Priority Algorithms for the Subset-Sum Problem

Yuli Ye and Allan Borodin

Department of Computer Science
University of Toronto
Toronto, ON, Canada M5S 3G4
{y3ye,bor}@cs.toronto.edu

Abstract. Greedy algorithms are simple, but their relative power is not well understood. The priority framework [5] captures a key notion of "greediness" in the sense that it processes (in some locally optimal manner) one data item at a time, depending on and only on the current knowledge of the input. This algorithmic model provides a tool to assess the computational power and limitations of greedy algorithms, especially in terms of their approximability. In this paper, we study priority algorithm approximation ratios for the Subset-Sum Problem, focusing on the power of revocable decisions. We first provide a tight bound of $\alpha \approx 0.657$ for irrevocable priority algorithms. We then show that the approximation ratio of fixed order revocable priority algorithms is between $\beta \approx 0.780$ and $\gamma \approx 0.852$, and the ratio of adaptive order revocable priority algorithms is between 0.8 and $\delta \approx 0.893$.

1 Introduction

Greedy algorithms are of great interest because of their simplicity and efficiency. In many cases they produce reasonable (and sometimes optimal) solutions. Surprisingly, it is not obvious how to formalize the concept of a greedy algorithm and given such a formalism how to determine its power and limitations with regard to natural combinatorial optimization problems. Borodin, Nielson and Rackoff [5] suggested the priority model which provides a rigorous framework to analyze greedy-like algorithms. In this framework, they define fixed order and adaptive (order) priority algorithms, both of which capture a key notion of greedy algorithms in the sense that they process one data item at a time. For fixed order priority, the ordering function used to evaluate the priority of a data item is fixed before execution of the algorithm, while for adaptive priority, the ordering function can change during every iteration of the algorithm. By restricting algorithms to this framework, approximability results and limitations [1] for many problems have been obtained; for example, scheduling problems [5, 18], facility

[1] We note that similar to the study of online competitive analysis, negative priority results are in some sense incomparable with hardness of approximation results as there are no explicit complexity considerations as to how a priority algorithm can choose its next item and how it decides what to do with that item. Negative results are derived from the structure of the algorithm.

location and set cover [1], job interval selection (JISP and WJISP) [12], and various graph problems [6, 9]. The original priority framework specified that decisions (being made for the current input item) are irrevocable. Even within this restrictive framework, the gap between the best known algorithm and provable negative remains significant for most problems. Following [10, 4], Horn [12] extended the priority framework to allow revocable acceptances when considering packing problems. That is, input items could be accepted and then later rejected, the only restriction being that a feasible solution is maintained at the end of each iteration. The revocable (decision) priority model is intuitively more powerful and almost as conceptually simple as the irrevocable model and it is perhaps surprising that it is not a more commonly used type of algorithm. Erlebach and Spieksma [10] and independently Bar-Noy et al. [4] provide a simple revocable priority approximation algorithm for the WJISP problem, and Horn [12] formalizes this model and provides an approximation upper bound [2] of $\approx 1/(1.17)$ for the special case of the weighted interval scheduling problem. Moore's [17] optimal "greedy algorithm" for the unweighted throughput maximization problem without release times (i.e. $1||\sum_j \bar{U}_j$ in Graham's scheduling notation) can be implemented as a fixed order revocable priority algorithm. It is not difficult to show that this problem cannot be solved optimally by an irrevocable priority algorithm.

The *Subset-Sum Problem* (SSP) is one the most fundamental NP-complete problems [11], and perhaps the simplest of its kind. Approximation algorithms for SSP have been studied extensively in the literature. The first FPTAS (for the more general knapsack problem) is due to Ibarra and Kim [13], and the best current approximation algorithm is due to Kellerer et al. [15], having time and space complexity $O(min\{\frac{n}{\varepsilon}, n+\frac{1}{\varepsilon^2}\log\frac{1}{\varepsilon}\})$ and $O(n+\frac{1}{\varepsilon})$ respectively. Greedy-like approximation algorithms have also been studied for SSP; an algorithm called greedy but using multiple passes, has approximation ratio 0.75, see [16]. In this paper, we study priority algorithms for SSP. Although in some sense one may consider SSP to be a "solved problem", the problem still presents an interesting challenge for the study of greedy algorithms. We believe the ideas employed for SSP will be applicable to the study of simple algorithms for other (say scheduling) problems which are not well understood, such as the throughput maximization problem (with release times) and some of its more tractable subcases. In particular, can we derive priority approximation algorithms for throughput maximization when all jobs have a fixed processing time (i.e. $1|r_j, p_j = p|\sum_j w_j\bar{U}_j$)? (We note that Horn's [12] $1/(1.17)$ bound also applies to this problem.) Baptiste [3] optimally solves this special case of throughput maximization using a dynamic programming algorithm with time complexity $O(n^7)$. (See also Chuzhoy et al. [8] and Chrobak et al. [7] for additional throughput maximization results.)

In spite of the conceptual simplicity of the SSP problem and the priority framework, there is still a great deal of flexibility in how one can design algorithms, both in terms of the ordering and in terms of which items to accept and

[2] As we are considering maximization problems in this paper, all approximation ratios will be ≤ 1 so that negative results become upper bounds on the ratio.

(for the revocable model) which items to discard in order to fit in a new item. We give a tight bound of $\alpha \approx 0.657$ for irrevocable priority algorithms showing that in this case adaptive ordering does not help. For fixed order revocable algorithms, we can show that the best approximation ratio is between $\beta \approx 0.780$ and $\gamma \approx 0.852$; for adaptive revocable priority algorithms, the best approximation ratio is between 0.8 and $\delta \approx 0.893$.

2 Definitions and Notation

We use **bold** font letters to denote sets of data items. For a given set $\mathbf{R}$ of data items, we use $|\mathbf{R}|$ to denote its cardinality and $\|\mathbf{R}\|$, its total weight. For a data item u, we use u to represent the singleton set $\{u\}$ and $2u$, the multi-set $\{u, u\}$; we also use u to represent the weight of u since it is the only attribute. The term u here is an overloaded term, but the meaning will become clear in the actual context. For set operations, we use $\oplus$ to denote set addition, and use $\ominus$ to denote set subtraction.

2.1 The Subset-Sum Problem

Given a set of n data items with positive weights and a capacity c, the *maximization version* of SSP is to find a subset such that the corresponding total weight is maximized without exceeding the capacity c. Without loss of generality, we make two assumptions. First of all, the weights are all scaled to their relative values to the capacity; hence we can use 1 instead of c for the capacity. Secondly, we assume each data item has weight $\in (0, 1]$. An *instance* σ of SSP is a set $\mathbf{I} = \{u_1, u_2, \ldots, u_n\}$ of n data items, where the set $\mathbf{I}$ is the *input set*, and $u_1, u_2, \ldots, u_n$ are the *data items*. A *feasible* solution of σ is a subset $\mathbf{B}$ of $\mathbf{I}$ such that $\|\mathbf{B}\| \le 1$. An *optimal* solution of σ is a feasible solution with maximum weight. We can formulate SSP as a solution of the following integer programming:

$$\text{maximize} \sum_{i=1}^{n} u_i x_i \tag{1}$$

$$\text{subject to} \sum_{i=1}^{n} u_i x_i \le 1, \tag{2}$$

where $x_i \in \{0, 1\}$ and $i \in [1, n]$. Let $\mathcal{A}$ be an algorithm for SSP, for a given instance σ, we denote $\mathcal{A}(\sigma)$ the solution achieved by $\mathcal{A}$ and $\textbf{OPT}(\sigma)$, the optimal solution, then the *approximation ratio* of $\mathcal{A}$ on σ is denoted by

$$\rho(\mathcal{A}, \sigma) = \frac{\|\mathcal{A}(\sigma)\|}{\|\textbf{OPT}(\sigma)\|}.$$

Let Σ be the set of all instances of SSP, then the *approximation ratio* of $\mathcal{A}$ over Σ is denoted by

$$\rho(\mathcal{A}) = \inf_{\sigma \in \Sigma} \rho(\mathcal{A}, \sigma).$$

In a particular analysis, the algorithm and the problem instance are usually fixed. For convenience, we often use **ALG**, **OPT** and ρ to denote the algorithm's solution, the optimal solution and the approximation ratio respectively.

2.2 Priority Model

We base our terminology and model on that of [5], and start with the class of fixed order irrevocable priority algorithms for SSP. For a given instance, a *fixed order irrevocable* priority algorithm maintains a feasible solution **B** throughout the algorithm. The structure of the algorithm [3] is as follows:

Fixed Order Irrevocable Priority
Ordering: Determine a total ordering of all possible data items
while **I** is not empty
  next := index of the data item in **I** that comes first in the ordering
  Decision: Decide whether or not to add u_{next} to **B**, and then remove u_{next} from **I**
end while

An *adaptive irrevocable* priority algorithm is similar to a fixed order one, but instead of looking at a data item according to some pre-determined ordering, the algorithm is allowed to reorder the remaining data items in **I** at each iteration. This gives the algorithm an advantage since now it can take into account the information that has been revealed so far to determine which is the best data item to consider next. The structure of an adaptive irrevocable priority algorithm is described as follows:

Adaptive Irrevocable Priority
while **I** is not empty
  Ordering: Determine a total ordering of all possible (remaining) data items
  next := index of the data item in **I** that comes first in the ordering
  Decision: Decide whether or not to add u_{next} to **B**, and then remove u_{next} from **I**
end while

The above defined priority algorithms are "irrevocable" in the sense that once a data item is admitted to the solution it cannot be removed. We can extend our

[3] We formalize the allowable (fixed) orderings by saying that the algorithm specifies a total ordering on all possible input items. The items that constitute the actual input set **I** will then inherit this ordering. That is, the priority model insists that the ordering satisfies Arrow's Independence of Irrelevant Attributes IIA Axion [2]. For adaptive orderings the algorithm can construct a new IIA ordering based on all the items that it has already seen as well as those items it can deduce are not in the input set.

notion of "fixed order" and "adaptive" to the class of revocable priority algorithms, where revocable decisions on accepted data items are allowed. Accordingly, those algorithms are called *fixed order revocable* and *adaptive revocable* priority algorithms respectively. The extension[4] to revocable acceptances provides additional power; for example, as shown in [14], *online irrevocable* algorithms for SSP cannot achieve any constant bound approximation ratio, while *online revocable* algorithms can achieve a tight approximation ratio of $\frac{\sqrt{5}-1}{2} \approx 0.618$.

2.3 Adversarial Strategy

We utilize an adversary in proving approximation bounds. For a given priority algorithm, we run the adversary against the algorithm in the following scheme. At the beginning of the algorithm, the adversary first presents a set of data items to the algorithm, possibly with some data items having the same weight. Furthermore, our adversary promises that the actual input is contained in this set[5]. Since weight is the only input parameter, the algorithm give the same priority to all items having the same weight [6]. At each step, the adversary asks the algorithm to select one data item in the remaining set and make a decision on that data item. Once the algorithm makes a decision on the selected item, the adversary then has the power to remove any number of data items in the remaining set; this repeats until the remaining set is empty, which then terminates the algorithm.

