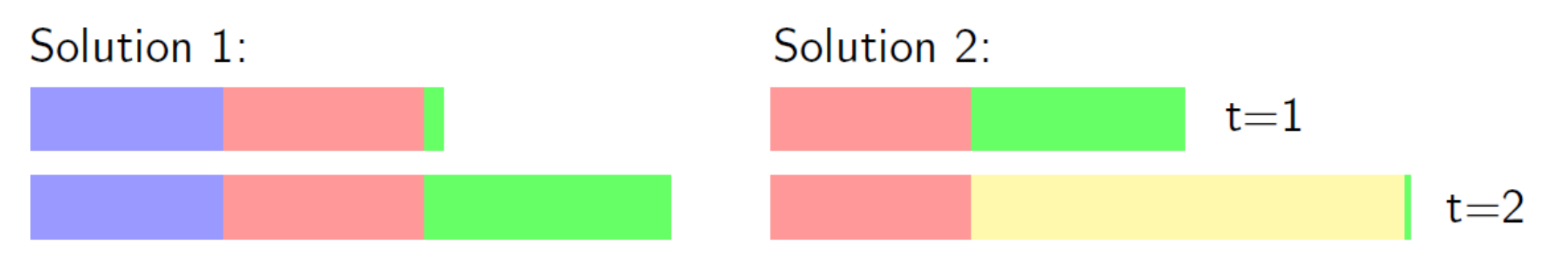
Approximation Algorithms for The Generalized Incremental Knapsack Problem

Yuri Faenza [1], Danny Segev [2], Lingyi Zhang [1]

[1] Columbia University, [2] Tel Aviv University

Generalized Incremental Knapsack

- Generalized Incremental Knapsack (GIK) is an extension of the classical knapsack problem to a multi-period, discrete setting:
  - In each period, the knapsack capacity is W_t;
  - W_t is a function non-decreasing in t;
  - Once an item has been packed in some period, it cannot be removed in later periods.
  - If item i is first inserted in the knapsack at time t, we earn profit $p_{i,t} \geq 0$.
  - The goal is to maximize the total profit gained over the horizon T.



IP Formulation

$$\begin{aligned}
\max \quad & \textstyle\sum_{t=1}^{T} \sum_{i=1}^{n} p_{i,t}(x_{i,t} - x_{i,t-1}) & \\
\text{s.t.} \quad & \textstyle\sum_{i=1}^{n} w_i x_{i,t} \leq W_t & \forall t \in [T], \\
& x_{i,t} \leq x_{i,t+1} & \forall t \in [T-1], i \in [n], \\
& x_{i,t} \in \{0,1\} & \forall t \in [T], i \in [n].
\end{aligned}$$

Motivation

- The problem naturally models scenarios where resources increase over time.
- The added time component of GIK provides an extra level of complexity and differs from other generalizations of the knapsack problem, such as the generalized assignment problem (GAP) or the multiple knapsack problem (MKP).

Known Results for Special Cases

- Restricted cases of GIK:
  - Time-invariant incremental knapsack (IIK): $p_{i,t} = p_i \cdot (T - t + 1)$ – that is, each item gains profit $p_i \geq 0$ for each time it is in the knapsack.
  - Incremental knapsack (IK): $p_{i,t} = \sum_{t' \geq t} \Delta_{t'} \cdot p_i$ – that is, if an item is in the knapsack at time t, it earns profit $p_i \cdot \Delta_t$.
- IIK is strongly NP-hard [2] and has a PTAS [3].
- IK has a PTAS [1].
- The above formulation (even for IIK) has an unbounded integrality gap [2].

