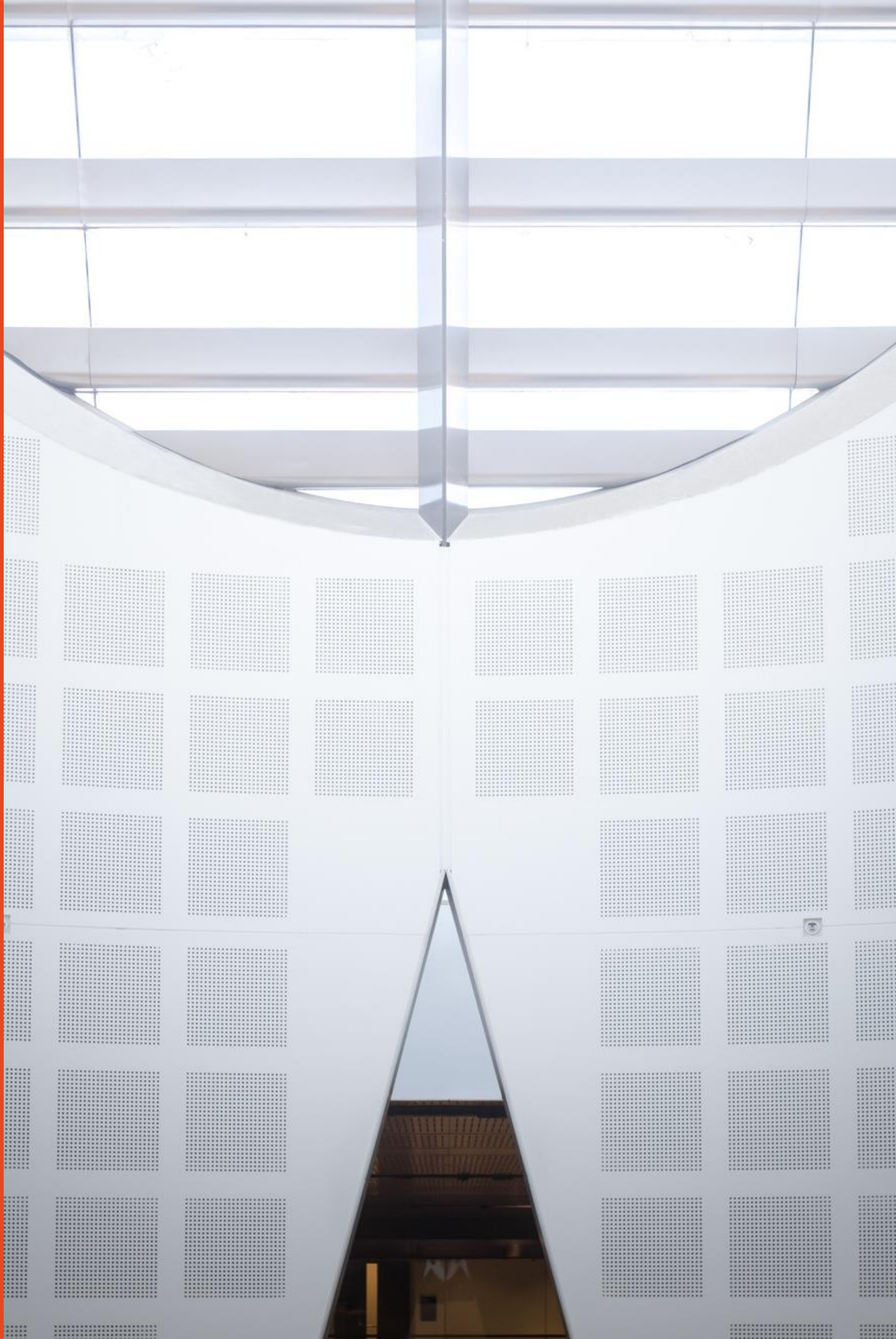
Online Cardinality Joint Replenishment Problem with Delay

Presented by

Ryder Chen

THE UNIVERSITY OF SYDNEY

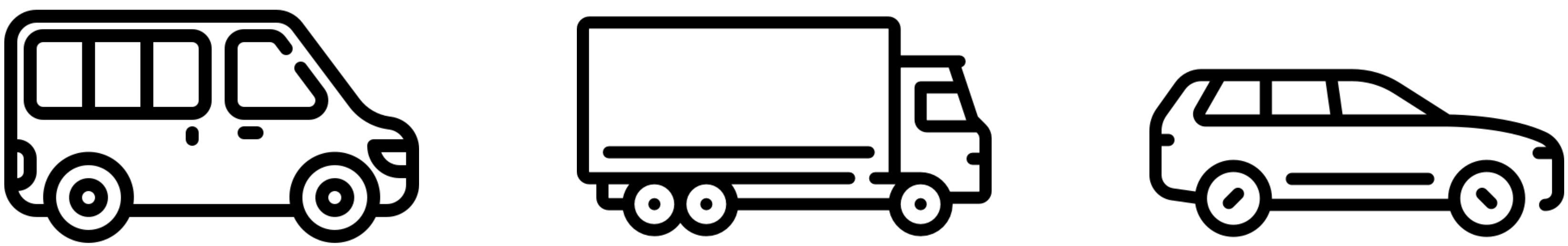


Problem Definition

- What is the Cardinality Joint Replenishment Problem?
- What is Delay?
- What is an Online Algorithm?

Cardinality Joint Replenishment Problem (JRP)

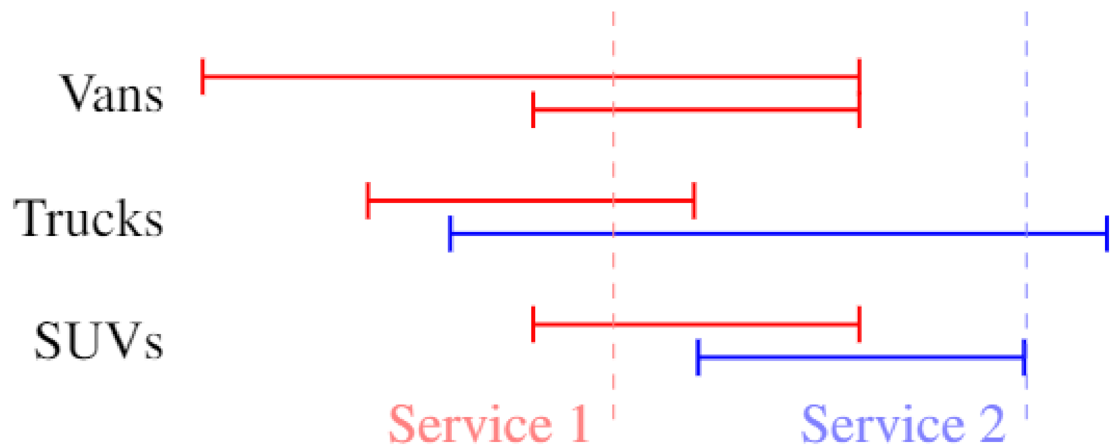
- “Generalisation” of Classical JRP
- Car factory that produces Vans, Trucks and SUVs



