates** function f.

1. function g **partitions** the universe U of request types into a **family of non-overlapping sets** $S_1, S_2, \ldots, S_k$ and given a set $S \subseteq U$ assigns it the cost of

$$g(S) = \sum_{i=1}^{k} f(S_i) \cdot \mathbb{1}\{S_i \cap S \neq \emptyset\}$$

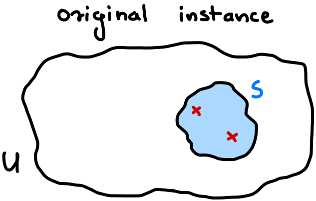
Towards the Reduction to Disjoint TCP

Theorem [Jia et al., STOC 2005]

For every subadditive service function f, there is a **disjoint service function** g that $O(\sqrt{n \log n})$-**approximates** function f.

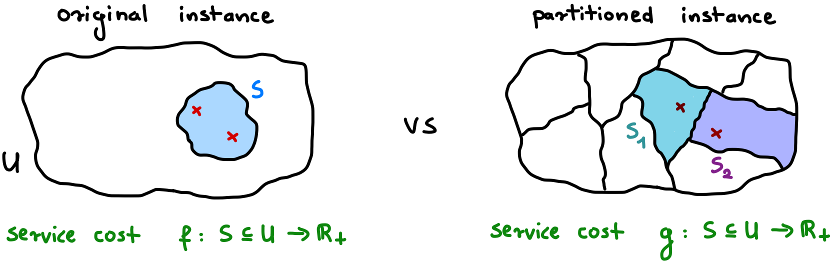
1. function g **partitions** the universe U of request types into a **family of non-overlapping sets** $S_1, S_2, \ldots, S_k$ and given a set $S \subseteq U$ assigns it the cost of
$$g(S) = \sum_{i=1}^{k} f(S_i) \cdot \mathbb{1}\{S_i \cap S \neq \emptyset\}$$
2. it holds that $f(S) \leq g(S) \leq \sqrt{n \log n} \cdot f(S)$ for every $S \subseteq U$

Subadditive JRP vs Disjoint TCP Acknowledgement

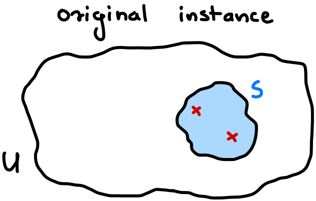


service cost $f: S \subseteq U \to \mathbb{R}_+$

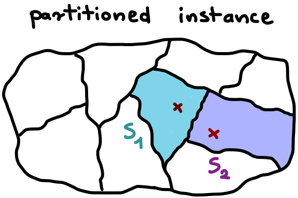
Subadditive JRP vs Disjoint TCP Acknowledgement



Subadditive JRP vs Disjoint TCP Acknowledgement



vs



service cost $f : S \subseteq U \to \mathbb{R}_+$

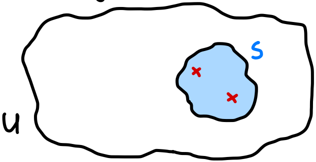
service cost $g : S \subseteq U \to \mathbb{R}_+$

$f(S)$

$g(S)$

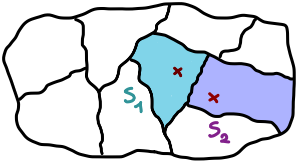
Subadditive JRP vs Disjoint TCP Acknowledgement

original instance



vs

partitioned instance



service cost $f : S \subseteq U \to \mathbb{R}_+$

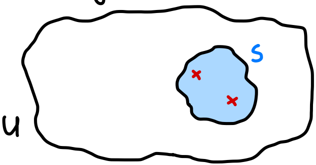
service cost $g : S \subseteq U \to \mathbb{R}_+$

$f(S)$

$$\underbrace{g(S)}_{=}$$

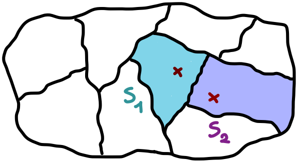
$$f(S_1) + f(S_2)$$

original instance



vs

partitioned instance



service cost $f : S \subseteq U \to \mathbb{R}_+$

service cost $g : S \subseteq U \to \mathbb{R}_+$

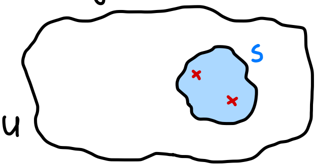
$f(S)$

$$f(S) \leqslant \underbrace{g(S)}_{= f(S_1) + f(S_2)} \leqslant \sqrt{n \log n}\, f(S)$$