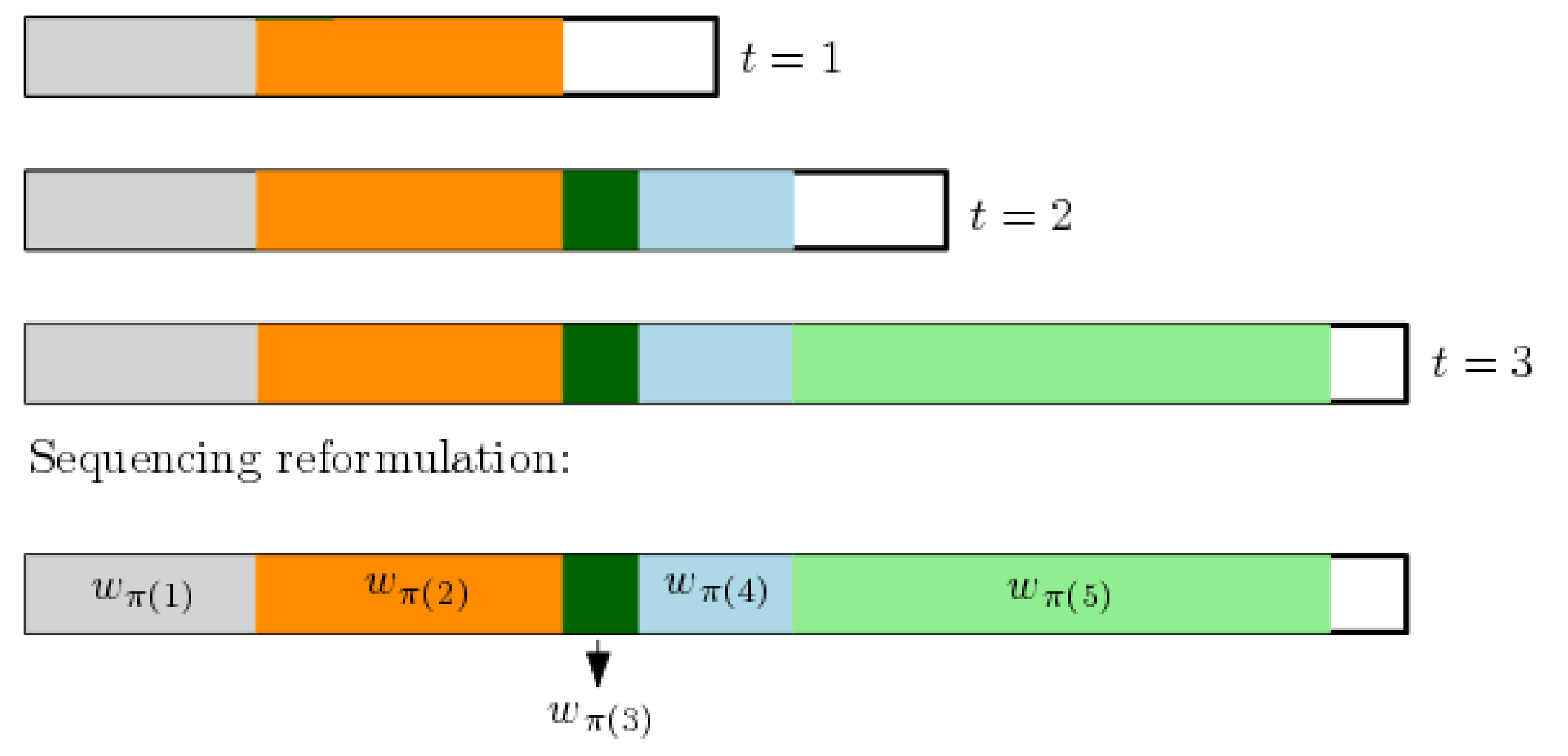
Our Contributions and Open Questions

- A polynomial-time $(\frac{1}{2} - \epsilon)$-approximation for GIK.
- A QPTAS for GIK.
- Can we improve upon the $(\frac{1}{2} - \epsilon)$-approximation in polynomial time?
- Does there exist a PTAS?