- Each car type requires specialised machinery
- Production cost dominated by machinery hiring cost
- The cost of hiring n machines is $f(n)$

e.g. Making 3 Vans and 1 Truck costs $f(2)$

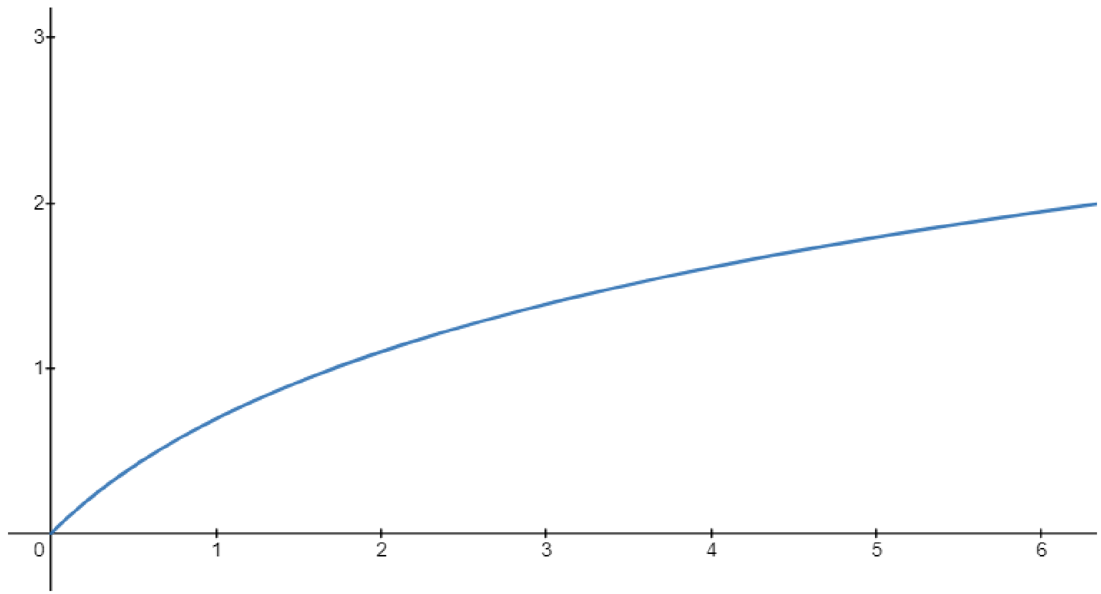
- Customers will request cars from the factory
- Each request will have a deadline
- Problem: The factory must decide when to serve each request whilst minimising costs



- Service 1 costs $f(3)$, Service 2 costs $f(2)$

What is f?

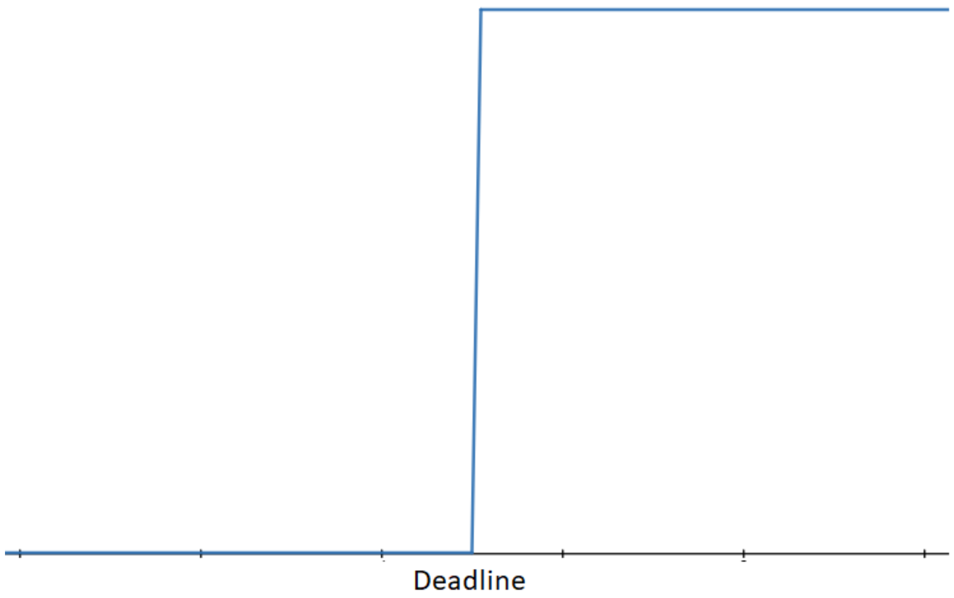
- Non-decreasing concave function



- Every additional piece of machinery hired is successively cheaper

Delay

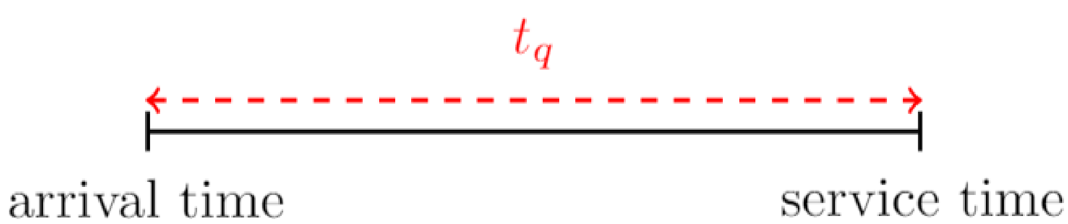
- Incur a penalty depending on how long a request is left unserved
- Delay generalises deadlines



- Delay is a better representation of reality

Delay

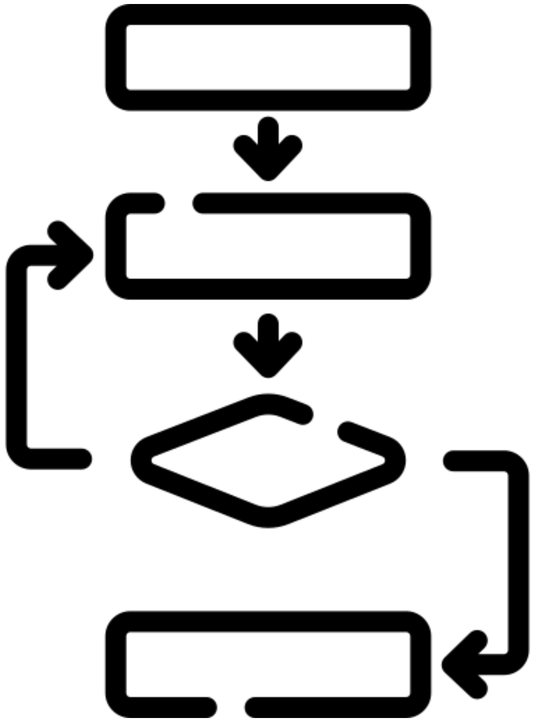
- Each request q has a delay penalty $d_q(t)$
  - arbitrary, non-decreasing, continuous function of time
  - $d_q(t) \rightarrow \infty$ as $t \rightarrow \infty$
- Serving q incurs a penalty of $d_q(t_q)$



Goal: Minimise Ordering + Delay costs

Online Cardinality JRP with Delay

- Offline Algorithm: All information given as input
- Online Algorithm: Runs in real time. Only has information up to the current time.
- Cannot change past decisions.
- No knowledge of the future.