Subadditive JRP vs Disjoint TCP Acknowledgement

Subadditive JRP

original instance



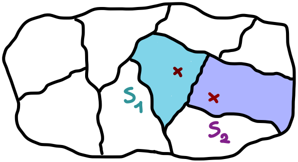
service cost $f : S \subseteq U \to \mathbb{R}_+$

$$f(S)$$

vs

Disjoint TCP Ack

partitioned instance

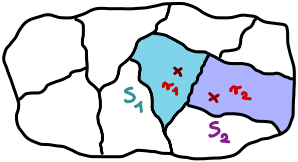


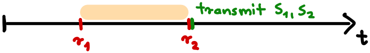
service cost $g : S \subseteq U \to \mathbb{R}_+$

$$f(S) \lesssim \underbrace{g(S)}_{= f(S_1) + f(S_2)} \lesssim \sqrt{n \log n}\, f(S)$$

Cost Comparison for Disjoint TCP Solutions

1. solution corresponding to Subadditive JRP (transmit S)



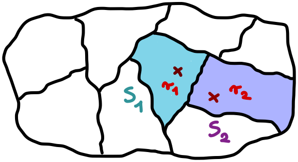


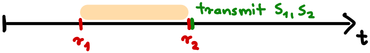
service cost: $g(S) = f(S_1) + f(S_2)$

delay cost: waiting of r_1

Cost Comparison for Disjoint TCP Solutions

① solution corresponding to Subadditive JRP (transmit S)

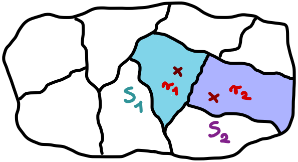


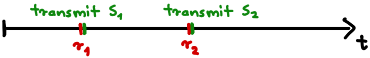


service cost: $g(S) = f(S_1) + f(S_2)$

delay cost: waiting of r_1

② optimal Disjoint TCP Acknowledgement solution





service cost: $g(S) = f(S_1) + f(S_2)$

delay cost: 0

Reduction Cost

Theorem [Jia et al., STOC 2005]

For every subadditive service function f, there is a **disjoint service function** g that $O(\sqrt{n \log n})$-**approximates** function f.

Reduction Cost

Theorem [Jia et al., STOC 2005]

For every subadditive service function f, there is a **disjoint service function** g that $O(\sqrt{n \log n})$-**approximates** function f.

Proposition

There is a **deterministic** 2-**competitive** algorithm for the Disjoint TCP Acknowledgement Problem.

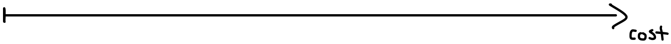
Reduction Cost

Theorem [Jia et al., STOC 2005]

For every subadditive service function f, there is a **disjoint service function** g that $O(\sqrt{n \log n})$-**approximates** function f.

Proposition

There is a **deterministic** 2-**competitive** algorithm for the Disjoint TCP Acknowledgement Problem.



Reduction Cost

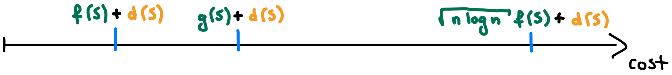
Theorem [Jia et al., STOC 2005]

For every subadditive service function f, there is a **disjoint service function** g that $O(\sqrt{n \log n})$-**approximates** function f.

Proposition

There is a **deterministic** 2-**competitive** algorithm for the Disjoint TCP Acknowledgement Problem.

S-optimal serving policy for Subadditive JRP



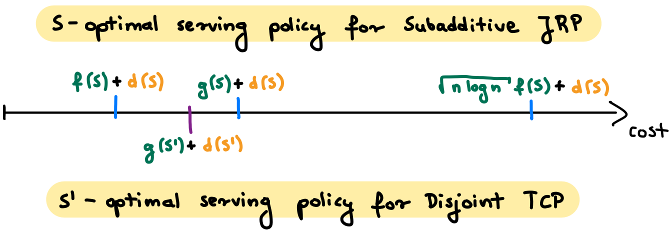
Reduction Cost

Theorem [Jia et al., STOC 2005]

For every subadditive service function f, there is a **disjoint service function** g that $O(\sqrt{n \log n})$-**approximates** function f.

Proposition

There is a **deterministic** 2-**competitive** algorithm for the Disjoint TCP Acknowledgement Problem.