$(\frac{1}{2} - \epsilon)$-Approximation

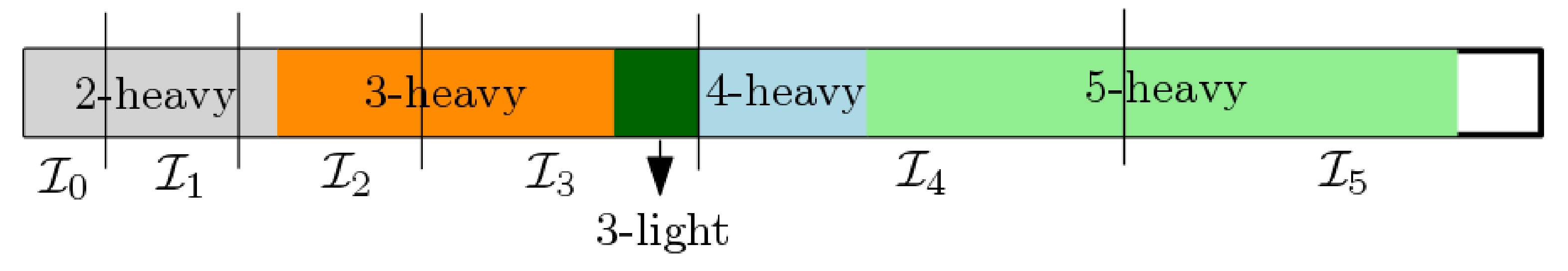
Sequencing Reformulation

We reformulate the GIK problem as a sequencing problem on a single machine.



- Completion time: $C_\pi(i) = \sum_{j \in [n]: \pi(j) \leq \pi(i)} w_j$.
- Profit: $\varphi_\pi(i) = \max\{p_{i,t} : t \in [T+1] \text{ and } W_t \geq C_\pi(i)\}$, with the convention that $W_{T+1} = \infty$ and $p_{i,T+1} = 0$.
- Equivalent GIK objective: find permutation $\pi : [n] \to [n]$ such that $\Psi(\pi) = \sum_{i \in [n]} \varphi_\pi(i)$ is maximized.

Bucketing



- We partition the interval $[0, \sum_{i \in [n]} w_i]$ into $K = \lceil \log_{1+\epsilon}(\sum_{i \in [n]} w_i) \rceil$ intervals $\mathcal{I}_0, \ldots, \mathcal{I}_K$.
- $\mathcal{I}_0 = [0,1]$; $\mathcal{I}_k = ((1+\epsilon)^{k-1}, (1+\epsilon)^k]$ for $k \in [K]$.
- An item is k-heavy if $w_i \geq \epsilon^2 \cdot (1+\epsilon)^k$, k-light otherwise.
- Profit decomposition:

$$\Psi(\pi) = \underbrace{\sum_{k \in [K]_0} \sum_{i \in H_k : C_\pi(i) \in \mathcal{I}_k} \varphi_\pi(i)}_{\Psi_{\text{heavy}}(\pi)} + \underbrace{\sum_{k \in [K]_0} \sum_{i \in L_k : C_\pi(i) \in \mathcal{I}_k} \varphi_\pi(i)}_{\Psi_{\text{light}}(\pi)} .$$

Heavy Items Profit Contribution

$(1-\epsilon)$-approximate dynamic programming with enumeration of internal permutation of k-heavy items whose completion time falls in each interval $\mathcal{I}_k$.

- Since by construction, the number of heavy items with completion time falling in each interval is bounded, the dynamic programming table can be computed in polynomial time.

Light Items Profit Contribution

- We can formulate an instance of the generalized assignment problem (GAP) over light items whose optimal solution is feasible and gives a $(1-\epsilon)$-optimal packing of light items in our original GIK instance.
- A "slightly infeasible", super-optimal solution to the GAP instance is obtained using the Shmoys-Tardos algorithm [4].
- Feasibility with negligible profit loss is restored via rounding.

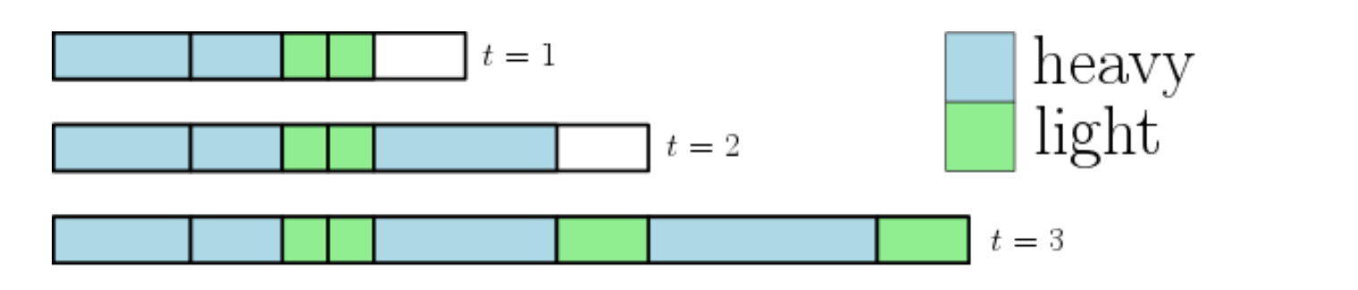
Theorem: $(\frac{1}{2} - \epsilon)$-Approximation