For convenience, we often use a diagram to illustrate an adversarial strategy. A diagram of an adversarial strategy is an acyclic directed graph, where each node represents a possible state of the strategy, and each arc indicates a possible transition. Each state of the strategy contains two boxes. The first box indicates the current solution maintained by the algorithm, the second box indicates the remaining set of data items maintained by the adversary. A state can be either terminal or non-terminal. A state is *terminal* if and only if it is a sink, in the sense that the adversary no longer need perform any additional action; we indicate a terminal state using **bold** boxes. Each transition also contains two lines of actions. The first line indicates the action taken by the algorithm and the second line indicates the action taken by the adversary. Sometimes the algorithm may need to reject certain data items in order to accept a new one, so an action may contain multiple operations which occur at the same time; we use $\oslash$ to indicate no action. Note that to calculate a bound for the approximation ratio of an algorithm, it is sufficient to consider the approximation ratios achieved in all terminal states. We will see such diagrams in Sect. 4.

[4] This extension applies to priority algorithms for packing problems.

[5] This assumption is optional. The approximation bounds clearly hold for a stronger adversary.

[6] Technically we can use an item number identifier to further distinguish items, but by providing sufficiently many copies of an item the adversary can effectively achieve what the statement claims.

3 General Simplifications

We first provide two simplifications for general priority algorithms, both of which are based on the approximation ratio, say θ, we want to achieve.

3.1 Implicit Terminal Conditions

Since we are interested in approximation algorithms, we can terminate an algorithm at any time if the approximation ratio of θ is achieved. This condition is called a *terminal condition*.

- For a fixed irrevocable priority algorithm, the terminal condition is satisfied, if at the beginning of some step of the algorithm, u is the next data item to be examined, $\mathbf{B}' = (\mathbf{B} \oplus u)$ and $\theta \le \|\mathbf{B}'\| \le 1$.
- For an adaptive irrevocable priority algorithm, the terminal condition is satisfied, if at the beginning of some step of the algorithm, there exists some $u \in \mathbf{I}$ and $\mathbf{B}' = (\mathbf{B} \oplus u)$ such that $\theta \le \|\mathbf{B}'\| \le 1$. In this case, u is the next data item, and the approximation ratio can be achieved.
- For a fixed revocable priority algorithm, the terminal condition is satisfied, if at the beginning of some step of the algorithm, u is the next data item to be examined, and there exists $\mathbf{B}' \subseteq (\mathbf{B} \oplus u)$ such that $\theta \le \|\mathbf{B}'\| \le 1$.
- For an adaptive revocable priority algorithm, the terminal condition is satisfied, if at the beginning of some step of the algorithm, there exists some $u \in \mathbf{I}$ and $\mathbf{B}' \subseteq (\mathbf{B} \oplus u)$ such that $\theta \le \|\mathbf{B}'\| \le 1$. In this case, u is the next data item, and the approximation ratio can be achieved.

It is clear that in all four cases, the algorithm can take $\mathbf{B}'$ as the final solution and immediately terminate. For any algorithm given in this paper, we will not explicitly state the check of the terminal condition; we assume that the algorithm tests the condition whenever it considers the next data item, and will terminate if the condition is satisfied. Note that here we do not impose any time bound for checking the terminal condition in the general model. But for all the algorithms studied in this paper, the extra check for the terminal condition does not increase much for the time complexity as the input against such test is highly restricted, and the running time is bounded by a constant.

3.2 Exclusion of Small and Extra Large Data Items

For the approximation ratio θ, a data item u is said to be in class $\mathbf{S}$ and $\mathbf{X}$ if $0 < u \le 1 - \theta$ and $\theta \le u \le 1$ respectively. A data item is *small* if it is in $\mathbf{S}$, and *extra large* if it is in $\mathbf{X}$; a data item is *relevant* if it is neither small nor extra large. Let Σ' be the set of instances of SSP whose input contains only relevant data items, and let $\mathcal{A}'$ be a priority algorithm over Σ'; we call $\mathcal{A}'$ a *restricted* algorithm. For a given instance $\sigma \in \Sigma$ with input $\mathbf{I}$, we let $\sigma' \in \Sigma'$ be the instance with input $\mathbf{I} \ominus \mathbf{S} \ominus \mathbf{X}$. An algorithm $\mathcal{A}$ over Σ is a *completion*[7] of

[7] This is only defined on valid algorithms, i.e., both solutions of $\mathcal{A}$ and $\mathcal{A}'$ have to be feasible.

$\mathcal{A}'$ if for any instance σ, either $\rho(\mathcal{A},\sigma) \geq \theta$ or $\mathbf{X} \cap \mathbf{I} = \emptyset$, $\mathcal{A}(\sigma) \ominus \mathbf{S} = \mathcal{A}'(\sigma')$ and $\mathcal{A}(\sigma) \cap \mathbf{S} = \mathbf{S} \cap \mathbf{I}$. This definition implies that if $\mathbf{X} \cap \mathbf{I} \neq \emptyset$, then we have $\rho(\mathcal{A},\sigma) \geq \theta$.

Proposition 1. *Let $\mathcal{A}$ be a completion of $\mathcal{A}'$. If $\mathcal{A}'$ achieves approximation ratio θ over Σ', then $\mathcal{A}$ achieves approximation ratio θ over Σ.*

Proof. Suppose that $\mathcal{A}'$ achieves approximation ratio θ over Σ', but $\mathcal{A}$ achieves approximation ratio less than θ over Σ, then $\mathbf{X} \cap \mathbf{I} = \emptyset$ and for a given instance σ, $\mathcal{A}(\sigma) \ominus \mathbf{S} = \mathcal{A}'(\sigma')$ and $\mathcal{A}(\sigma) \cap \mathbf{S} = \mathbf{S} \cap \mathbf{I}$. Hence, we have

$$\|\textbf{OPT}(\sigma) \cap \mathbf{S}\| \leq \|\mathbf{S} \cap \mathbf{I}\| = \|\mathcal{A}(\sigma) \cap \mathbf{S}\|.$$

Therefore, approximation ratio of $\mathcal{A}$ on σ is given by

$$\rho(\mathcal{A},\sigma) = \frac{\|\mathcal{A}(\sigma)\|}{\|\textbf{OPT}(\sigma)\|} = \frac{\|\mathcal{A}(\sigma) \cap \mathbf{S}\| + \|\mathcal{A}(\sigma) \ominus \mathbf{S}\|}{\|\textbf{OPT}(\sigma) \cap \mathbf{S}\| + \|\textbf{OPT}(\sigma) \ominus \mathbf{S}\|} \geq \frac{\|\mathcal{A}(\sigma) \ominus \mathbf{S}\|}{\|\textbf{OPT}(\sigma) \ominus \mathbf{S}\|}.$$

On the other hand, since the algorithm $\mathcal{A}'$ achieves approximation ratio of θ on σ', we have

$$\rho(\mathcal{A}',\sigma') = \frac{\|\mathcal{A}'(\sigma')\|}{\|\textbf{OPT}(\sigma')\|} \geq \theta.$$

Since $\mathcal{A}'(\sigma') = \mathcal{A}(\sigma) \ominus \mathbf{S}$, and $\|\textbf{OPT}(\sigma')\| \geq \|\textbf{OPT}(\sigma) \ominus \mathbf{S}\|$. Therefore,

$$\rho(\mathcal{A},\sigma) \geq \frac{\|\mathcal{A}(\sigma) \ominus \mathbf{S}\|}{\|\textbf{OPT}(\sigma) \ominus \mathbf{S}\|} \geq \frac{\|\mathcal{A}'(\sigma')\|}{\|\textbf{OPT}(\sigma')\|} \geq \theta.$$

Therefore, the algorithm $\mathcal{A}$ achieves approximation ratio of θ, which is a contradiction. □

Proposition 1 shows that if we have a restricted algorithm which achieves approximation ratio θ over Σ' and it has a completion, then the completion achieves approximation ratio θ over Σ. For a given class $\mathbb{C}$ of priority algorithms, a completion is $\mathbb{C}$-*preserving* if both algorithms are in $\mathbb{C}$.

Proposition 2. *Let $\mathbb{C}$ be the class of* [fixed | adaptive] [irrevocable | revocable] *priority algorithms, then there exists a $\mathbb{C}$-preserving completion from a restricted algorithm in $\mathbb{C}$.*

Proof. Given a restricted algorithm $\mathcal{A}'$ in $\mathbb{C}$, the algorithm $\mathcal{A}$ is constructed from $\mathcal{A}'$ with three phases:

- **Phase 1:** if there is an extra large data item then take that data item as the final solution, and terminate the algorithm; otherwise go to **Phase 2**.
- **Phase 2:** run the algorithm $\mathcal{A}'$ on σ', and then go to **Phase 3**.
- **Phase 3:** greedily fill the solution with small data items until either an overflow or the exhaustion of small data items; terminate the algorithm.

It is clear that the above construction preserves the class of fixed irrevocable priority algorithms, which is the most restricted class among the four combinations. Therefore, such construction is $\mathbb{C}$-preserving. There are three possible ways for algorithm $\mathcal{A}$ to terminate. If the algorithm $\mathcal{A}$ terminates in **Phase 1**, then it clearly achieves approximation ratio of θ; otherwise, it terminates in **Phase 3**. There are two cases. If the termination is caused by an overflow of a small data item, then the total weight of the solution is $\geq \theta$, hence the target approximation ratio is also achieved. If the termination is caused by the exhaustion of small data items, then there exists no extra large data item and the solution admits all the small data items. Furthermore, the relevant data items kept in the solution by $\mathcal{A}$ on σ are the same as those in the solution by $\mathcal{A}'$ on σ'. In all three cases, the definition of completion is satisfied. Therefore the algorithm $\mathcal{A}$ is a $\mathbb{C}$-preserving completion of $\mathcal{A}'$. □

Corollary 1. *Let* $\mathbb{C}$ *be the class of* [fixed | adaptive] [irrevocable | revocable] *priority algorithms, then there exists an* θ*-approximation algorithm in* $\mathbb{C}$ *if and only if there exists an* θ*-approximation restricted algorithm in* $\mathbb{C}$.

Proof. This follows immediately by Proposition 1 and 2. □

Proposition 3. *Let* $\mathbb{C}$ *be the class of* [non-increasing | non-decreasing] *order revocable priority algorithms, then there exists a* $\mathbb{C}$*-preserving completion from a restricted algorithm in* $\mathbb{C}$.

Proof. Given a restricted algorithm $\mathcal{A}'$ in $\mathbb{C}$, the algorithm $\mathcal{A}$ is constructed from $\mathcal{A}'$ as follows:

- If encounters an extra large item, then the terminal condition is satisfied.
- If encounters a relevant data item, then it acts the same as $\mathcal{A}'$ unless the terminal condition is satisfied.
- If encounters a small data item, then that data item always stays in the solution until the end of the algorithm or the terminal condition is satisfied.