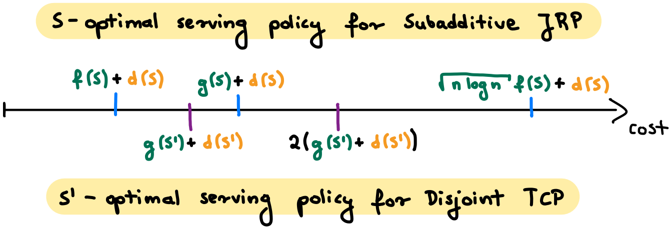
Reduction Cost

Theorem [Jia et al., STOC 2005]

For every subadditive service function f, there is a **disjoint service function** g that $O(\sqrt{n \log n})$-**approximates** function f.

Proposition

There is a **deterministic** 2-**competitive** algorithm for the Disjoint TCP Acknowledgement Problem.



Reduction Cost

Theorem [Jia et al., STOC 2005]

For every subadditive service function f, there is a **disjoint service function** g that $O(\sqrt{n \log n})$-**approximates** function f.

Proposition

There is a **deterministic** 2-**competitive** algorithm for the Disjoint TCP Acknowledgement Problem.

Theorem (Subadditive JRP)

There exists a **deterministic** $O(\sqrt{n \log n})$-**competitive** algorithm for **Non-Clairvoyant Subadditive JRP**.

Competitive Algorithm for TCP Acknowledgement

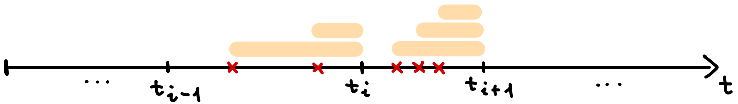
- consider a **greedy** algorithm that transmits S_i whenever waiting requests accumulate a **delay cost equal to the service cost** $f(S_i)$

Competitive Algorithm for TCP Acknowledgement

- consider a **greedy** algorithm that transmits S_i whenever waiting requests accumulate a **delay cost equal to the service cost** $f(S_i)$
- assume that this algorithm transmits S_i at times t_1, t_2, ..., t_l

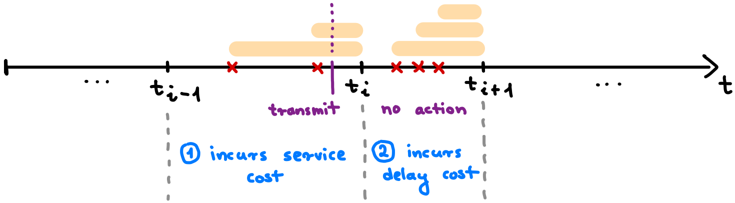
Competitive Algorithm for TCP Acknowledgement

- consider a **greedy** algorithm that transmits S_i whenever waiting requests accumulate a **delay cost equal to the service cost** $f(S_i)$
- assume that this algorithm transmits S_i at times $t_1, t_2, \ldots, t_l$
- then within **each interval** $[0, t_1], (t_1, t_2], \ldots, (t_{l-1}, t_l]$ optimal offline solution either incurs the **service cost** of $f(S_i)$ or the **delay cost** of the same value



Competitive Algorithm for TCP Acknowledgement

- consider a **greedy** algorithm that transmits S_i whenever waiting requests accumulate a **delay cost equal to the service cost** $f(S_i)$
- assume that this algorithm transmits S_i at times $t_1, t_2, \ldots, t_l$
- then within **each interval** $[0, t_1], (t_1, t_2], \ldots, (t_{l-1}, t_l]$ optimal offline solution either incurs the **service cost** of $f(S_i)$ or the **delay cost** of the same value



Improved Results

Theorem (Subadditive JRP)

There exists a **deterministic** $O(\sqrt{n \log n})$**-competitive** algorithm for **Non-Clairvoyant Subadditive JRP**.

Improved Results

Theorem (Subadditive JRP)

There exists a **deterministic** $O(\sqrt{n \log n})$-**competitive** algorithm for **Non-Clairvoyant Subadditive JRP**.

Reduction Lemma

If there exists a **disjoint** service function g that α-**approximates** f, then there exists a **non-clairvoyant** 2α-**competitive** algorithm for every Subadditive JRP instance with service cost function f.

Improved Results

Theorem (Subadditive JRP)

There exists a **deterministic** $O(\sqrt{n \log n})$-**competitive** algorithm for **Non-Clairvoyant Subadditive JRP**.

Reduction Lemma

If there exists a **disjoint** service function g that α-**approximates** f, then there exists a **non-clairvoyant** 2α-**competitive** algorithm for every Subadditive JRP instance with service cost function f.

Can we achieve **better competitiveness** for some **subproblems**?

Improved Results

Theorem (Subadditive JRP)

There exists a **deterministic** $O(\sqrt{n \log n})$-**competitive** algorithm for **Non-Clairvoyant Subadditive JRP**.