Taking the more profitable of the heavy solution and the light solution gives a polynomial-time $(\frac{1}{2} - \epsilon)$-approximated solution.

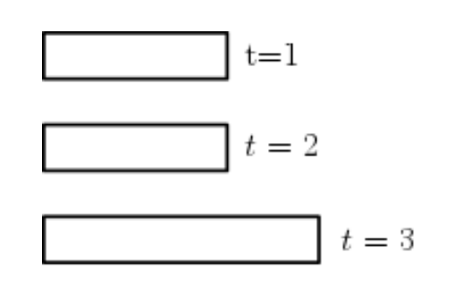
QPTAS

A self-improving algorithm:

(1) "Guess" the heavy items



(2) Create a residual GIK instance removing the weights of guessed heavy items and solve it to $(\frac{1}{2} - \epsilon)$-optimality.



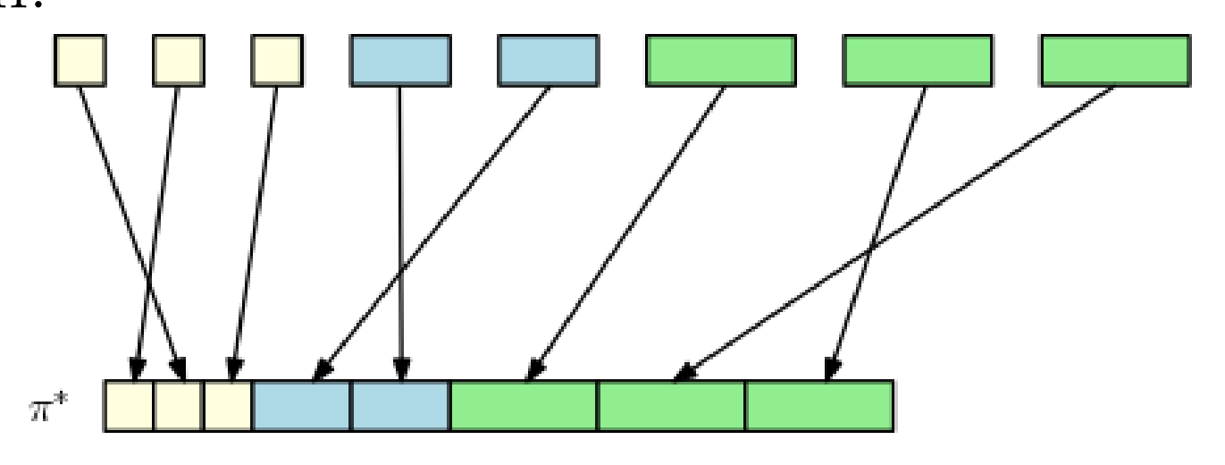
(3) Independently, find an $(1-\epsilon)$-approximated solution for light items in the original instance.

- Take the better of steps (1)+(2) and step (3) gives a $(\frac{2}{3} - O(\epsilon))$-approximation.
- Repeating steps (1)-(3), but now using the $(\frac{2}{3} - O(\epsilon))$-approximated algorithm in step (2) gives a $(\frac{3}{4} - O(\epsilon))$-approximated algorithm.
- Repeat for $\lceil \frac{2}{\delta} \rceil$ rounds gives a $(1-\delta)$-approximated solution.