It is clear that the above construction preserves the class of [non-increasing | non-decreasing] order revocable priority algorithms. Therefore, it is $\mathbb{C}$-preserving. There are two possible ways for algorithm $\mathcal{A}$ to terminate. If the termination of the algorithm is caused by a satisfaction of the terminal condition, then it clearly achieves approximation ratio of θ; otherwise, there exists no extra large data item and the solution admits all the small data items. Furthermore, the relevant data items kept in the solution by $\mathcal{A}$ on σ are the same as those in the solution by $\mathcal{A}'$ on σ'. In both cases, the definition of completion is satisfied. Therefore the algorithm $\mathcal{A}$ is a $\mathbb{C}$-preserving completion of $\mathcal{A}'$. □

Corollary 2. *Let* $\mathbb{C}$ *be the class of* [non-increasing | non-decreasing] *order revocable priority algorithms, then there exists an* θ*-approximation algorithm in* $\mathbb{C}$ *if and only if there exists an* θ*-approximation restricted algorithm in* $\mathbb{C}$.

Proof. This follows immediately by Proposition 1 and 3. □

Note that all the algorithms studied in this paper are covered by either Corollary 1 or 2, we can now safely assume the original input contains only relevant data items.

4 Priority Algorithms and Approximation Bounds

We first define four constants that will be used for our results. Let α, β, γ and δ be the real roots (respectively) of the equations $2x^3+x^2-1=0$, $2x^2+x-2=0$, $10x^2-5x-3=0$ and $6x^2-2x-3=0$ between 0 and 1. The corresponding values are shown in Table 1.

Table 1. Corresponding values.

name	α	β	γ	δ
value	≈ 0.657	≈ 0.780	≈ 0.852	≈ 0.893

We now give a tight bound for irrevocable priority algorithms. It is interesting that there is no approximability difference between fixed order and adaptive irrevocable priority algorithms.

Theorem 1. *There is a fixed order irrevocable priority algorithm for SSP with approximation ratio α, and every irrevocable priority algorithm for SSP has approximation ratio at most α.*

The case for revocable priority algorithms is more interesting. The ability to make revocable acceptances gives the algorithm a certain flexibility to "regret". The data items admitted into the solution are never "safe" until the termination of the algorithm. Therefore, if there is enough "room", it never hurts to accept a data item no matter how "bad" it is, as we can always reject it later at any time and with no cost. For the rest of the paper, we assume our algorithms will take advantage of this property. Besides this, the algorithm also assumes that the current set of accepted items $\mathbf{B}$ operates in one of the following four modes:

1. `Queue Mode:` In this mode, accepted items are discarded in the FIFO order to accommodate the new data item u.
2. `Queue_1 Mode:` In this mode, the first accepted item is never discarded, the rest data items are discarded in the FIFO order to accommodate the new data item u.
3. `Stack Mode:` In this mode, accepted items are discarded in the FILO order to accommodate the new data item u.
4. `Optimum Mode:` In this mode, accepted items are discarded to maximize $\|\mathbf{B}\|$; the new data item u may also be discarded for this purpose.

We use $\mathbf{B}_{\text{mode}}$ to represent the operational mode of $\mathbf{B}$. The algorithm can switch among these four modes during the processing of data items; we do not explicitly mention in the algorithm what data items are being discarded since it is well-defined under its operational mode.

We start with fixed order revocable priority algorithm by giving two tight bounds for non-increasing order (i.e. items are ordered so that $u_1 \geq u_2 \ldots \geq u_n$) and non-decreasing order revocable priority algorithms.

Theorem 2. *There is a non-increasing order revocable priority algorithm for SSP that has approximation ratio α, and every such algorithm has approximation ratio at most α. (Note that the simple ordering here is different from the fixed order irrevocable algorithm of Theorem 1.)*

Theorem 3. *There is a non-decreasing order revocable priority algorithm for SSP that has approximation ratio β, and every such algorithm has approximation ratio at most β.*

The improvement using non-decreasing order is perhaps counter-intuitive [8] as one might think it is more flexible to fill in with small items at the end. Next, we give a approximation bound for any fixed order revocable priority algorithm; this exhibits the first approximation gap we are unable to close. The technique used in the proof is based on a chain of possible item priorities. It turns out, in order to achieve certain approximation ratio, some data items must be placed before some other data items. This order relation is transitive and therefore, has to be acyclic.

Theorem 4. *No fixed order revocable priority algorithm of SSP can achieve approximation ratio better than γ.*

Proof. Let $u_1 = 0.2$, $u_2 = \frac{1}{2}\gamma - \frac{1}{10} \approx 0.326$, $u_3 = 0.5$, and $u_4 = 0.8$. For a data item u, we denote by $rank(u)$ its priority. There are four cases:

1. If $rank(u_4) > rank(u_3)$, then the adversarial strategy is shown in Fig. 1.

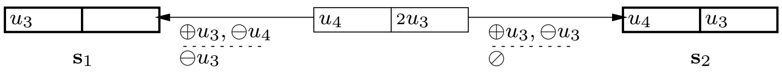

Fig. 1. Adversarial strategy for $rank(u_4) > rank(u_3)$.

If the algorithm terminates via state $\mathbf{s}_1$, then

$$\rho = \frac{\|\mathbf{ALG}\|}{\|\mathbf{OPT}\|} = \frac{u_3}{u_4} < \gamma.$$

If the algorithm terminates via state $\mathbf{s}_2$, then

$$\rho = \frac{\|\mathbf{ALG}\|}{\|\mathbf{OPT}\|} \le \frac{u_4}{2u_3} = u_4 < \gamma.$$

2. If $rank(u_3) > rank(u_2)$, then the adversarial strategy is shown in Fig. 2.

[8] As another example, in the identical machines makespan problem, it is provably advantageous to consider the largest items first.

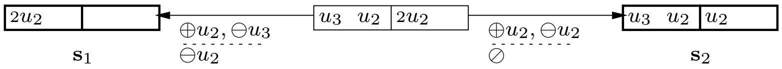

Fig. 2. Adversarial strategy for $rank(u_3) > rank(u_2)$.

If the algorithm terminates via state $\mathbf{s}_1$, then

$$\rho = \frac{\|\mathbf{ALG}\|}{\|\mathbf{OPT}\|} = \frac{2u_2}{u_2+u_3} = \frac{\gamma - \frac{1}{5}}{\frac{1}{2}+\frac{1}{2}\gamma - \frac{1}{10}} = \frac{10\gamma - 2}{5\gamma + 4} < \gamma.$$

If the algorithm terminates via state $\mathbf{s}_2$, then

$$\rho = \frac{\|\mathbf{ALG}\|}{\|\mathbf{OPT}\|} \leq \frac{u_2+u_3}{3u_2} = \frac{\frac{1}{2}+\frac{1}{2}\gamma - \frac{1}{10}}{\frac{3}{2}\gamma - \frac{3}{10}} = \frac{5\gamma + 4}{15\gamma - 3} < \gamma.$$

3. If $rank(u_2) > rank(u_1)$, then the adversarial strategy is shown in Fig. 3.

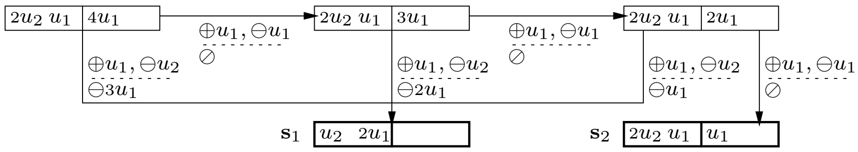

Fig. 3. Adversarial strategy for $rank(u_2) > rank(u_1)$.

If the algorithm terminates via state $\mathbf{s}_1$, then

$$\rho = \frac{\|\mathbf{ALG}\|}{\|\mathbf{OPT}\|} \leq \frac{2u_1+u_2}{u_1+2u_2} = \frac{\frac{2}{5}+\frac{1}{2}\gamma - \frac{1}{10}}{\frac{1}{5}+\gamma - \frac{1}{5}} = \frac{\frac{1}{2}\gamma + \frac{3}{10}}{\gamma} = \frac{5\gamma+3}{10\gamma} \leq \gamma.$$

If the algorithm terminates via state $\mathbf{s}_2$, then

$$\rho = \frac{\|\mathbf{ALG}\|}{\|\mathbf{OPT}\|} \leq \frac{u_1+2u_2}{5u_1} = u_1 + 2u_2 = \frac{1}{5} + \gamma - \frac{1}{5} = \gamma.$$

4. If $rank(u_1) > rank(u_2) > rank(u_3) > rank(u_4)$, then the adversarial strategy is shown in Fig. 4.
If the algorithm terminates via state $\mathbf{s}_1$, then

$$\rho = \frac{\|\mathbf{ALG}\|}{\|\mathbf{OPT}\|} = \frac{u_1+u_3}{u_2+u_3} = \frac{\frac{1}{5}+\frac{1}{2}}{\frac{1}{2}\gamma - \frac{1}{10} + \frac{1}{2}} = \frac{7}{5\gamma+4} < \gamma.$$

If the algorithm terminates via state $\mathbf{s}_2$, then

$$\rho = \frac{\|\mathbf{ALG}\|}{\|\mathbf{OPT}\|} \leq \frac{u_2+u_3}{u_1+u_4} = u_2 + u_3 = \frac{1}{2}\gamma - \frac{1}{10} + \frac{1}{2} < \gamma.$$

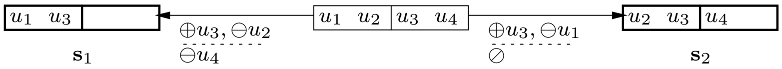

Fig. 4. Adversarial strategy for $rank(u_1) > rank(u_2) > rank(u_3) > rank(u_4)$.

As a conclusion, no fixed order revocable priority algorithm of SSP can achieve approximation ratio better than γ. This completes the proof. $\square$

Finally, we study adaptive revocable priority algorithms. This is the strongest class of algorithms studied in this paper and arguably represents the ultimate approximation power of greedy algorithms (for packing problems). We show that no such algorithm can achieve an approximation ratio better than δ, and then we develop a relatively subtle algorithm having approximation ratio 0.8 in Theorem 6, thus leaving another gap in what is provably the best approximation ratio possible.

Theorem 5. *No adaptive revocable priority algorithm of SSP can achieve approximation ratio better than* δ.