Reduction Lemma

If there exists a **disjoint** service function g that α-**approximates** f, then there exists a **non-clairvoyant** 2α-**competitive** algorithm for every Subadditive JRP instance with service cost function f.

Can we achieve **better competitiveness** for some **subproblems**?

Theorem (Multi-Level Aggregation)

There exists an efficient deterministic $O(\sqrt{n})$-competitive algorithm for the Non-Clairvoyant **Multi-Level Aggregation** problem.

Multi-Level Aggregation

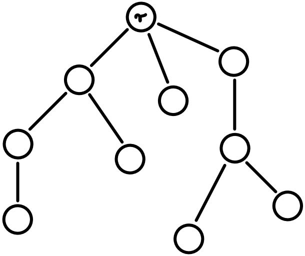
- the service function f is defined by a rooted **aggregation tree** T, where each **node** corresponds to a different **request type**

Multi-Level Aggregation

- the service function f is defined by a rooted **aggregation tree** T, where each **node** corresponds to a different **request type**
- let r be the root of T and let $c(v)$ be the cost of node v for $v \in T$

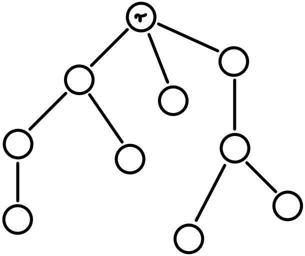
Multi-Level Aggregation

- the service function f is defined by a rooted **aggregation tree** T, where each **node** corresponds to a different **request type**
- let r be the root of T and let $c(v)$ be the cost of node v for $v \in T$
- for a subset V of nodes, $f(V)$ is defined to be the total cost of the nodes in the **minimal subtree** connecting V to r

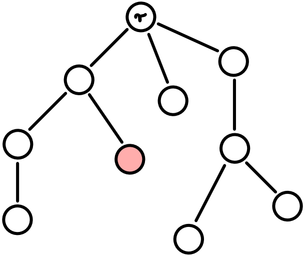


aggregation tree

- the service function f is defined by a rooted **aggregation tree** T, where each **node** corresponds to a different **request type**
- let r be the root of T and let $c(v)$ be the cost of node v for $v \in T$
- for a subset V of nodes, $f(V)$ is defined to be the total cost of the nodes in the **minimal subtree** connecting V to r



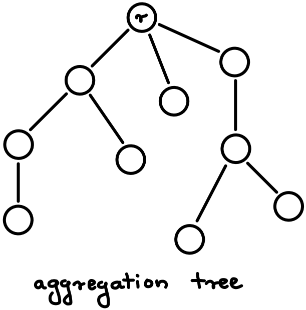
aggregation tree

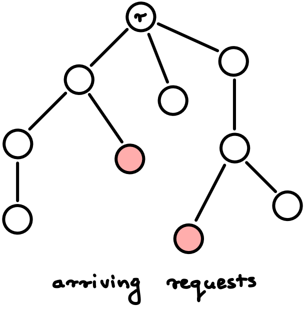


arriving requests

Multi-Level Aggregation

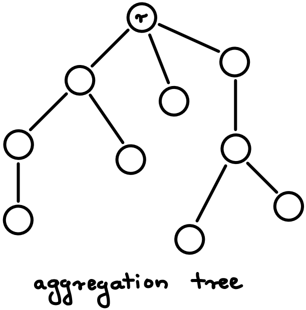
- the service function f is defined by a rooted **aggregation tree** T, where each **node** corresponds to a different **request type**
- let r be the root of T and let $c(v)$ be the cost of node v for $v \in T$
- for a subset V of nodes, $f(V)$ is defined to be the total cost of the nodes in the **minimal subtree** connecting V to r

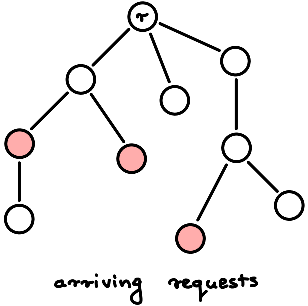




Multi-Level Aggregation

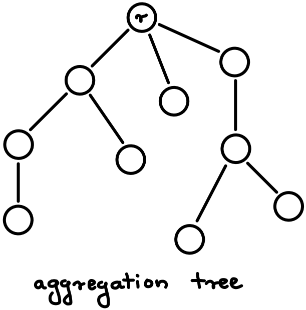
- the service function f is defined by a rooted **aggregation tree** T, where each **node** corresponds to a different **request type**
- let r be the root of T and let $c(v)$ be the cost of node v for $v \in T$
- for a subset V of nodes, $f(V)$ is defined to be the total cost of the nodes in the **minimal subtree** connecting V to r