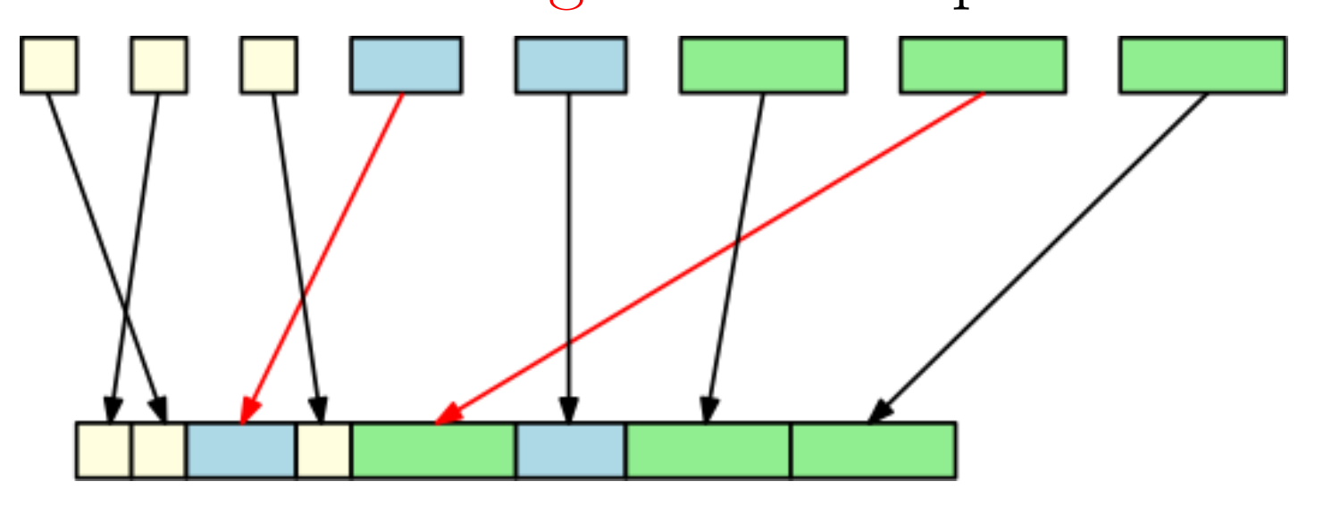
Running time: Guessing the heavy items takes time $O((nT)^{\frac{1}{\delta^5} \cdot \log(n \cdot \frac{w_{\max}}{w_{\min}})})$, thus the entire procedure takes time $O((nT)^{\frac{1}{\delta^5} \cdot \log(n \cdot \frac{w_{\max}}{w_{\min}})}) \cdot |\mathcal{I}|^{O(1)})$.

Removing the Dependence on $\frac{w_{\max}}{w_{\min}}$

Ideal situation:



Real situation – small crossing is almost optimal:



Lemma 1 *Suppose items are ordered in weight clusters $\mathcal{C}_1, \ldots \mathcal{C}_M$ such that for any $m_1 < m_2$, if $i \in \mathcal{C}_{m_1}$, $j \in \mathcal{C}_{m_2}$, $w_i < w_j$. There exists a permutation π with $\Psi(\pi) \geq (1-\epsilon)\Psi(\pi^*)$ such that for every $m \in [M]$, at most $O(\frac{\log M}{\epsilon})$ items from weight clusters $\mathcal{C}_{m+1}, \ldots, \mathcal{C}_M$ appear in π before items in $\mathcal{C}_m$.*

Idea for QPTAS:

- For every $m \in [M]$, we can use dynamic programming to "guess" these $O(\frac{\log M}{\epsilon})$ crossing items and their insertion times.
- For $m = 1, \ldots, M$, iteratively use the QPTAS for bounded $\frac{w_{\max}}{w_{\min}}$ on the residual instance to find the internal near-optimal permutation for items in $\mathcal{C}_m$ that are not crossing.

References

[1] Aouad, Ali, and Segev, Danny. "An approximate dynamic programming approach to the incremental knapsack problem." arXiv:2010.07633. (2020).

[2] Bienstock, Daniel, Sethuraman, Jay, and Ye, Chun. "Approximation Algorithms for the Incremental Knapsack Problem via Disjunctive Programming." arXiV:1311.4563. (2013).

[3] Faenza, Yuri, and Malinovic, Igor. "A PTAS for the Time-Invariant Incremental Knapsack problem." *Proceedings of ISCO* (2018).

[4] Shmoys, David and Tardos, Éva. "An approximation algorithm for the generalized assignment problem." *Mathematical Programming* (1993).