Proof. Let $u_1 = \frac{1}{3}\delta \approx 0.298$ and $u_2 = 0.5$. For a given algorithm, we utilize the following adversary strategy shown in Fig. 5. If the algorithm terminates via

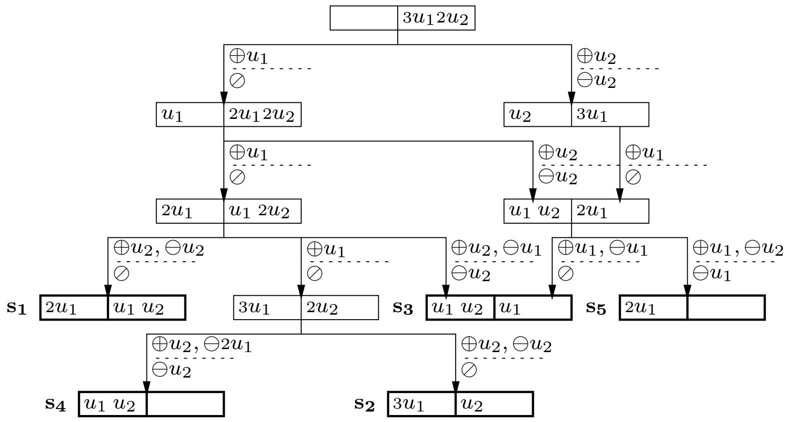

Fig. 5. Adversarial strategy for adaptive revocable priority algorithms.

state $\mathbf{s}_1$ or $\mathbf{s}_2$, then

$$\rho = \frac{\|\mathbf{ALG}\|}{\|\mathbf{OPT}\|} \le \frac{3u_1}{2u_2} = 3u_1 = \delta.$$

If the algorithm terminates via state $\mathbf{s}_3$ or $\mathbf{s}_4$, then

$$\rho = \frac{\|\mathbf{ALG}\|}{\|\mathbf{OPT}\|} \le \frac{u_1 + u_2}{3u_1} = \frac{\frac{1}{3}\delta + \frac{1}{2}}{\delta} = \frac{2\delta + 3}{6\delta} \le \delta.$$

If the algorithm terminates via state $\mathbf{s}_5$, then

$$\rho = \frac{\|\mathbf{ALG}\|}{\|\mathbf{OPT}\|} = \frac{2u_1}{u_1 + u_2} = \frac{\frac{2}{3}\delta}{\frac{1}{3}\delta + \frac{1}{2}} = \frac{4\delta}{2\delta + 3} < \delta.$$

In all three cases, the adversary forces the algorithm to have approximation ratio no better than δ; this completes the proof. □

We now show an adaptive revocable priority algorithm which achieves approximation ratio 0.8 . The algorithm uses an ordering of data items which is determined by its distance to 0.3, i.e., the closer a data item to 0.3, the higher its priority is, breaking tie arbitrarily. Note that this ordering does not make the algorithm fixed order priority, as this ordering may be interrupted if at any point of time a terminal condition is satisfied. The algorithm is described below.

Algorithm ADAPTIVE

B $:= \emptyset$;
if the first data item is in $(0.2, 0.35)$ **then**
  $\mathbf{B}_{\text{mode}} :=$ `Queue`;
else
  $\mathbf{B}_{\text{mode}} :=$ `Queue_1`;
end if
while I contains a data item $\in (0.2, 0.4]$ **do**
  let u be the next data item in **I**;
  $\mathbf{I} := \mathbf{I} \ominus u$;
  accept u;
end while
if B contains exactly three data items and all are $\in (0.2, 0.3]$ **then**
  $\mathbf{B}_{\text{mode}} :=$ `Stack`;
else
  $\mathbf{B}_{\text{mode}} :=$ `Optimum`;
end if
while I contains a data item $\in (0.4, 0.8)$ **do**
  let u be the next data item in **I**;
  $\mathbf{I} := \mathbf{I} \ominus u$;
  accept u;
end while

Theorem 6. *Algorithm* ADAPTIVE *achieves approximation ratio 0.8 for SSP.*

It is seemingly a small step from a 0.78 algorithm to a 0.8 algorithm, but the latter algorithm requires a substantially more refined approach and detalied analysis. The merit, we believe, in studying such a class of "simple algorithms" is that the simplicity of the structure suggests algorithmic ideas and allows a careful analysis of such algorithms. Limiting ourselves to simple algorithmic forms and exploiting the flexibility within such forms may very well give us a better understanding of the structure of a given problem and a better chance to derive new algorithms.

5 Conclusion

We analyze different types of priority algorithms for SSP leaving open two approximability gaps, one for fixed order and one for adaptive revocable priority algorithms. It is interesting that such gaps and non-trivial algorithms exist for such a simple class of algorithms and such a basic problem as SSP. We optimistically believe that surprisingly good algorithms can be designed within the revocable priority framework for problems which are currently not well understood.

References

1. Angelopoulos, S. and Borodin, A.: On the power of priority algorithms for facility location and set cover. Algorithmica **40** (2004) 271–291
2. Arrow, K.: Social Choice and Individual Values. Wiley (1951)
3. Baptiste, P.: Polynomial time algorithms for minimizing the weighted number of late jobs on a single machine with equal processing times. Journal of Scheduling **2** (1999) 245–252
4. Bar-Noy, A., Guha, S., Naor, J. and Schieber, B.: Approximating throughput in real-time scheduling. SIAM Journal of Computing **31** (2001) 331–352
5. Borodin, A., Nielsen, M., and Rackoff C.: (Incremental) priority algorithms. Algorithmica **37** (2003) 295–326
6. Borodin, A., Boyar, J., and Larsen K.: Priority algorithms for graph optimization problems. Lecture Notes in Computer Science **3351** (2005) 126–139
7. Chrobak, M., Durr, C., Jawor, W., Kowalik, L. and Kurowski, M. On scheduling equal length jobs to maximize throughput. To appear in Journal of Scheduling.
8. Chuzhoy, J., Ostrovsky, R. and Rabani, Y.: Approximation algorithms for the job interval scheduling problem and related scheduling problems. In Proceedings of 42nd Annual IEEE Symposium of Foundations of Computer Science (2001) 348–356
9. Davis, S. and Impagliazzo, R.: Models of greedy algorithms for graph problems. In Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (2004) 381–390
10. Erlebach, T. and Spieksma, F.: Interval selection: Applications, algorithms, and lower bounds. Journal of Algorithms **46** (2003) 27–53
11. Garey, M. and Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-completeness. Freeman (1979)
12. Horn, S.: One-pass algorithms with revocable acceptances for job interval selection. Master's thesis, University of Toronto (2004)
13. Ibarra, O. and Kim, C.: Fast approximation algorithms for the knapsack and sum of subset problem. Journal of the ACM **22** (1975) 463–468

14. Iwama, K. and Taketomi, S.: Removable online knapsack problems. Lecture Notes in Computer Science **2380** (2002) 293–305
15. Kellerer, H., Mansini, R., Pferschy, U. and Speranza, M.: An efficient fully polynomial approximation scheme for the subset-sum problem. Journal of Computer and System Science **66** (2003) 349–370
16. Martello, S. and Toth, P.: Knapsack Problems: Algorithms and Computer Implementations. John Wiley and Sons (1990)
17. Moore, J.: An n-job, one machine sequencing algorithm algorithm for minimizing the number of late jobs. Management Science **15** (1968) 102–109
18. Regev, O.: Priority algorithms for makespan minimization in the subset model. Information Processing Letters **84** (2002) 153–157

A Proof of Theorem 1

Lemma 1. *No adaptive irrevocable priority algorithm of SSP can achieve approximation ratio better than* α.

Proof. We show that for any irrevocable priority algorithm and every small $\epsilon > 0$, $\rho < \alpha + \epsilon$. Let

$$\begin{array}{ll} u_1 = \alpha \approx 0.657, & u_2 = 2\alpha^3 + \alpha^2\epsilon \approx 0.568, \\ u_3 = \alpha^2 \approx 0.432, & u_4 = \alpha^2 - \alpha^2\epsilon \approx 0.432; \end{array}$$

then for any algorithm, the adversary presents one copy of u_1, u_2, u_4 and two copies of u_3 to the algorithm. Let u be the data item with highest priority among these four types, then if the algorithm discards u on its first decision, then the adversary can remove the rest of the data items and force infinite approximation ratio. We therefore assume that the algorithm always takes the first data item into its solution, then depending on which is the first data item the algorithm selects, there are four cases:

1. If $u = u_1$, then the adversary removes two copies of u_3; since $u_1 + u_2 > 1$ and $u_1 + u_4 > 1$, the algorithm can take neither of the two, hence $\|\textbf{ALG}\| = u_1$. On the other hand, $u_2 + u_4 = 1$, hence $\|\textbf{OPT}\| = u_2 + u_4$. Therefore, the approximation ratio is

$$\rho = \frac{\|\textbf{ALG}\|}{\|\textbf{OPT}\|} = \frac{u_1}{u_2 + u_4} = \frac{\alpha}{2\alpha^3 + \alpha^2} = \alpha.$$

2. If $u = u_2$, then the adversary removes one copy of u_1 and one copy of u_4; since $u_2 + u_3 > 1$, the algorithm can take neither of the two, hence $\|\textbf{ALG}\| = u_2$. On the other hand, $u_2 < 2u_3 < 1$, hence $\|\textbf{OPT}\| = 2u_3$. Therefore, the approximation ratio is

$$\rho = \frac{\|\textbf{ALG}\|}{\|\textbf{OPT}\|} = \frac{u_2}{2u_3} = \frac{2\alpha^3 + \alpha^2\epsilon}{2\alpha^2} = \alpha + \frac{\epsilon}{2}.$$

3. If $u = u_3$, then the adversary removes one copy of u_2, u_3 and u_4; since $u_1 + u_3 > 1$, the algorithm cannot take u_1, hence $\|\textbf{ALG}\| = u_3$. On the other hand, $u_3 < u_1 < 1$, hence $\|\textbf{OPT}\| = u_1$. Therefore, the approximation ratio is

$$\rho = \frac{\|\textbf{ALG}\|}{\|\textbf{OPT}\|} = \frac{u_3}{u_1} = \frac{\alpha^2}{\alpha} = \alpha.$$

4. If $u = u_4$, then the adversary removes two copies of u_3; since $u_1 + u_4 > 1$, the algorithm cannot take u_1, hence $\|\textbf{ALG}\| = u_4$. On the other hand, $u_4 < u_1 < 1$, hence $\|\textbf{OPT}\| = u_1$. Therefore, the approximation ratio is

$$\rho = \frac{\|\textbf{ALG}\|}{\|\textbf{OPT}\|} = \frac{u_4}{u_1} = \frac{\alpha^2 - \alpha^2\epsilon}{\alpha} < \frac{\alpha^2}{\alpha} = \alpha.$$

Therefore, in all cases, $\rho < \alpha + \epsilon$; this completes the proof. $\square$

We now give a fixed irrevocable priority algorithm that achieves approximation of α. As justified earlier in Section 3, we only consider relevant data items for the input set; we partition all possible relevant data items into two sets: **M** and **L**. A data item u is said to be in class **M** and **L** if $1-\alpha < u \le \alpha^2$ and $\alpha^2 < u < \alpha$ respectively; see Fig. 6. We specify the ordering of data items for the fixed

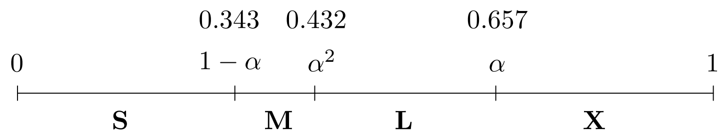

Fig. 6. Classification for the fixed irrevocable priority algorithm.

irrevocable priority algorithm: it first orders data items in **L** non-decreasingly, and then data items in **M** non-decreasingly. The algorithm, which is described below, uses this ordering to achieve the approximation ratio α.