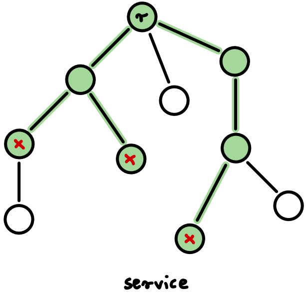




Multi-Level Aggregation

- the service function f is defined by a rooted **aggregation tree** T, where each **node** corresponds to a different **request type**
- let r be the root of T and let $c(v)$ be the cost of node v for $v \in T$
- for a subset V of nodes, $f(V)$ is defined to be the total cost of the nodes in the **minimal subtree** connecting V to r





Service Cost Comparison

Goal: minimize the cost of serving set V of request types (nodes)

Service Cost Comparison

Goal: minimize the cost of serving set V of request types (nodes)

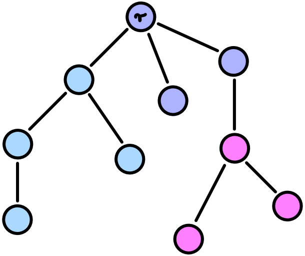
Optimal solution: cost $f(V)$ of the minimum spanning tree connecting V to the root r

Service Cost Comparison

Goal: minimize the cost of serving set V of request types (nodes)

Optimal solution: cost $f(V)$ of the minimum spanning tree connecting V to the root r

Disjoint approach: partition all the nodes U in T into a family of disjoint subsets $T_1, T_2, \ldots, T_k$; pay $\sum_{i=1}^{k} f(T_i) \cdot \mathbb{1}\{T_i \cap V \neq \emptyset\}$

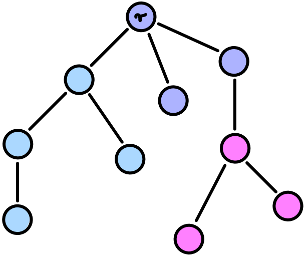


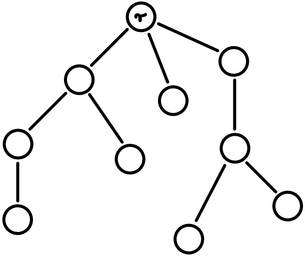
Service Cost Comparison

Goal: minimize the cost of serving set V of request types (nodes)

Optimal solution: cost $f(V)$ of the minimum spanning tree connecting V to the root r

Disjoint approach: partition all the nodes U in T into a family of disjoint subsets $T_1, T_2, \ldots, T_k$; pay $\sum_{i=1}^{k} f(T_i) \cdot \mathbb{1}\{T_i \cap V \neq \emptyset\}$



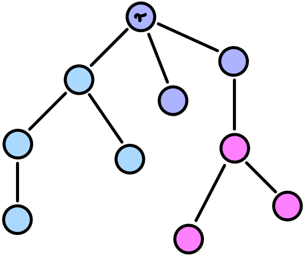


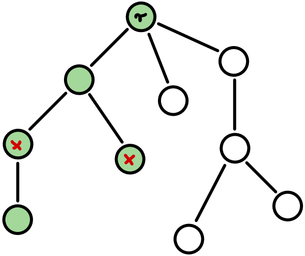
Service Cost Comparison

Goal: minimize the cost of serving set V of request types (nodes)

Optimal solution: cost $f(V)$ of the minimum spanning tree connecting V to the root r

Disjoint approach: partition all the nodes U in T into a family of disjoint subsets $T_1, T_2, \ldots, T_k$; pay $\sum_{i=1}^{k} f(T_i) \cdot \mathbb{1}\{T_i \cap V \neq \emptyset\}$



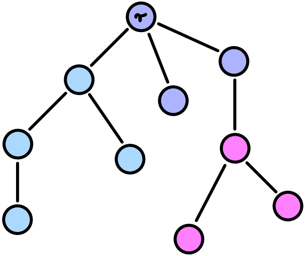


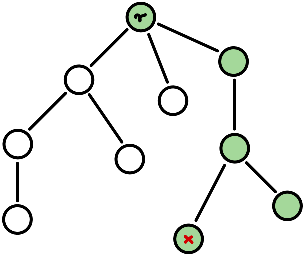
Service Cost Comparison

Goal: minimize the cost of serving set V of request types (nodes)

Optimal solution: cost $f(V)$ of the minimum spanning tree connecting V to the root r

Disjoint approach: partition all the nodes U in T into a family of disjoint subsets $T_1, T_2, \ldots, T_k$; pay $\sum_{i=1}^{k} f(T_i) \cdot \mathbb{1}\{T_i \cap V \neq \emptyset\}$