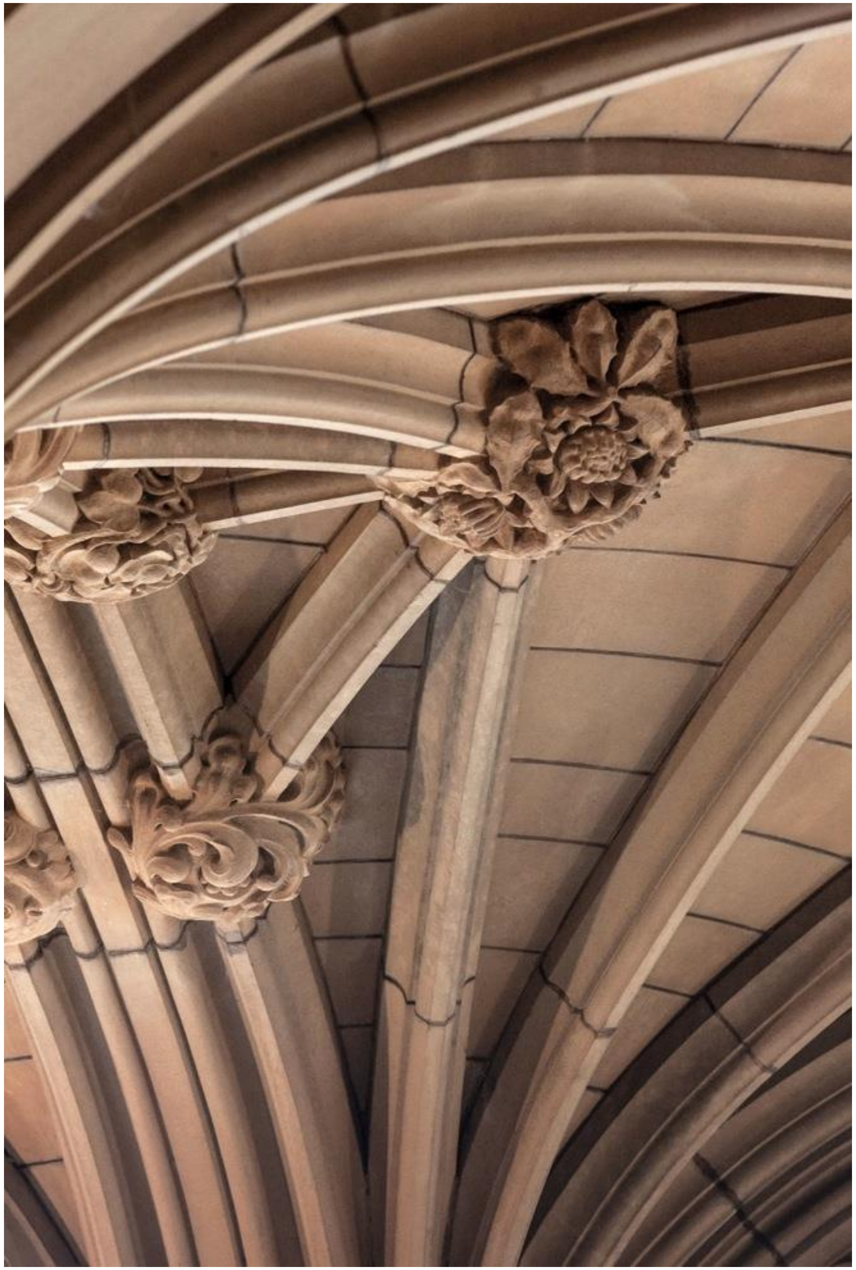


Evaluating Online Algorithms

- Competitive Ratio
- How many times worse is the online solution compared to the optimal, offline solution?

e.g. If $ALG \leq 2 \times OPT$ then the algorithm is 2-competitive OR $O(1)$-competitive

Background Literature



Offline JRP Literature

Problem	Result	Reference
Classical JRP	NP-Hard	Becchetti et. al. 2009
Classical JRP	1.574-approx. algorithm	Bienkowski et. al. 2013
Cardinality JRP	5-approx. algorithm	Cheung et. al. 2015

Online JRP Literature

	Deadlines	Delay
Classical JRP	3-competitive (Buchbinder et. al. 2008)	3-competitive (Buchbinder et. al. 2008)
Cardinality JRP	$O(1)$-competitive (Khatkar J. 2020)	

Gap in Literature

- Limited understanding of online JRP
- $O(1)$-competitive algorithm for Cardinality JRP with Deadlines but what about delay?

	Deadlines	Delay
Classical JRP	3-competitive (Buchbinder et. al. 2008)	3-competitive (Buchbinder et. al. 2008)
Cardinality JRP	$O(1)$-competitive (Khatkar J. 2020)	?

GOAL: Develop a $O(1)$-competitive algorithm for Online Cardinality JRP with Delay

Limitations of Existing Work

– Unclear how to generalise Khatkar's algorithm for Cardinality JRP with Deadlines

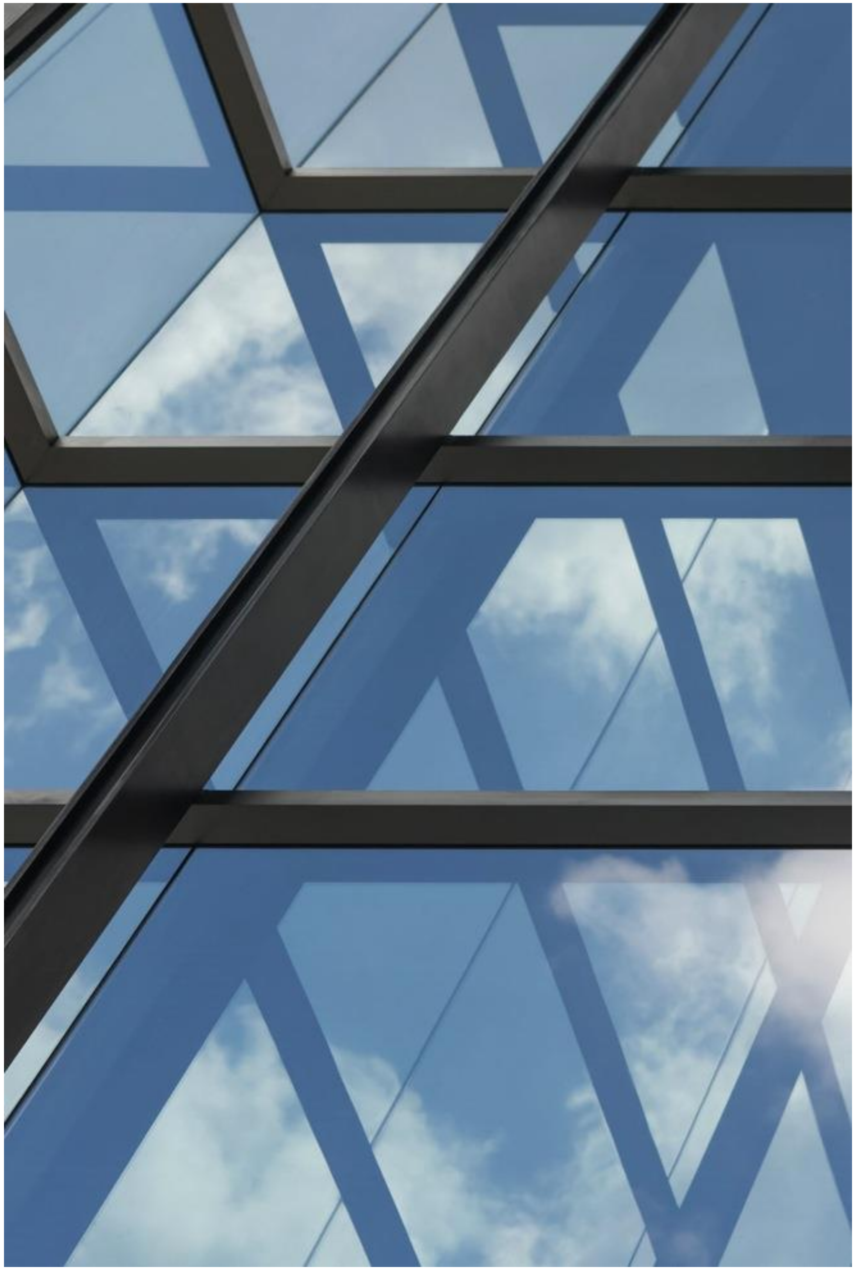
– Unclear how to generalise other works on JRP and related problems

– Problem is non-trivial and will require innovation and extensions of past work

Results

	Deadlines	Delay
Classical JRP	3-competitive (Buchbinder et. al. 2008)	3-competitive (Buchbinder et. al. 2008)
Cardinality JRP	$O(1)$-competitive (Khatkar J. 2020)	$O(1)$-competitive (My Result)