Algorithm 1

B $:= \emptyset$;
for $i := 1$ to n **do**
    let u_i be the next data item in **I**;
    $\mathbf{I} := \mathbf{I} \ominus u_i$;
    if $\|\mathbf{B}\| + u_i \le 1$ **then**
        $\mathbf{B} := \mathbf{B} \oplus u_i$;
    end if
end for

Lemma 2. *There is a fixed order irrevocable priority algorithm for SSP with approximation ratio α.*

Proof. It is sufficient to show that Alg. 1 achieves this approximation ratio. Note that if ALG contains more than one data item, the $\|\text{ALG}\| > \frac{2}{3} > \alpha$; hence the algorithm achieves approximation ratio α. Suppose now ALG contains only one data item u_j. If $u_j \in \mathbf{M}$, then $|\mathbf{L} \cap \mathbf{I}| = 0$ and $|\mathbf{M} \cap \mathbf{I}| = 1$. Therefore, we have $\|\text{OPT}\| = \|\text{ALG}\| = u_j$; the algorithm achieves approximation ratio 1. Suppose now $u_j \in \mathbf{L}$, we then consider data items in OPT. Note that OPT can contain at most two data items; there are five cases:

1. If $\textsc{opt}$ contains exactly one data item $u_r \in \mathbf{M}$, then $|\mathbf{L} \cap \mathbf{I}| = 0$ and hence $u_j \in \mathbf{M}$. Since $u_j \in \mathbf{L}$, this is a contradiction; this case is impossible.
2. If $\textsc{opt}$ contains exactly one data item $u_r \in \mathbf{L}$, then the approximation ratio is
$$\rho = \frac{\|\textsc{alg}\|}{\|\textsc{opt}\|} = \frac{u_j}{u_r} > \frac{\alpha^2}{\alpha} = \alpha.$$
3. If $\textsc{opt}$ contains two data items $u_r, u_s \in \mathbf{M}$ with $u_r \leq u_s$, then we have $u_j + u_s > 1$. Therefore, the approximation ratio is
$$\rho = \frac{\|\textsc{alg}\|}{\|\textsc{opt}\|} = \frac{u_j}{u_r + u_s} > \frac{1 - u_s}{u_r + u_s} > \frac{1 - \alpha^2}{2\alpha^2} = \frac{2\alpha^3}{2\alpha^2} = \alpha.$$
4. If $\textsc{opt}$ contains two data items u_r, u_s with $u_r \in \mathbf{M}$ and $u_s \in \mathbf{L}$, then $\textsc{alg}$ should have contained the smallest data item in $\mathbf{M} \cap \mathbf{I}$ and the smallest data item in $\mathbf{L} \cap \mathbf{I}$, or the smallest two data items in $\mathbf{L} \cap \mathbf{I}$, and hence a contradiction; this case is impossible.
5. If $\textsc{opt}$ contains two data items $u_r, u_s \in \mathbf{L}$ with $u_r \leq u_s$, then $\textsc{alg}$ should have contained the smallest two data items in $\mathbf{L} \cap \mathbf{I}$, and hence a contradiction; this case is impossible.

Therefore, Alg. 1 achieves the approximation ratio α. □

Theorem 1 is immediate by Lemmas 1 and 2.

B Proof of Theorem 2

Lemma 3. *No non-increasing order revocable priority algorithm of SSP can achieve approximation ratio better than α.*

Proof. We show that for any non-increasing order revocable priority algorithm and every small $\epsilon > 0$, $\rho < \alpha + \epsilon$. Let

$$u_1 = \alpha \approx 0.657, \qquad u_2 = 2\alpha^3 + \alpha^2\epsilon \approx 0.568,$$
$$u_3 = \alpha^2 \approx 0.432, \qquad u_4 = \alpha^2 - \alpha^2\epsilon \approx 0.432;$$

we utilize the following adversarial strategy shown in Fig. 7.

If the algorithm terminates via state $\mathbf{s}_1$, then $\|\textsc{alg}\| \leq u_1$. Since $u_2 + u_4 = 2\alpha^3 + \alpha^2 = 1$, we have $\|\textsc{opt}\| = u_2 + u_4$. Therefore, the approximation ratio is

$$\rho = \frac{\|\textsc{alg}\|}{\|\textsc{opt}\|} \leq \frac{u_1}{u_2 + u_4} = \frac{\alpha}{2\alpha^3 + \alpha^2} = \alpha.$$

If the algorithm terminates via state $\mathbf{s}_2$, then $\|\textsc{alg}\| \leq u_2$ and $\|\textsc{opt}\| = 2u_3$. Therefore, the approximation ratio is

$$\rho = \frac{\|\textsc{alg}\|}{\|\textsc{opt}\|} \leq \frac{u_2}{2u_3} = \frac{2\alpha^3 + \alpha^2\epsilon}{2\alpha^2} = \alpha + \frac{1}{2}\epsilon.$$

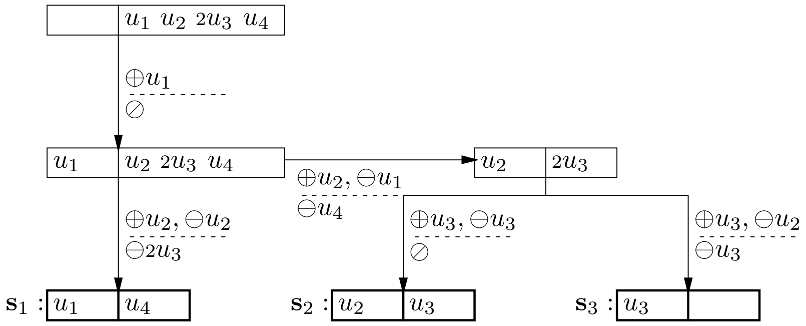

Fig. 7. Adversarial strategy for fixed non-increasing order.

If the algorithm terminates via state $\mathbf{s}_3$, then $\|\mathbf{ALG}\| = u_3$ and $\|\mathbf{OPT}\| = u_1$. Therefore, the approximation ratio is

$$\rho = \frac{\|\mathbf{ALG}\|}{\|\mathbf{OPT}\|} = \frac{u_3}{u_1} = \frac{\alpha^2}{\alpha} = \alpha.$$

In all three cases, the adversary forces the algorithm to have approximation ratio $\rho < \alpha + \epsilon$; this completes the proof. $\square$

We now give a non-increasing order revocable priority algorithm that achieves the approximation ratio α. We use the same partition as shown in Fig. 6; the algorithm is described below.

Algorithm 2

$\mathbf{B} := \emptyset$;
$\mathbf{B}_{\text{mode}} :=$ `Queue`;
for $i := 1$ to n **do**
  let u_i be the next data item in $\mathbf{I}$;
  $\mathbf{I} := \mathbf{I} \ominus u_i$;
  if $\|\mathbf{B}\| + u_i \leq 1$ **then**
    accept u_i;
  else if $u_i \in \mathbf{L}$ **then**
    accept u_i;
  else
    reject u_i;
  end if
end for

Lemma 4. *There is a non-increasing order revocable priority algorithm of SSP with approximation ratio α.*

Proof. It is sufficient to show that Alg. 2 achieves this approximation ratio. We prove this by induction on i. At the end of the i^{th} iteration of the **for** loop, we let $\textbf{ALG}_i$ and $\textbf{OPT}_i$ be the algorithm's solution and the optimal solution respectively; these are solutions with respect to the first i data items $\{u_1, \ldots, u_i\}$. We also let ρ_i to be the current approximation ratio,

$$\rho_i = \frac{\|\textbf{ALG}_i\|}{\|\textbf{OPT}_i\|}.$$

It is clear that at the end of the first iteration, the algorithm has approximation ratio $\rho_1 = 1 > \alpha$. Suppose there is no early termination and the α-approximation is maintained for the first $k-1$ iterations, i.e., $\|\textbf{ALG}_i\| < \alpha$ and $\rho_i \geq \alpha$ for all $i < k$; we examine the case for $i = k$, $k \geq 2$. Note that $|\textbf{ALG}_i| = 1$ for all $i < k$, for otherwise $\|\textbf{ALG}_i\| \geq \alpha$. The algorithm examines data items in non-increasing order; for data items in $\mathbf{L}$, it always accepts the current one into the solution, no matter whether or not the data item fits; for data items in $\mathbf{M}$, it always rejects the current one if it does not fit. In that sense, it tends to keep the smallest data item in $\mathbf{L} \cap \mathbf{I}$ and the largest data item in $\mathbf{M} \cap \mathbf{I}$. There are only three possibilities for u_k during the k^{th} iteration; it must fall into one of the lines 6, 8 and 10:

1. **Line 6:** then the algorithm accepts u_k into the solution without discarding any accepted data item in the solution. Hence we have $|\textbf{ALG}_i| = 2$, the terminal condition is satisfied. Therefore, the algorithm terminates and achieves approximation ratio α.
2. **Line 8:** then at the end of $(k-1)^{\text{th}}$ iteration, $\textbf{ALG}_{k-1}$ contains only u_{k-1}. Since the terminal condition is not satisfied during the k^{th} iteration, we have $u_{k-1} + u_k > 1$. Therefore, the algorithm accepts u_k into the solution by discarding u_{k-1}. Note that u_k and u_{k-1} are the two smallest data items examined so far and $u_{k-1} + u_k > 1$, hence $\textbf{OPT}_k$ contains exactly one data item, say u_r, in $\mathbf{L}$. Therefore, the approximation ratio is
$$\rho_k = \frac{\|\textbf{ALG}_k\|}{\|\textbf{OPT}_k\|} = \frac{u_k}{u_r} \geq \frac{\alpha^2}{\alpha} = \alpha.$$
3. **Line 10:** then $u_k \in \mathbf{M}$ and the algorithm rejects u_k, therefore $\textbf{ALG}_k = \textbf{ALG}_{k-1}$. Furthermore, at the end of $(k-1)^{\text{th}}$ iteration, $\textbf{ALG}_{k-1}$ contains exactly one data item. Observe that it cannot be a data item in $\mathbf{M}$; for otherwise, the terminal condition is satisfied. Therefore, $\textbf{ALG}_{k-1}$ contains exactly one data item, say u_j, in $\mathbf{L}$. Since the terminal condition is not satisfied during the k^{th} iteration, we have $u_k + u_j > 1$; therefore,
$$\|\textbf{ALG}_k\| = \|\textbf{ALG}_{k-1}\| = u_j > 1 - u_k \geq 1 - \alpha^2 = 2\alpha^3.$$
Note that for all the data items examined so far, u_k is the smallest in $\mathbf{M}$, u_j is the smallest in $\mathbf{L}$, and $u_k + u_j > 1$, hence $\textbf{OPT}_k$ can either contain

one data item in $\mathbf{L}$ or two data items in $\mathbf{M}$; in both cases $\|\mathbf{OPT}_k\| < 2\alpha^2$. Therefore, the approximation ratio is

$$\rho_k = \frac{\|\mathbf{ALG}_k\|}{\|\mathbf{OPT}_k\|} > \frac{2\alpha^3}{2\alpha^2} = \alpha.$$

We now have shown that in all three cases, the algorithm maintains the approximation ratio $\rho_i \geq \alpha$ for $i = k$; this completes the induction. Therefore, Alg. 2 achieves approximation ratio α. □

Theorem 2 is immediate by Lemmas 3 and 4.