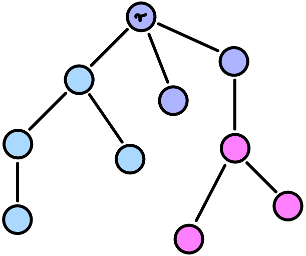


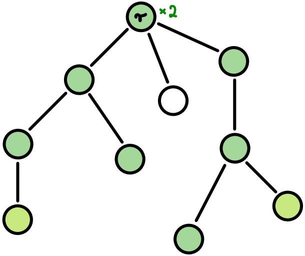
Service Cost Comparison

Goal: minimize the cost of serving set V of request types (nodes)

Optimal solution: cost $f(V)$ of the minimum spanning tree connecting V to the root r

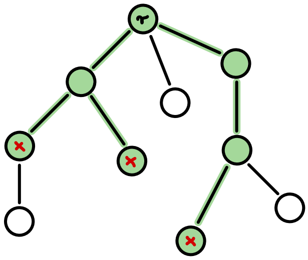
Disjoint approach: partition all the nodes U in T into a family of disjoint subsets $T_1, T_2, \ldots, T_k$; pay $\sum_{i=1}^{k} f(T_i) \cdot \mathbb{1}\{T_i \cap V \neq \emptyset\}$





Heavy Clusters

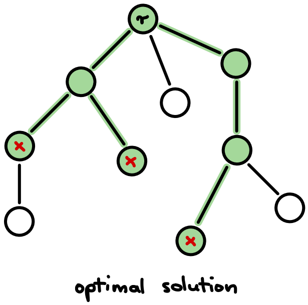
Goal: find a node v for which both the cost of its **subtree** and the cost of the **path** leading to the root are at most $O(\sqrt{n})c(v)$

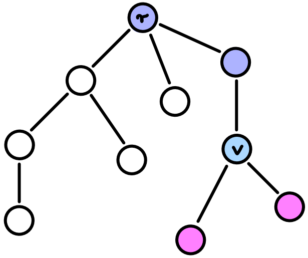


optimal solution

Heavy Clusters

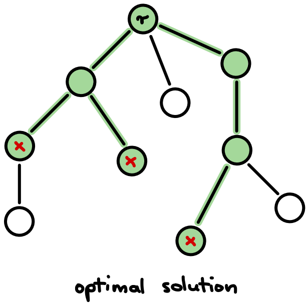
Goal: find a node v for which both the cost of its **subtree** and the cost of the **path** leading to the root are at most $O(\sqrt{n})c(v)$

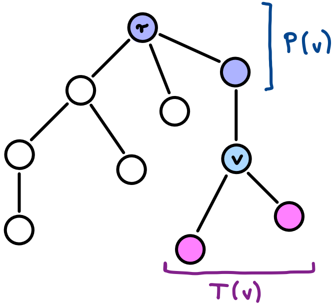




Heavy Clusters

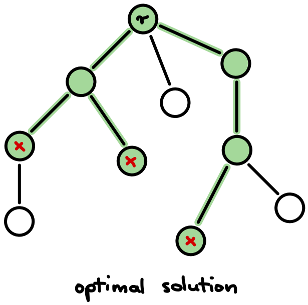
Goal: find a node v for which both the cost of its **subtree** and the cost of the **path** leading to the root are at most $O(\sqrt{n})c(v)$

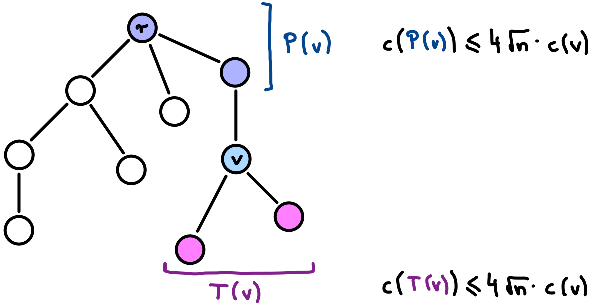




Heavy Clusters

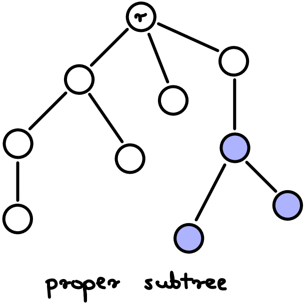
Goal: find a node v for which both the cost of its **subtree** and the cost of the **path** leading to the root are at most $O(\sqrt{n})c(v)$