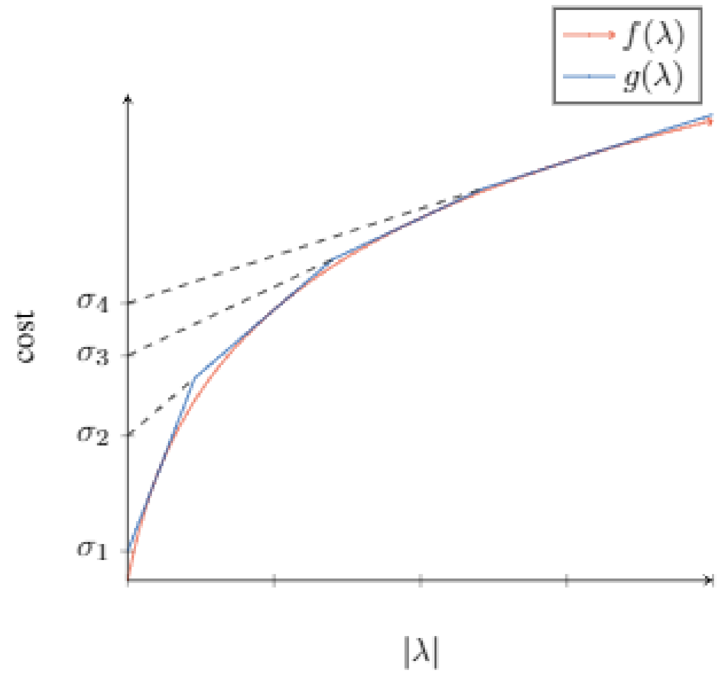
The Algorithm



THE UNIVERSITY OF SYDNEY

A piecewise reduction

- Approximate $f(x)$ using a piecewise affine function $g(x) = \min\{\sigma_l + \delta_l x : l \in [1, n]\}$
- $f(x) \leq g(x) \leq O(1) \times f(x)$ (Guha et. al. 2001)
- Constant competitive solutions under $g(x)$ are also constant competitive under $f(x)$



- Solution needs to specify which piece of $g(x)$ each service uses as its cost function

Algorithm challenges

1) **When** do we serve?

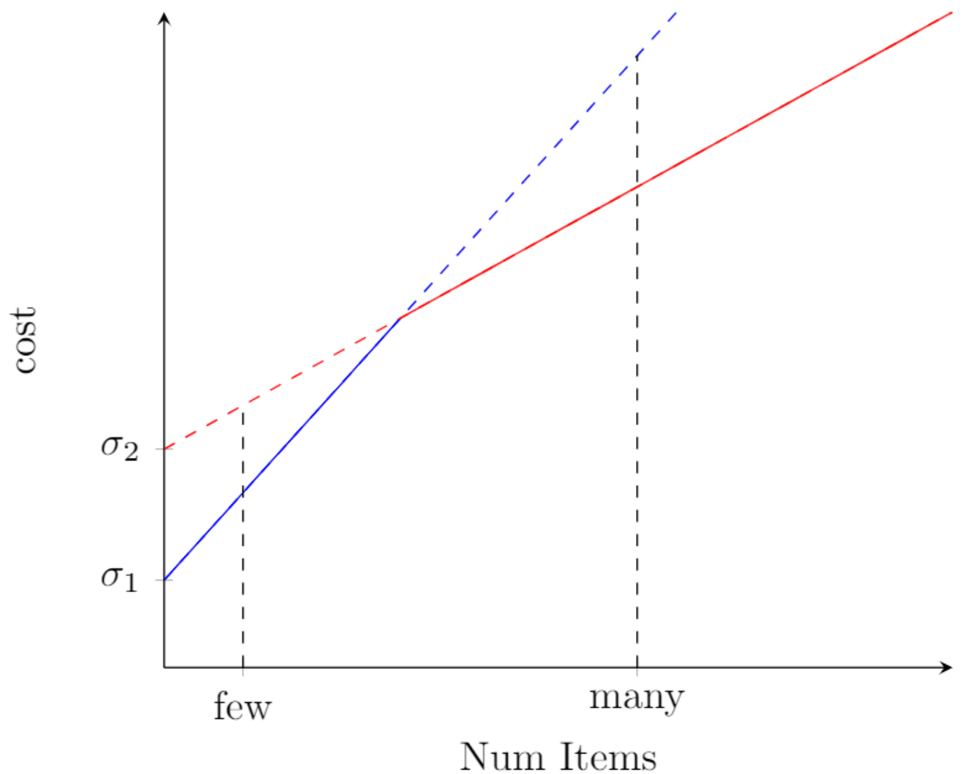
- Want to minimise delay
- No deadlines so how much delay is too much?

2) **What** do we serve?

- Ordering in bulk is cheaper
- Serving requests on the same item type incurs no extra cost

3) **How** much to serve?

- Which piece of the piecewise function do we serve with?



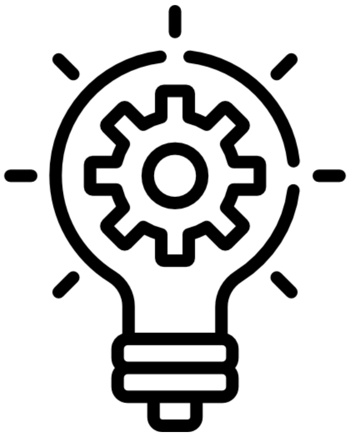
Key Contribution 1: Extended Investments

- Traditionally, services pay for the delay on requests it serves
- Investments introduced by Azar and Touitou (2020)
- Instead, pay for delay on requests even on those not served
- Pay for delay accumulated in the future
- Delay costs = Investment costs

Key Contribution 1: Extended Investments

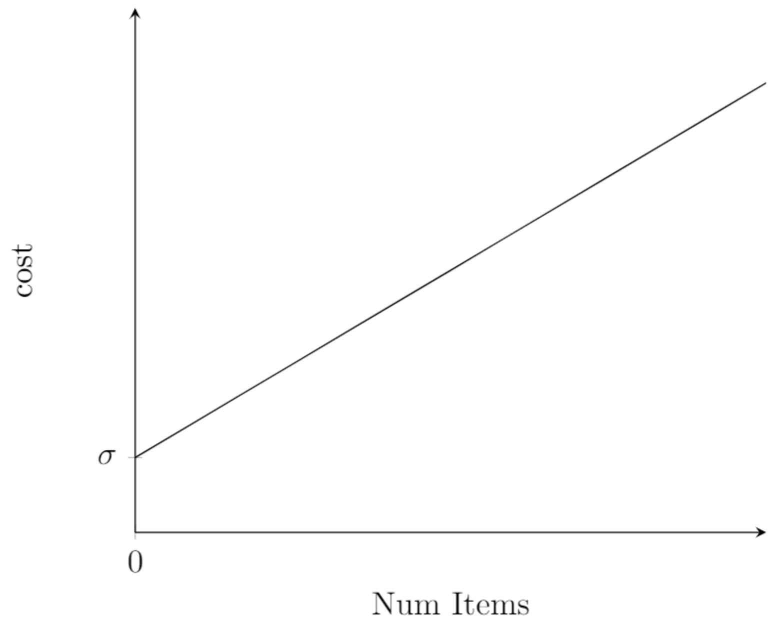
- Azar and Touitou (2020) do not keep track of and use how much has been invested
- **My Innovation:** Keep track of amount invested and serve requests when this reaches a threshold

- This innovation leads to a novel algorithm analysis technique



A specific instance

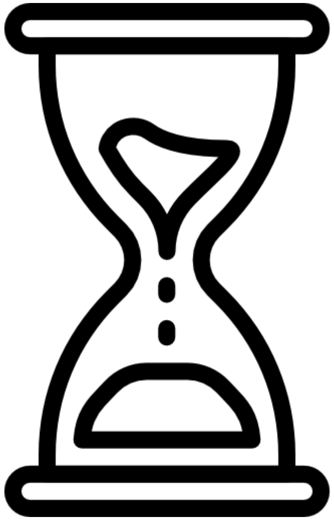
– Assume the cost function $g(x) = \sigma + \delta x$



– Algorithm does not need to decide how much to serve

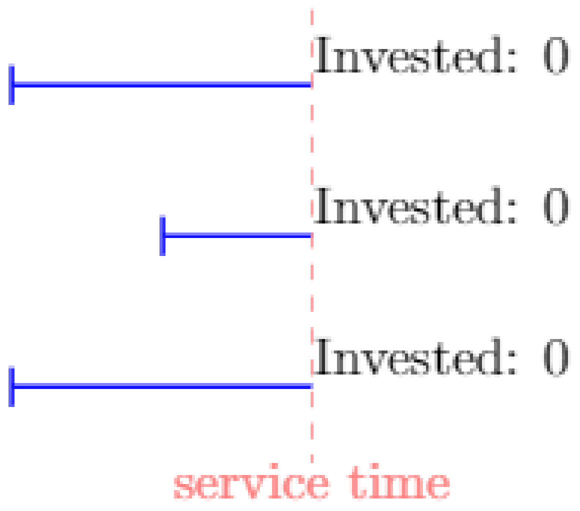
When do we serve requests?