C Proof of Theorem 3

Lemma 5. *No non-decreasing order revocable priority algorithm of SSP can achieve approximation ratio better than β.*

Proof. We show that for any non-decreasing order revocable priority algorithm and every small $\epsilon > 0$, $\rho < \beta + \epsilon$. Let

$$u_1 = 1 - \beta \approx 0.220,\ u_2 = \tfrac{1}{2}\beta + \tfrac{1}{4}\epsilon \approx 0.390,\ u_3 = \beta \approx 0.780;$$

we utilize the following adversarial strategy shown in Fig. 8.

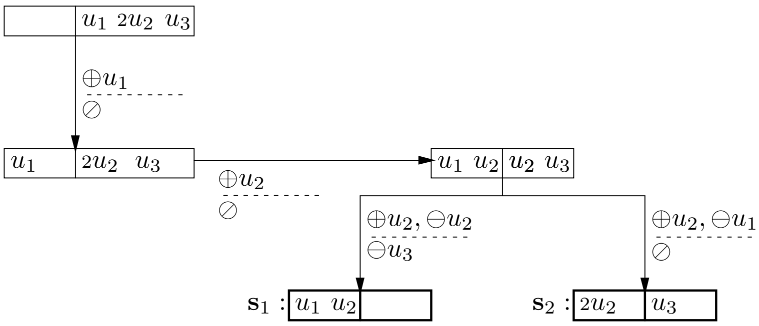

Fig. 8. Adversarial strategy for fixed non-decreasing order.

If the algorithm terminates via state $\mathbf{s}_1$, then $\|\mathbf{ALG}\| = u_1 + u_2$ and $\|\mathbf{OPT}\| = 2u_2$. Note that $u_1 + u_2 = 1 - \frac{1}{2}\beta + \frac{1}{4}\epsilon = \beta^2 + \frac{1}{4}\epsilon$. Therefore, the approximation ratio is

$$\rho = \frac{\|\mathbf{ALG}\|}{\|\mathbf{OPT}\|} = \frac{u_1 + u_2}{2u_2} = \frac{1 - \frac{1}{2}\beta + \frac{1}{4}\epsilon}{\beta + \frac{1}{2}\epsilon} = \frac{\beta^2 + \frac{1}{4}\epsilon}{\beta + \frac{1}{2}\epsilon} < \beta.$$

If the algorithm terminates via state $\mathbf{s}_2$, then $\|\textbf{ALG}\| \le 2u_2$. Since $u_1 + u_3 = 2 = 1$, we have $\|\textbf{OPT}\| = u_1 + u_3$. Therefore, the approximation ratio is

$$\rho = \frac{\|\textbf{ALG}\|}{\|\textbf{OPT}\|} \le \frac{2u_2}{u_1 + u_3} = \beta + \frac{1}{2}\epsilon.$$

In both cases, the adversary forces the algorithm to have approximation ratio $\rho < \beta + \epsilon$; this completes the proof. $\square$

We now give a non-decreasing order revocable priority algorithm that achieves the approximation ratio β. For all possible relevant data items, we partition them into two sets: **M** and **L**. A data item u is said to be in class **M** and **L** if $1 - \beta < u \le \frac{1}{2}\beta$ and $\frac{1}{2}\beta < u < \beta$ respectively; see Fig. 9.

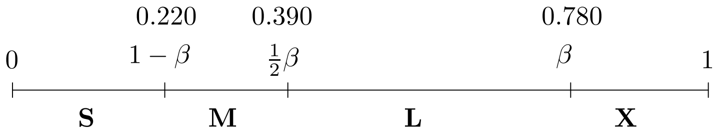

Fig. 9. Classification for non-decreasing order.

The algorithm achieves this ratio is surprisingly simple; it basically set $\mathbf{B}_{\text{mode}}$ to `Stack` mode, and accepts each data item in turn; see Alg. 3.

Algorithm 3

$\mathbf{B} := \emptyset$;
$\mathbf{B}_{\text{mode}} :=$ `Stack`;
for $i := 1$ to n **do**
  let u_i be the next data item in $\mathbf{I}$;
  $\mathbf{I} := \mathbf{I} \ominus u_i$;
  accept u_i;
end for

Lemma 6. *There is a non-decreasing order revocable priority algorithm of SSP with approximation ratio β.*

Proof. It is sufficient to show that Alg. 3 achieves this approximation ratio. We prove this by induction on i. At the end of the i^{th} iteration of the **for** loop, we let

$\textbf{ALG}_i$ and $\textbf{OPT}_i$ be the algorithm's solution and the optimal solution respectively; and let ρ_i be the current approximation ratio. It is clear that at the end of the first iteration, the algorithm has approximation ratio $\rho_1 = 1 > \beta$. Suppose there is no early termination and the approximation ratio $\rho_i \geq \beta$ is maintained for all $i < k$, we examine the case for $i = k$. There are two possibilities. The first case is that $\|\mathbf{B}\| + u_k \leq 1$, then the algorithm accepts u_k into the solution without discarding any accepted data item in the solution. Hence we have $\|\textbf{OPT}_k\| \geq \|\textbf{OPT}_{k-1}\| + u_k$; for otherwise, we can take $\textbf{OPT}_{k-1} \oplus u_k$ as $\textbf{OPT}_k$, which leads to a contradiction. Therefore, the approximation ratio is

$$\rho_k = \frac{\|\textbf{ALG}_k\|}{\|\textbf{OPT}_k\|} \geq \frac{\|\textbf{ALG}_{k-1}\| + u_k}{\|\textbf{OPT}_{k-1}\| + u_k} \geq \frac{\|\textbf{ALG}_{k-1}\|}{\|\textbf{OPT}_{k-1}\|} = \rho_{k-1} \geq \beta.$$

This leaves the case when $\|\mathbf{B}\| + u_k > 1$. Note that the algorithm examines data items in non-decreasing order; u_1 is the smallest data item in $\mathbf{I}$ and u_2 is the second smallest. These two data items are at the bottom of the stack, hence tend to stay in the stack unless we see a data item with sufficient large weight:

– **Observation 1:** if at certain stage of the algorithm, after accepting u_k, there is only one data item $u_k \neq u_1$ in the stack, then we have $u_1 + u_k > 1$; see Fig. 10.

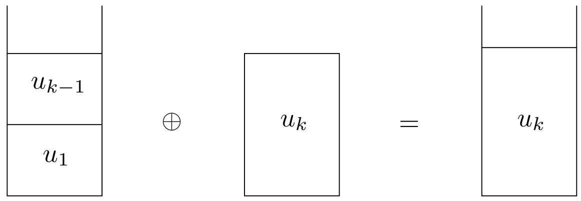

Fig. 10. Stack with one data item.

– **Observation 2:** if at certain stage of the algorithm, after accepting u_k, there are only two data items in the stack and the top of the stack is $u_k \neq u_2$, then we have $u_1 + u_2 + u_k > 1$; see Fig. 11.

The above two inequalities are useful in our analysis. Now we consider data items in $\textbf{OPT}_k$; there are six cases:

1. If $|\textbf{OPT}_k| = 1$, then $\textbf{OPT}_k = \{u_k\}$. Since the algorithm always accepts the current data item, we have $u_k \in \textbf{ALG}_k$, and hence $\textbf{ALG}_k = \textbf{OPT}_k$. Therefore, the approximation ratio is

$$\rho_k = \frac{\|\textbf{ALG}_k\|}{\|\textbf{OPT}_k\|} = 1.$$

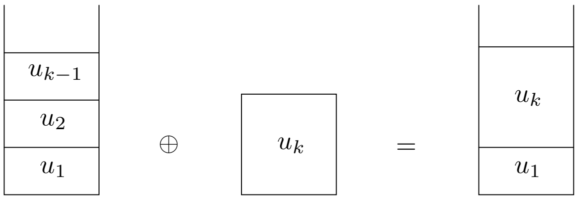

Fig. 11. Stack with two data items.

2. If $\textbf{OPT}_k$ contains exactly two data items $u_r, u_s \in \mathbf{M}$ with $u_r \le u_s$, then we have both u_1 and u_2 in M; we now consider data items in $\textbf{ALG}_k$.
   (a) If $|\textbf{ALG}_k| = 1$, then $\textbf{ALG}_k = \{u_k\}$; by Observation 1, we have
   $$u_1 + u_k > 1.$$
   Since $u_r + u_s \le \beta$, we have $1 - u_1 \ge 1 - \frac{1}{2}\beta = \beta^2$. Therefore, the approximation ratio is
   $$\rho_k = \frac{\|\textbf{ALG}_k\|}{\|\textbf{OPT}_k\|} = \frac{u_k}{u_r + u_s} > \frac{1 - u_1}{u_r + u_s} \ge \frac{\beta^2}{\beta} = \beta.$$
   (b) If $|\textbf{ALG}_k| = 2$, then $\textbf{ALG}_k = \{u_1, u_k\}$; by Observation 2, we have
   $$u_1 + u_2 + u_k > 1.$$
   Since $u_r + u_s \le \beta$, we have $1 - u_2 \ge 1 - \frac{1}{2}\beta = \beta^2$. Therefore, the approximation ratio is
   $$\rho_k = \frac{\|\textbf{ALG}_k\|}{\|\textbf{OPT}_k\|} = \frac{u_1 + u_k}{u_r + u_s} > \frac{1 - u_2}{u_r + u_s} \ge \frac{\beta^2}{\beta} = \beta.$$
   (c) If $|\textbf{ALG}_k| \ge 3$, then $\{u_1, u_2, u_k\} \in \textbf{ALG}$. Note that $u_1 + u_2 > 2 - 2\beta$. Therefore, the approximation ratio is
   $$\rho_k = \frac{\|\textbf{ALG}_k\|}{\|\textbf{OPT}_k\|} \ge \frac{u_1 + u_2 + u_k}{u_r + u_s} > \frac{2 - 2\beta + u_s}{\frac{1}{2}\beta + u_s} > 1,$$
   which is a contradiction; this case is impossible.
3. If $\textbf{OPT}_k$ contains exactly two data items $u_r \in \mathbf{M}$ and $u_s \in \mathbf{L}$, then we have u_1 in M; we now consider data items in $\textbf{ALG}_k$.
   (a) If $|\textbf{ALG}_k| = 1$, then $\textbf{ALG}_k = \{u_k\}$; by Observation 1, we have
   $$u_1 + u_k > 1. \tag{3}$$
   Note that $u_1 + u_s \le u_r + u_s \le 1$. When the algorithm considers u_s, u_1 is already in the solution. Since the terminal condition is not satisfied, we have
   $$u_1 + u_s < \beta. \tag{4}$$

Combining (3) and (4), we have $u_k > 1 - \beta + u_s$. Therefore,

$$\rho_k = \frac{\|\mathbf{ALG}_k\|}{\|\mathbf{OPT}_k\|} = \frac{u_k}{u_r + u_s} > \frac{1 - \beta + u_s}{\frac{1}{2}\beta + u_s} > \beta.$$

(b) If $|\mathbf{ALG}_k| = 2$, then $\mathbf{ALG}_k = \{u_1, u_k\}$. Therefore

$$\rho_k = \frac{\|\mathbf{ALG}_k\|}{\|\mathbf{OPT}_k\|} = \frac{u_1 + u_k}{u_r + u_s} > \frac{1 - \beta + u_s}{\frac{1}{2}\beta + u_s} > \beta.$$

(c) If $|\mathbf{ALG}_k| \geq 3$, then $\{u_1, u_2, u_k\} \in \mathbf{ALG}$. Note that $u_1 + u_2 > 2 - 2\beta$. Therefore, the approximation ratio is

$$\rho_k = \frac{\|\mathbf{ALG}_k\|}{\|\mathbf{OPT}_k\|} \geq \frac{u_1 + u_2 + u_k}{u_r + u_s} > \frac{2 - 2\beta + u_s}{\frac{1}{2}\beta + u_s} > 1,$$

which is a contradiction; this case is impossible.