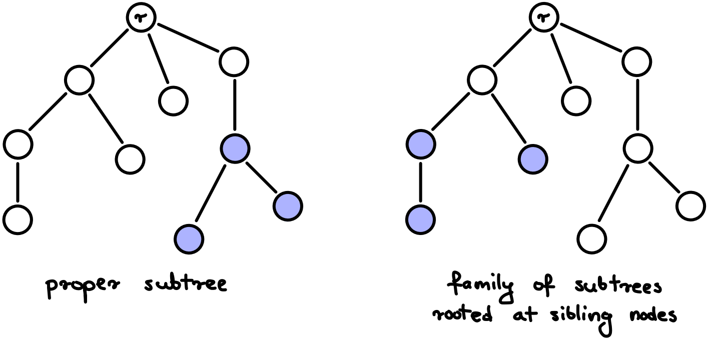
Light Clusters

Goal: find a node v which **subtree** is of size roughly $\sqrt{n}$

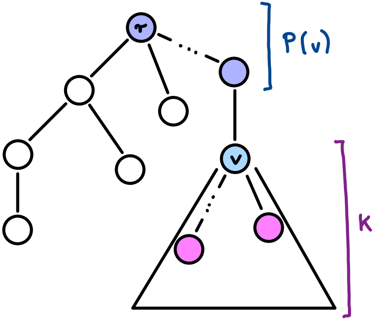


Light Clusters

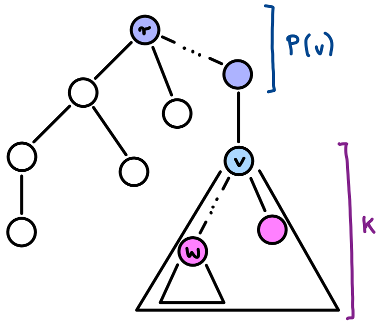
Goal: find a node v which **subtree** is of size roughly $\sqrt{n}$