- Services will spend at most $O(1) \times \sigma$
- **Idea:** Make a service when delay $\approx$ expected ordering cost
- Wait until requests accumulate total delay of σ



What do we serve?

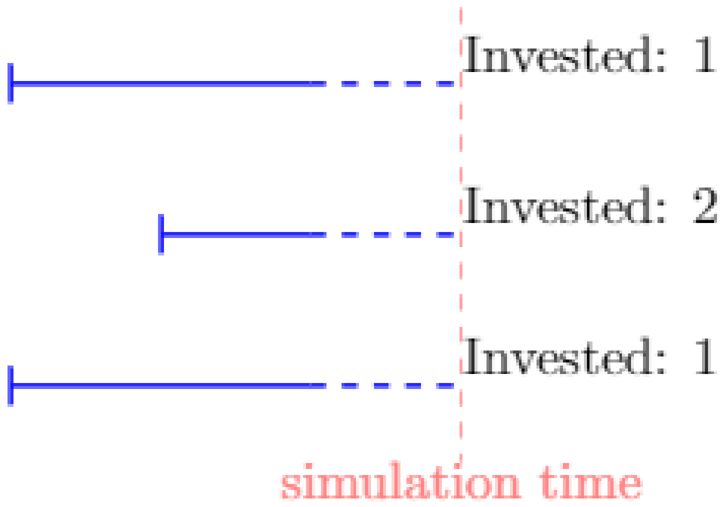
- Invest a total of σ in the future delay of requests
- If δ is invested in an item type, serve it



What do we serve?

– Invest a total of σ in the future delay of requests

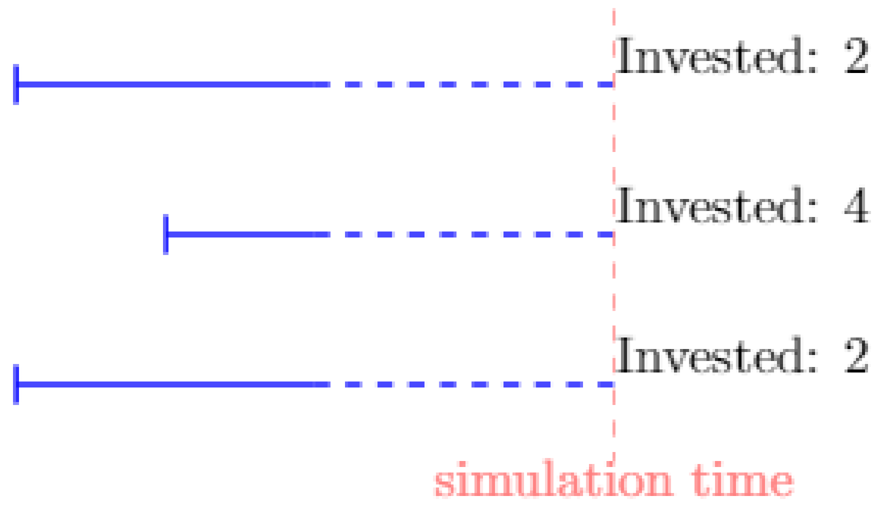
– If δ is invested in an item type, serve it



What do we serve?

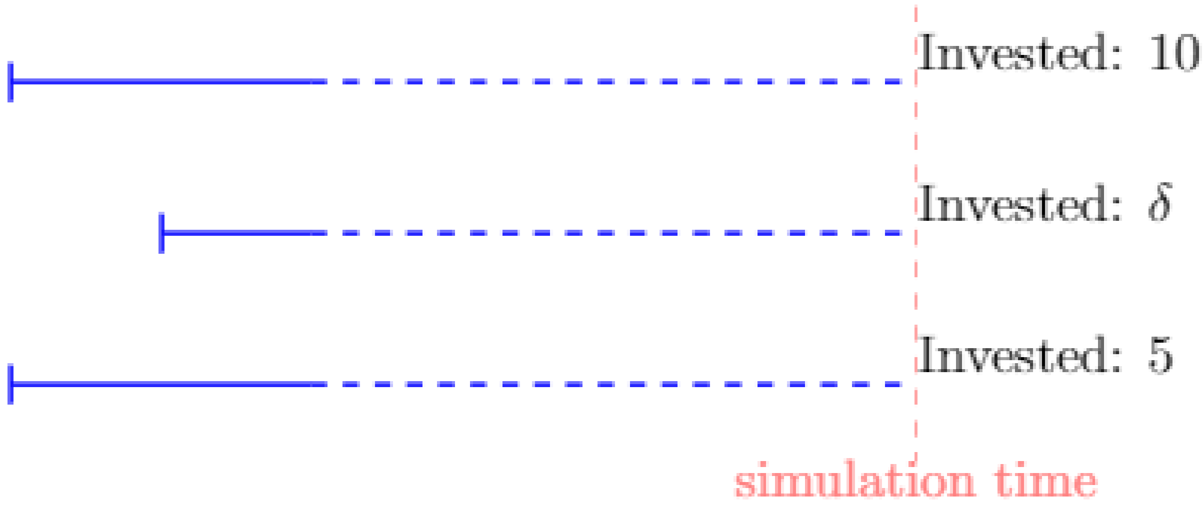
– Invest a total of σ in the future delay of requests

– If δ is invested in an item type, serve it



What do we serve?

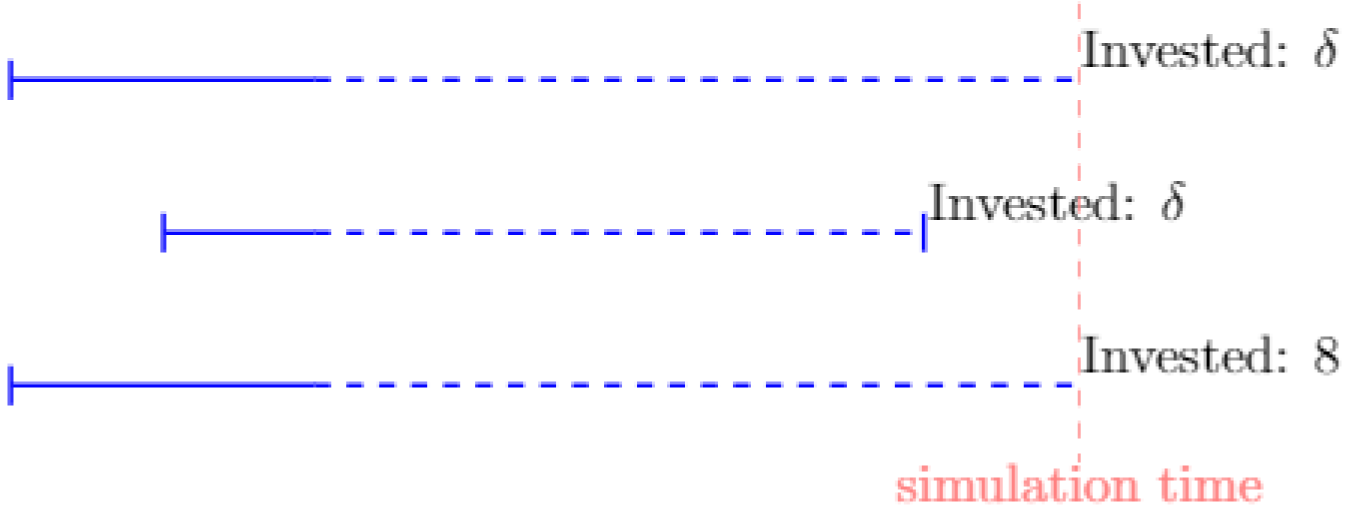
– Invest a total of σ in the future delay of requests
– If δ is invested in an item type, serve it



What do we serve?