4. If $\mathbf{OPT}_k$ contains exactly two data items $u_r, u_s \in \mathbf{L}$ with $u_r \leq u_s$, then we have

$$\beta \leq u_r + u_{r+1} \leq u_r + u_s \leq 1.$$

When the algorithm considers u_{r+1}, u_r is already in the solution; hence the terminal condition is satisfied.

5. If $\mathbf{OPT}_k$ contains exactly three data items $u_r, u_s, u_t \in \mathbf{M}$ with $u_r \leq u_s \leq u_t$, then we have both u_1 and u_2 in M; we now consider data items in $\mathbf{ALG}_k$.

(a) If $|\mathbf{ALG}_k| = 1$, then $\mathbf{ALG}_k = \{u_k\}$; by Observation 1, we have

$$u_1 + u_k > 1. \tag{5}$$

Note that $u_1 + u_r + u_s \leq u_1 + u_s + u_{s+1} \leq u_r + u_s + u_t \leq 1$. When the algorithm considers u_{s+1}, u_1 and u_s are already in the solution. Since the terminal condition is not satisfied, we have $u_1 + u_s + u_{s+1} < \beta$. Therefore,

$$u_1 + u_r + u_s < \beta. \tag{6}$$

Combining (5) and (6), we have $u_k > 1 - \beta + u_r + u_s$. Since both u_r and u_s are in $\mathbf{M}$, we have $u_r + u_s > 2 - 2\beta > \frac{1}{2}\beta$. Therefore,

$$\rho_k = \frac{\|\mathbf{ALG}_k\|}{\|\mathbf{OPT}_k\|} = \frac{u_k}{u_r + u_s + u_t} > \frac{1 - \beta + (u_r + u_s)}{\frac{1}{2}\beta + (u_r + u_s)} > \beta.$$

(b) If $|\mathbf{ALG}_k| = 2$, then $\mathbf{ALG}_k = \{u_1, u_k\}$; by Observation 2, we have

$$u_1 + u_2 + u_k > 1. \tag{7}$$

– If $r = 1$, then $u_2 + u_r + u_s \leq u_1 + u_2 + u_{s+1} \leq u_r + u_s + u_t \leq 1$. When the algorithm considers u_{s+1}, u_1 and u_2 are already in the solution. Since the terminal condition is not satisfied, we have $u_1 + u_2 + u_{s+1} < \beta$. Therefore,

$$u_2 + u_r + u_s < \beta. \tag{8}$$

– If $r > 1$, then $u_2+u_r+u_s \leq u_2+u_s+u_{s+1} \leq u_r+u_s+u_t \leq 1$. When the algorithm considers u_{s+1}, u_2 and u_s are already in the solution. Since the terminal condition is not satisfied, we have $u_2+u_s+u_{s+1} < \beta$. Therefore, the inequality (8) is also satisfied.

Therefore, in both cases we have (8). Combining (7) and (8), we have $u_1 + u_k > 1 - \beta + u_r + u_s$. Since both u_r and u_s are in $\mathbf{M}$, we have $u_r + u_s > 2 - 2\beta > \frac{1}{2}\beta$. Therefore,

$$\rho_k = \frac{\|\textbf{ALG}_k\|}{\|\textbf{OPT}_k\|} = \frac{u_1 + u_k}{u_r + u_s + u_t} > \frac{1 - \beta + (u_r + u_s)}{\frac{1}{2}\beta + (u_r + u_s)} > \beta.$$

(c) If $|\textbf{ALG}_k| \geq 3$, then $\{u_1, u_2, u_k\} \subseteq \textbf{ALG}$. Note that $u_1 + u_r + u_s \leq u_1 + u_s + u_{s+1} \leq u_r + u_s + u_t \leq 1$. When the algorithm considers u_{s+1}, u_1 and u_s are already in the solution. Since the terminal condition is not satisfied, we have $u_1 + u_s + u_{s+1} < \beta$. Therefore,

$$u_1 + u_r + u_s < \beta. \tag{9}$$

Then we have $u_r + u_s < \beta - u_1 < \beta - (1 - \beta) = 2\beta - 1$. Note that $u_1 + u_2 > 2 - 2\beta$. Therefore,

$$\rho_k = \frac{\|\textbf{ALG}_k\|}{\|\textbf{OPT}_k\|} \geq \frac{u_1 + u_2 + u_k}{u_r + u_s + u_t} > \frac{2 - 2\beta + u_t}{2\beta - 1 + u_t} > \beta.$$

6. If $\textbf{OPT}_k$ contains exactly three data items $u_r, u_s \in \mathbf{M}$ and $u_t \in \mathbf{L}$ with $u_r \leq u_s$, then we have

$$\beta < 2 - 2\beta + \frac{1}{2}\beta < u_1 + u_2 + u_t \leq u_r + u_s + u_t \leq 1.$$

When the algorithm considers u_t, u_1 and u_2 are already in the solution; hence the terminal condition is satisfied.

We now have shown that in all six cases, the algorithm maintains the approximation ratio $\rho_i \geq \beta$ for $i = k$; this completes the induction. Therefore, Alg. 3 achieves approximation ratio β. □

Theorem 3 is immediate by Lemmas 5 and 6.

D Proof of Theorem 6

For notational convenience, we use the following symbols to denote data items in different classes; see Table 2. The algorithm can be divided into two stages. In the first stage (while loop) data items of type $\lozenge$ are processed; in the second stage (while loop) data items of type $\blacklozenge$ are processed. *For a given instance, we now assume that the algorithm* ADAPTIVE *achieves approximation ratio less than 0.8 and try to derive a contradiction.* First, we let $\mathbf{S} = u_1, u_2, \ldots, u_n$ be the input sequence of data items according to the algorithm's ordering, and let $\mathbf{S}_1 = u_1, u_2, \ldots, u_m$ and $\mathbf{S}_2 = u_{m+1}, u_{m+2}, \ldots, u_n$ be the input sequences of data items for the first stage and second stage respectively.

Table 2. Corresponding symbols.

class	$(0.2, 0.4]$	$(0.4, 0.8)$	$(0.2, 0.3]$	$(0.3, 0.4]$
symbol	$\lozenge$	$\blacklozenge$	$\triangledown$	$\vartriangle$

Lemma 7. *Let u be the next data item to be examined and $\mathbf{B}' = (\mathbf{B} \oplus u)$. If $\mathbf{B}'$ contains three data items u_i, u_j, u_k, with $i < j < k$, then (u_i, u_j, u_k) cannot be any of the following types:*

$$\begin{array}{ll} (\triangledown, \triangledown, \vartriangle), & (\triangledown, \vartriangle, \triangledown), \\ (\vartriangle, \vartriangle, \triangledown), & (\vartriangle, \triangledown, \vartriangle); \end{array}$$

Proof. This is immediate as the total weight of those triples are in $[0.8, 1]$. Since at each step, the algorithm adaptively checks for a terminal condition, it can detect those triples and terminate early. $\square$

Lemma 8. *The input sequence $\mathbf{S}$ is of one of the following sequence types:*

1. $\triangledown, \triangledown, \triangledown, \ldots, \triangledown, \triangledown, \triangledown$;
2. $\vartriangle, \triangledown, \triangledown, \ldots, \triangledown, \triangledown, \triangledown$;
3. $\vartriangle, \vartriangle, \vartriangle, \ldots, \vartriangle, \vartriangle, \vartriangle$;
4. $\triangledown, \vartriangle, \vartriangle, \ldots, \vartriangle, \vartriangle, \vartriangle$;

assuming $\mathbf{S}_1 = u_1, \ldots, u_m$ and each type has m data items.

Proof. This clearly holds if $\mathbf{S}_1$ contains at most two data items. Suppose now $\mathbf{S}_1$ contains more than two data items; there are four cases:

1. If $(u_1, u_2) = (\triangledown, \triangledown)$, then $\mathbf{B}$ is in `Queue Mode` for the first stage. Suppose $\vartriangle$ occurs after u_2, then let u_{i+1} be the first such occurrence; when the algorithm considers u_{i+1}, $\{u_{i-1}, u_i\} \subseteq \mathbf{B}$. Since $(u_{i-1}, u_i, u_{i+1}) = (\triangledown, \triangledown, \vartriangle)$, this contradicts Lemma 7.
2. If $(u_1, u_2) = (\triangledown, \vartriangle)$, then $\mathbf{B}$ is in `Queue Mode` for the first stage. Suppose $\triangledown$ occurs after u_2, then let u_{i+1} be the first such occurrence; when the algorithm considers u_{i+1}, $\{u_{i-1}, u_i\} \subseteq \mathbf{B}$. Since either $(u_{i-1}, u_i, u_{i+1}) = (\vartriangle, \vartriangle, \triangledown)$, or $(u_{i-1}, u_i, u_{i+1}) = (\triangledown, \vartriangle, \triangledown)$, this contradicts Lemma 7.
3. If $(u_1, u_2) = (\vartriangle, \triangledown)$, then we have two cases:
   (a) If $u_1 < 0.35$, then $\mathbf{B}$ is in `Queue Mode` for the first stage. Suppose $\vartriangle$ occurs after u_2, then let u_{i+1} be the first such occurrence; when the algorithm considers u_{i+1}, $\{u_{i-1}, u_i\} \subseteq \mathbf{B}$. Since either $(u_{i-1}, u_i, u_{i+1}) = (\triangledown, \triangledown, \vartriangle)$, or $(u_{i-1}, u_i, u_{i+1}) = (\vartriangle, \triangledown, \vartriangle)$, this contradicts Lemma 7.
   (b) If $u_1 \geq 0.35$ then $\mathbf{B}$ is in `Queue_1 Mode` for the first stage. Suppose $\vartriangle$ occurs after u_2, then let u_{i+1} be the first such occurrence; when the algorithm considers u_{i+1}, $\{u_1, u_i\} \subseteq \mathbf{B}$. Since $(u_1, u_i, u_{i+1}) = (\vartriangle, \triangledown, \vartriangle)$, this contradicts Lemma 7.
4. If $(u_1, u_2) = (\vartriangle, \vartriangle)$, then we have two cases:

(a) If $u_1 < 0.35$, then $\mathbf{B}$ is in `Queue Mode` for the first stage. Suppose $\triangledown$ occurs after u_2, then let u_{i+1} be the first such occurrence; when the algorithm considers u_{i+1}, $\{u_{i-1}, u_i\} \subseteq \mathbf{B}$. Since $(u_{i-1}, u_i, u_{i+1}) = (\triangle, \triangle, \triangledown)$, this contradicts Lemma 7.