Subtree Cost vs Path Cost

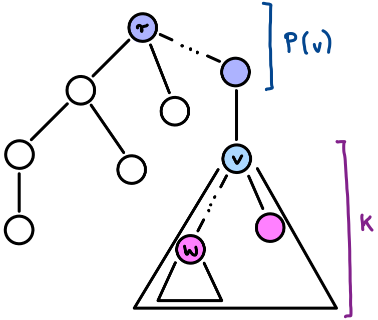


Subtree Cost vs Path Cost



take the heaviest node $w \in K$

Subtree Cost vs Path Cost

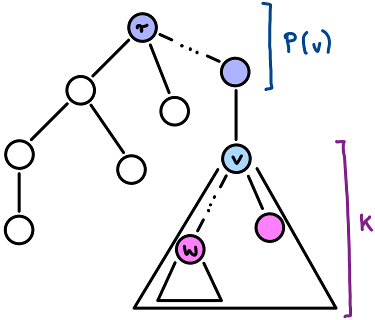


take the heaviest node $w \in K$

$$c(w) \geq \frac{c(K)}{2\sqrt{n}}$$

$$2\sqrt{n} \cdot c(w) \geq c(K)$$

Subtree Cost vs Path Cost



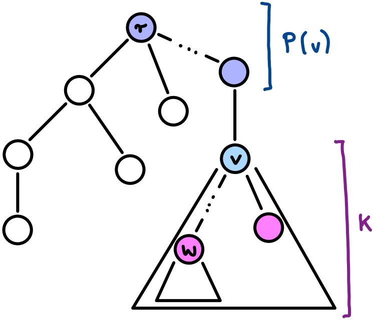
take the heaviest node $w \in K$

$$c(w) \geq \frac{c(K)}{2\sqrt{n}}$$

$$2\sqrt{n} \cdot c(w) \geq c(K)$$

assume: $c(P(v)) < c(K)$

Subtree Cost vs Path Cost



take the heaviest node $w \in K$

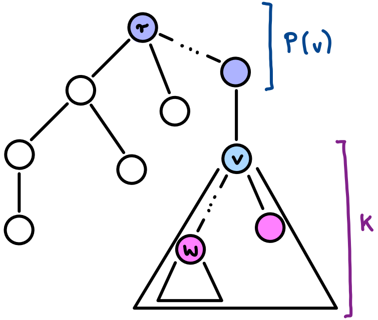
$$c(w) \geq \frac{c(K)}{2\sqrt{n}}$$

$$2\sqrt{n} \cdot c(w) \geq c(K)$$

assume: $c(P(v)) < c(K)$

$$c(P(w)) \leq c(P(v)) + c(K)$$

Subtree Cost vs Path Cost



take the heaviest node $w \in K$

$$c(w) \geq \frac{c(K)}{2\sqrt{n}}$$

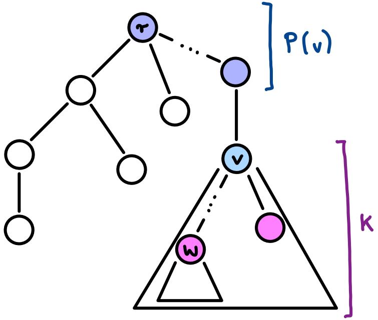
$$2\sqrt{n} \cdot c(w) \geq c(K)$$

assume: $c(P(v)) < c(K)$

$$c(P(w)) \leq c(P(v)) + c(K)$$

$$\leq 2c(K)$$

Subtree Cost vs Path Cost



take the heaviest node $w \in K$

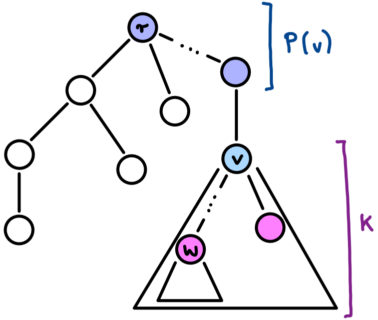
$$c(w) \geq \frac{c(K)}{2\sqrt{n}}$$

$$2\sqrt{n} \cdot c(w) \geq c(K)$$

assume: $c(P(v)) < c(K)$

$$c(P(w)) \leq c(P(v)) + c(K)$$
$$\leq 2c(K)$$
$$\leq 4\sqrt{n} \cdot c(K)$$

Subtree Cost vs Path Cost



take the heaviest node $w \in K$

$$c(w) \geq \frac{c(K)}{2\sqrt{n}}$$

$$2\sqrt{n} \cdot c(w) \geq c(K)$$

assume: $c(P(v)) < c(K)$

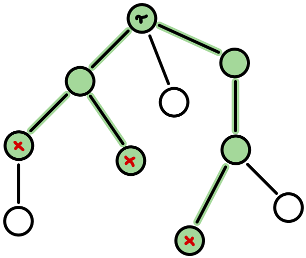
$$\begin{aligned} c(P(w)) &\leq c(P(v)) + c(K) \\ &\leq 2c(K) \\ &\leq 4\sqrt{n} \cdot c(K) \end{aligned}$$

thus: $c(P(v)) \geq c(K)$

Approximation Factor Analysis

Theorem (Multi-Level Aggregation)

For any MLA service function f, there exists a **disjoint** service function g that $O(\sqrt{n})$-**approximates** f. It can be found in time **polynomial** with respect to the MLA instance defining f.

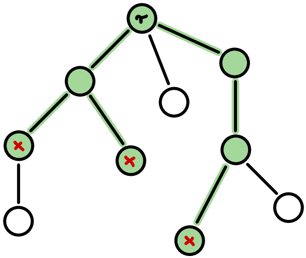


budget: $\sqrt{n} \cdot f(\,)$

Approximation Factor Analysis

Theorem (Multi-Level Aggregation)

For any MLA service function f, there exists a **disjoint** service function g that $O(\sqrt{n})$-**approximates** f. It can be found in time **polynomial** with respect to the MLA instance defining f.



budget: $\sqrt{n} \cdot f(\bullet)$

heavy cluster:

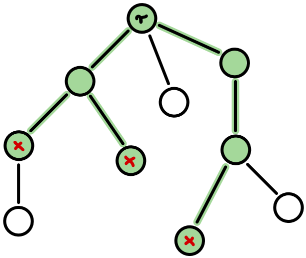
take heavy node v from $\bullet$

pay at most $8\sqrt{n} \cdot c(v)$ for the subtree and path

Approximation Factor Analysis

Theorem (Multi-Level Aggregation)

For any MLA service function f, there exists a **disjoint** service function g that $O(\sqrt{n})$-**approximates** f. It can be found in time **polynomial** with respect to the MLA instance defining f.



budget: $\sqrt{n} \cdot f( )$

heavy cluster:

take heavy node v from

pay at most $8\sqrt{n} \cdot c(v)$ for the subtree and path

light cluster:

at most $\sqrt{n}$ of them

$f(\text{path}) \geq f(\text{subtree})$

Conclusions

Our results

- we provide a **simpler, modular framework** that matches the competitive ratio established by Touitou for the same class of generalized JRP
- we obtain tight $O(\sqrt{n})$-competitive algorithms for two significant problems: **Multi-Level Aggregation** and **Weighted Symmetric Subadditive JRP**

Thank you!