– Invest a total of σ in the future delay of requests

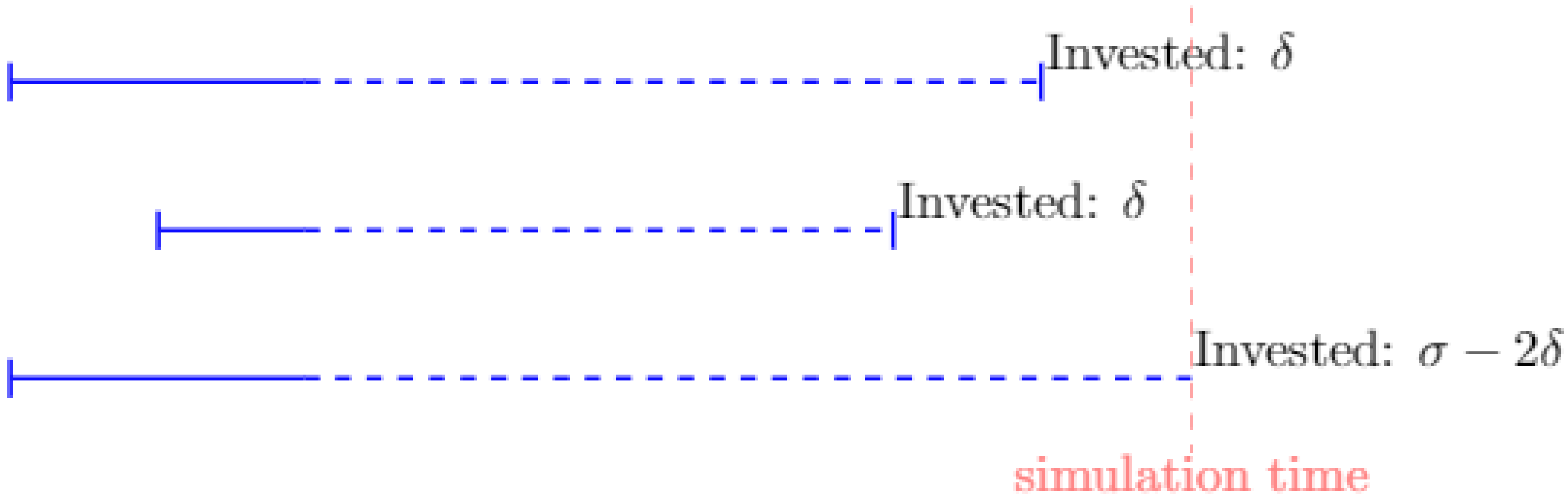
– If δ is invested in an item type, serve it



What do we serve?

– Invest a total of σ in the future delay of requests

– If δ is invested in an item type, serve it



– Balances need to serve as much as possible with leaving requests to wait for other requests on the same item type

Proof sketch

- **Disclaimer:** Very simplified proof sketch
- $ALG =$ Ordering costs + Investment costs
- Ordering costs $\leq$ Investment costs $\leq OPT$

  $\Rightarrow ALG \leq 2 \times OPT$

Ordering costs $\leq$ Investment costs

- Serving a request costs δ
- Serve requests when δ has been invested into it
- An ordering cost of δ is only incurred when an investment cost of δ has already been paid

Investment costs $\leq OPT$

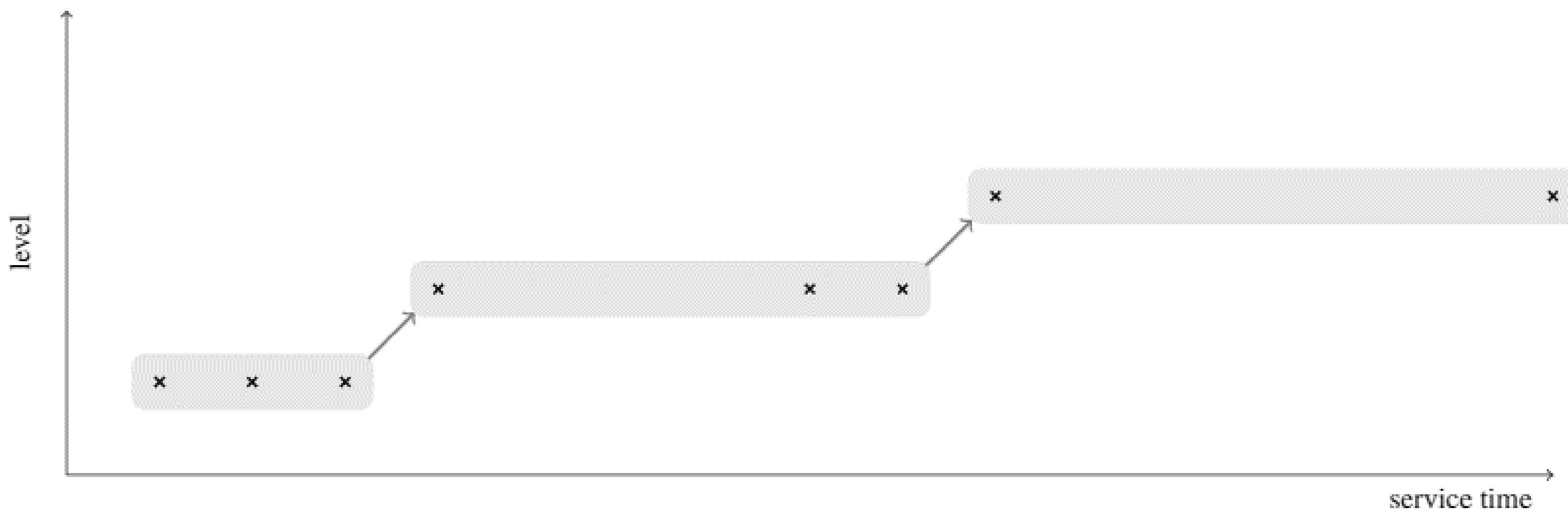
- OPT must serve all requests
- Consider any request q
- There can be at most δ invested in q
- OPT pays δ to serve q
- Investments in $q \leq OPT$'s ordering cost for q
- Apply this argument to all requests to get the desired result

Why is this analysis technique novel?

- Previously: Delay costs $\leq$ Ordering costs $\leq OPT$
- Now: Ordering costs $\leq$ Investment costs $\leq OPT$
- Delay costs = Investment costs
- My approach is the opposite of what is traditionally done

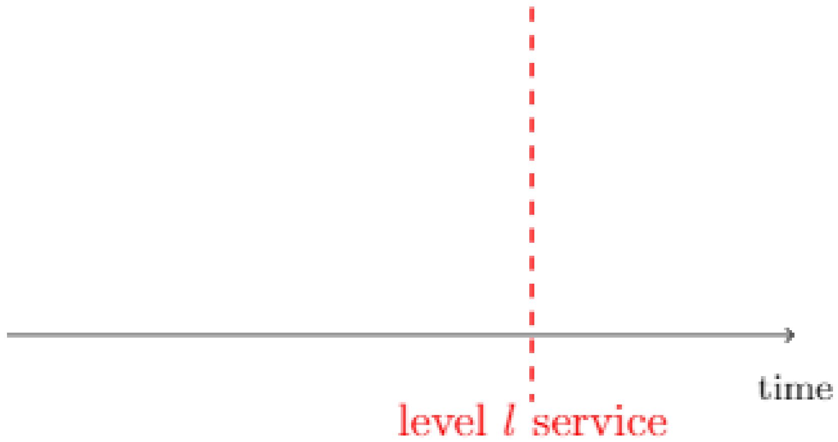
Generalising to arbitrary piecewise affine functions

- Use service levels as done by Khatkar (2020)
- A level l service will serve using the level l piece $\sigma_l + \delta_l x$
- Services will start at level 1 and upgrade over time



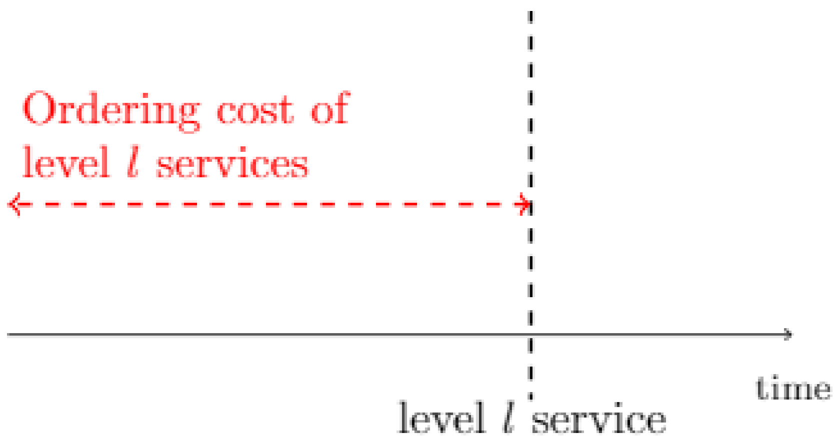
Key Contribution 2: Upgrading services

Previously (Khatkar 2020)



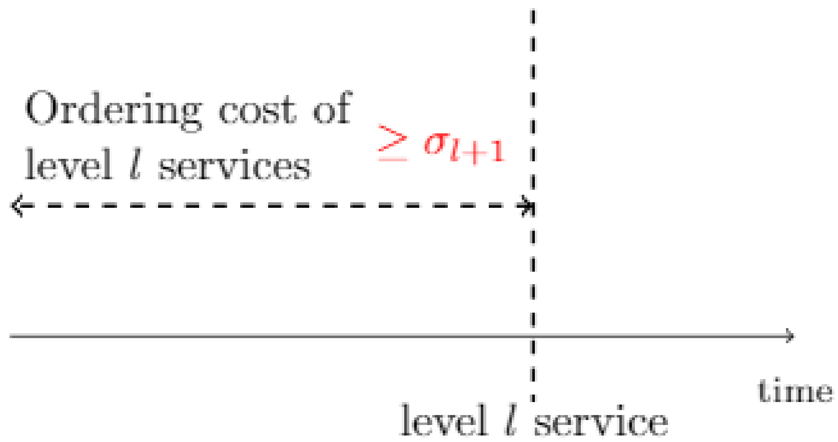
Key Contribution 2: Upgrading services

Previously (Khatkar 2020)



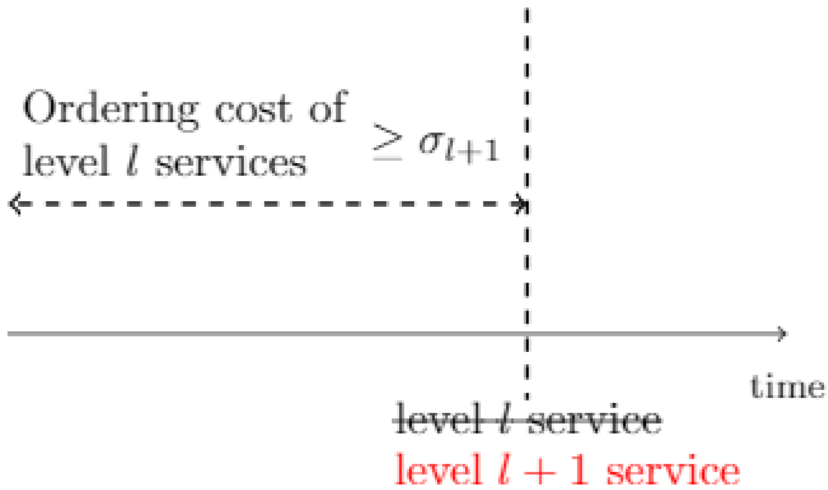
Key Contribution 2: Upgrading services

Previously (Khatkar 2020)



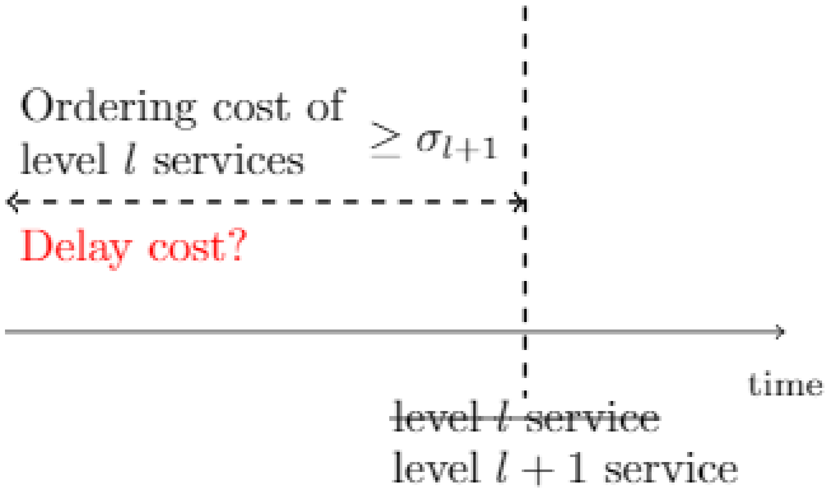
Key Contribution 2: Upgrading services

Previously (Khatkar 2020)



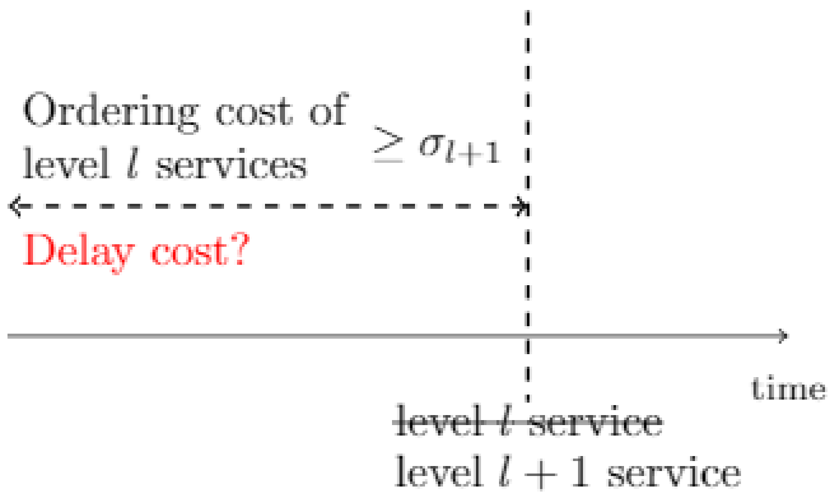
Key Contribution 2: Upgrading services

Previously (Khatkar 2020)

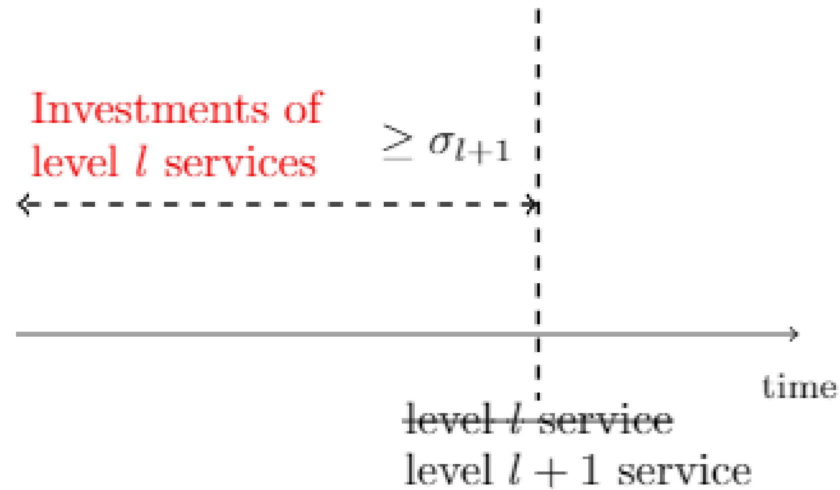


Key Contribution 2: Upgrading services

Previously (Khatkar 2020)



Now (My contribution)



- Ordering costs $\leq$ Investment costs, Investments $=$ Delays

  $\Rightarrow$ My contribution bounds both the ordering and delay costs

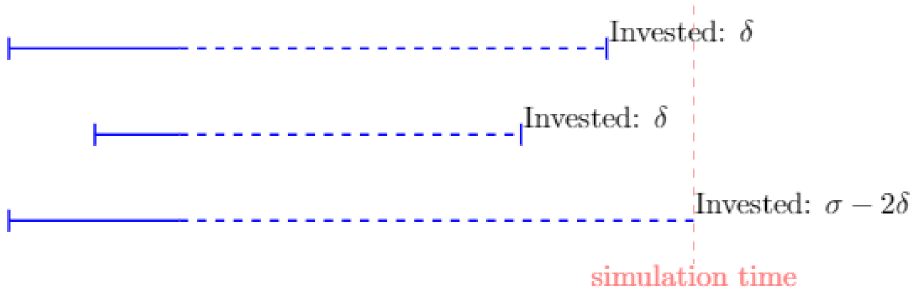
- Services made using algorithm from earlier

Summary: Algorithm

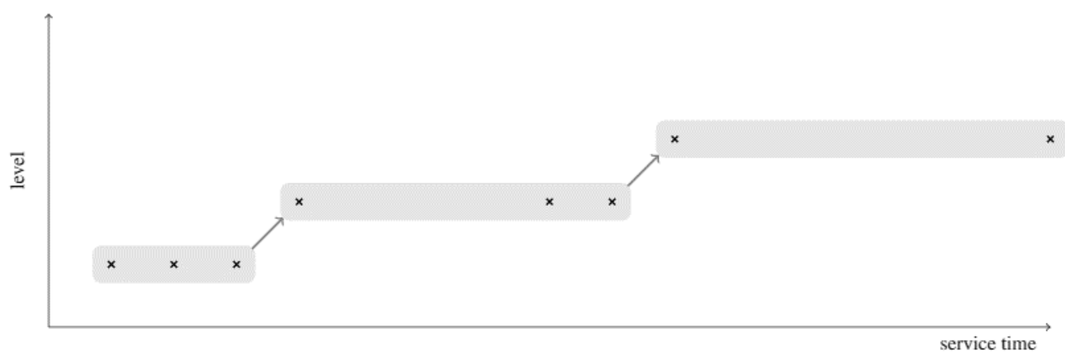
1) **When** do we serve?

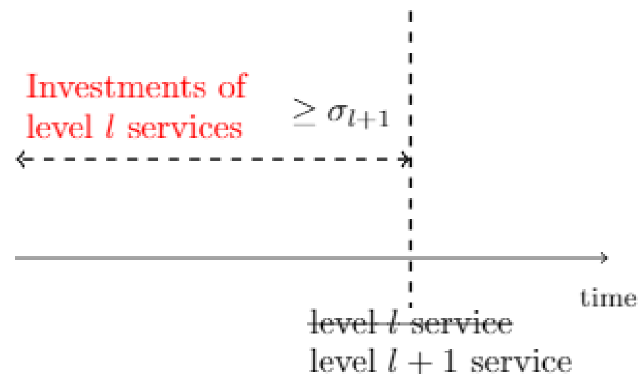
Delay ≈ Expected Ordering Cost

2) **What** do we serve?



3) **How** much to serve?





Summary: Analysis

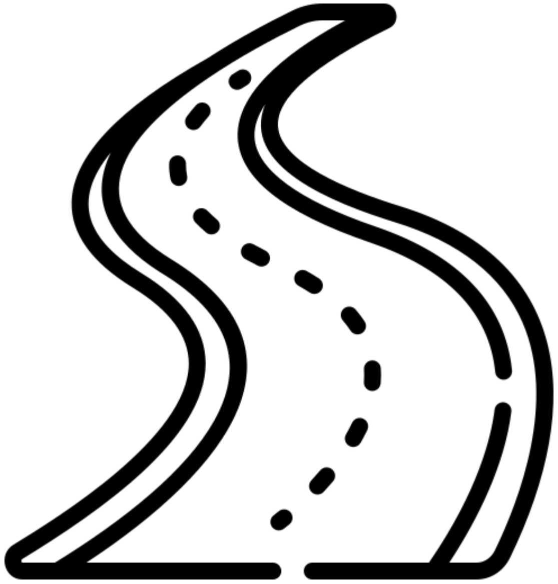
- $O(1)$-competitive algorithm for Cardinality JRP with Delay
- Ordering costs $\leq$ Investment costs $\leq OPT$
- $ALG =$ Ordering costs $+$ Investment costs

  $\Rightarrow ALG \leq 2 \times OPT$


SERVICE LEVELS
INVESTMENTS
PIECEWISE REDUCTION
SERVICE POINTERS
TRIGGERING REQUESTS
PRIMARY SERVICES
CHARGED COSTS
REQUEST LEVELS
SERVICE CHAINS
CHARGED INTERVALS

Future Work

- Create online algorithms for other JRP variants
- Applying the idea of extended investments to other problems with delay



Questions?

References

Becchetti, L., Marchetti-Spaccamela, A., Vitaletti, A., Korteweg, P., Skutella, M., Stougie, L.: Latency-constrained aggregation in sensor networks. ACM Transactions on Algorithms 6(1), 13:1–13:20 (2009)

Bienkowski M., Byrka J., Chrobak M., Dobbs N., Nowicki T., Sviridenko M., Swirszcz G., Young N.E. Approximation Algorithms for the Joint Replenishment Problem with Deadlines. In: Fomin F.V., Freivalds R., Kwiatkowska M., Peleg D. (eds) Automata, Languages, and Programming. ICALP 2013. Lecture Notes in Computer Science, vol 7965. Springer, Berlin, Heidelberg. (2013)

Cheung M., Elmachtoub A., Levi R., Shmoys D. The submodular joint replenishment problem. Mathematical Programming. 158. 1-27. (2015)

Buchbinder N., Kimbrel T., Levi R., Makarychev K., Sviridenko M. Online Make-to-Order Joint Replenishment Model: Primal-Dual Competitive Algorithms. Operations Research. 61. 952-961. 10.1145/1347082.1347186. (2008)

Khatkar J.., Online Joint Replenishment with Deadline on Subadditive Costs, Honours Thesis, The University of Sydney (2020)

Azar Y., Touitou N. Beyond Tree Embeddings - a Deterministic Framework for Network Design with Deadlines or Delay. 61st IEEE Annual Symposium on Foundations of Computer Science, FOCS, Durham, NC, USA (2020)

Guha S., Meyerson A., Munagala K. A constant factor approximation for the single sink edge installation problems. In Proceedings of the thirty-third annual ACM symposium on Theory of computing (STOC '01). Association for Computing Machinery, New York, NY, USA, 383–388. (2001)

Further Proof Details

- $ALG =$ Shared ordering costs σ + Individual ordering costs δx + Triggering delay costs σ + Investment costs
- 3 service types: primary, normal, tail
- Services cost $\leq 3\ \sigma$ (excluding individual costs)
- Tail service cost $\leq$ Primary service cost
- Primary services are disjoint so define intervals $[a, t]$ within which we are guaranteed OPT cost $\approx$ primary service cost
- Normal services have an investment cost of σ which we can charge all other costs to. Since Investment costs $\leq OPT$ then normal service costs $\leq OPT$