(b) If $u_1 \geq 0.35$, then $\mathbf{B}$ is in `Queue_1 Mode` for the first stage. Suppose $\triangledown$ occurs after u_2, then let u_{i+1} be the first such occurrence; when the algorithm considers u_{i+1}, $\{u_1, u_i\} \subseteq \mathbf{B}$. Since $(u_1, u_i, u_{i+1}) = (\triangle, \triangle, \triangledown)$, this contradicts Lemma 7.

Therefore, $\mathbf{S}_1$ must be of one of the input sequence types above. □

Lemma 8 is useful as it eliminates many cases we need to consider. We now give two observations:

- **Observation I:** for the input sequence types 1 and 2, the data items in $\mathbf{S}_1$ is non-increasing, and the total weight of the first three data items of $\mathbf{S}_1$ is no greater than 1.
- **Observation II:** for the input sequence types 3 and 4, the data items in $\mathbf{S}_1$ is non-decreasing. and the total weight of the first three data items of $\mathbf{S}_1$ is greater than 0.8.

The data items in $\mathbf{S}_2$ is non-decreasing in both cases. For convenience, we denote $\textsc{alg}'$ the algorithm's solution at the end of the first stage; we also suppose $\mathbf{S}$ is non-empty; for otherwise, the problem becomes trivial.

Lemma 9. $|\textsc{opt}| > 1$.

Proof. Suppose now $|\textsc{opt}| = 1$, there are two cases:

1. If $\mathbf{S}$ contains a data item of type $\blacklozenge$, then u_n is the largest data item in $\mathbf{S}$ and $\textsc{opt} = \{u_n\}$. No matter which operational mode $\mathbf{B}$ is in, u_n is always accepted into the solution. Since u_n is the last data item in $\mathbf{S}$, the algorithm achieves the optimal solution in this case, which is a contradiction.
2. If $\mathbf{S}$ does not contain a data item of type $\blacklozenge$, then the entire input sequence $\mathbf{S}$ contain exactly one data item of type $\lozenge$, hence the algorithm easily achieves the optimal solution, which is a contradiction.

Therefore, both cases are impossible, and hence $|\textsc{opt}| > 1$. □

Lemma 10. *No feasible solution contains two data items in* $\mathbf{S}_2$*.*

Proof. We prove this by contradiction. Suppose there is a feasible solution which contains two data items in $\mathbf{S}_2$, then it is clear that $0.8 \leq u_{m+1} + u_{m+2} \leq 1$, and $u_{m+1} \leq 0.5$; we exam the beginning of the second stage when the algorithm considers u_{m+1}. Note that the algorithm has to reject u_{m+1}, for otherwise, when the algorithm considers the next data item u_{m+2}, the terminal condition is satisfied.

1. If $|\textsc{alg}'| < 2$, then $\|\textsc{alg}'\| \leq 0.4$. Since $\|\textsc{alg}'\| + u_{m+1} \leq 1$, the algorithm must then accept u_{m+1} into its solution, which is a contradiction.

2. If $|\textbf{ALG}'| = 2$, then $\mathbf{B}$ is in `Optimum Mode` for the second stage. Let $\textbf{ALG}' = \{u_i, u_j\}$, then we have $u_i + u_j < u_j + u_{m+1} < u_{m+1} + u_{m+2} \leq 1$, the algorithm must then accept u_{m+1} into its solution, which is a contradiction.
3. If $|\textbf{ALG}'| = 3$, then we have $\mathbf{S}_1$ is of either input sequence type 1 or 2; for otherwise, by Lemma 8, either $\textbf{ALG}' = \{\triangle, \triangle, \triangle\}$ or $\textbf{ALG}' = \{\triangledown, \triangle, \triangle\}$, and hence $\|\textbf{ALG}'\| > 0.8$, which is a contradiction.
   (a) If $\textbf{ALG}' = \{\triangle, \triangledown, \triangledown\}$, then $\textbf{ALG}' = \{u_1, u_{m-1}, u_m\}$. Since $u_1 \geq 0.3$, we have

$$0.4 \leq u_{m-1} + u_m \leq 0.5. \tag{10}$$

   If there exist a data item in $[0.4, 0.5]$, then combined with (10), that data item will interrupt the ordering and the terminal condition is satisfied. Hence, there exists no data item in $[0.4, 0.5]$, Therefore, No feasible solution contains two data items in $\mathbf{S}_2$.
   (b) If $\textbf{ALG}' = \{\triangledown, \triangledown, \triangledown\}$, then during the second stage, $\mathbf{B}$ is in `Stack Mode`, hence the algorithm must then accept u_{m+1} into its solution, which is a contradiction.

Therefore, no feasible solution contains two data items in $\mathbf{S}_2$. This completes the proof. □

Lemma 11. $|\textbf{OPT}| < 4 \Rightarrow \|\textbf{OPT}\| < 0.8$.

Proof. We prove this by contradiction. Suppose that $|\textbf{OPT}| < 4$ and $\|\textbf{OPT}\| \geq 0.8$; by Lemma 9, $|\textbf{OPT}| > 1$. There are two cases:

1. If $|\textbf{OPT}| = 2$, then let $\textbf{OPT} = \{u_r, u_s\}$ with $u_r \leq u_s$. Note that if $(u_r, u_s) = (\blacklozenge, \blacklozenge)$, then it contradicts Lemma 10. Therefore $(u_r, u_s) = (\lozenge, \blacklozenge)$, then u_r always enters the solution. This is because during the first stage, $\mathbf{B}$ is either in `Queue_1 Mode` or `Queue Mode`, every data item in $\mathbf{S}_1$ gets a chance to enter the solution. Since $u_r + u_s \geq 0.8$, u_s will interrupt the ordering and become the next data item to be examined when u_r is in $\mathbf{B}$, hence the terminal condition is satisfied; this is a contradiction.
2. If $|\textbf{OPT}| = 3$, then let $\textbf{OPT} = \{u_r, u_s, u_t\}$ with $u_r \leq u_s \leq u_t$, there are two possibilities:
   (a) If $(u_r, u_s, u_t) = (\lozenge, \lozenge, \lozenge)$, then consider the first three elements in $\mathbf{S}$. If $\mathbf{S}_1$ is of either input sequence type 1 or 2, then by **Observation I**, we have

$$0.8 \leq u_r + u_s + u_t \leq u_1 + u_2 + u_3 \leq 1.$$

   Therefore, the terminal condition is satisfied when the algorithm accepts the first, second and third data item; this is a contradiction. If $\mathbf{S}_1$ is of either input sequence type 3 or 4, then by **Observation II**, we have

$$0.8 \leq u_1 + u_2 + u_3 \leq u_r + u_s + u_t \leq 1.$$

   Therefore, the terminal condition is satisfied when the algorithm accepts the first, second and third data item; this is a contradiction.

(b) If $(u_r, u_s, u_t) = (\Diamond, \Diamond, \blacklozenge)$, then consider the input sequence $\mathbf{S}$. If $\mathbf{S}_1$ is of either input sequence type 1 or 2, then at the end of the first stage, $\{u_{m-1}, u_m\} \subseteq \textbf{ALG}'$. Since u_{m-1} and u_m are the two smallest data items, we have

$$0.8 \leq u_{m-1} + u_m + u_t \leq u_r + u_s + u_t \leq 1.$$

Therefore, the terminal condition is satisfied at the end of the first stage; this is a contradiction. If $\mathbf{S}_1$ is of either input sequence type 3 or 4, then by **Observation II**, we have

$$0.8 \leq u_1 + u_2 + u_3 \leq u_r + u_s + u_t \leq 1.$$

Therefore, the terminal condition is satisfied when the algorithm accepts the first, second and third data item; this is a contradiction.

Therefore, $\|\textbf{OPT}\| < 0.8$. This completes the proof. □

Lemma 12. $\mathbf{B}_{\text{mode}} = \texttt{Optimum} \Rightarrow \|\textbf{ALG}'\| < 0.8\|\textbf{OPT}\|$.

Proof. Suppose that $\|\textbf{ALG}'\| \geq 0.8\|\textbf{OPT}\|$, then since $\mathbf{B}$ in `Optimum Mode` during the second stage, $\|\mathbf{B}\|$ is non-decreasing. Therefore $\|\textbf{ALG}'\| \leq \|\textbf{ALG}\|$, the approximation ratio is

$$\rho = \frac{\|\textbf{ALG}\|}{\|\textbf{OPT}\|} \geq \frac{\|\textbf{ALG}'\|}{\|\textbf{OPT}\|} \geq 0.8,$$

which is a contradiction. □

Lemma 13. $|\textbf{ALG}'| \neq 0$.

Proof. Suppose that $|\textbf{ALG}'| = 0$, then $\mathbf{S}_1$ is empty. By Lemma 9, $|\textbf{OPT}| > 1$, hence **OPT** contains at least two data items in $\mathbf{S}_2$; this contradicts Lemma 10. Therefore, $|\textbf{ALG}'| \neq 0$. □

Lemma 14. $|\textbf{ALG}'| \neq 1$.

Proof. Suppose that $|\textbf{ALG}'| = 1$, then $\mathbf{S}_1$ contains only one data item. By Lemmas 9 and 10, $|\textbf{OPT}| > 1$, and **OPT** contains at most one data item in $\mathbf{S}_2$. Therefore $\textbf{OPT} = \{u_1, u_i\}$. Since $\mathbf{B}$ is in `Optimum Mode`, then by Lemma 10, the algorithm achieves the optimal solution, which is a contradiction. Therefore, $|\textbf{ALG}'| \neq 1$. □

Lemma 15. $|\textbf{ALG}'| \neq 2$.

Proof. Suppose that $|\textbf{ALG}'| = 2$, then $|\textbf{OPT}| < 4$. This is because if $|\textbf{OPT}| = 4$, $\mathbf{S}_1$ is of either input sequence type 1 or 2, and hence $|\textbf{ALG}'| = 3$, which is a contradiction. By Lemma 9, $|\textbf{OPT}| > 1$. There are two cases:

1. If $|\textbf{OPT}| = 2$, then let $\textbf{OPT} = \{u_r, u_s\}$ with $u_r \leq u_s$. By Lemma 10 and 11, u_r is in $\mathbf{S}_1$ and $\|\textbf{OPT}\| < 0.8$; there are two possibilities:

(a) If $(u_r, u_s) = (\Diamond, \Diamond)$, then if $m = 2$, then u_r and u_s are the only two data items in $\mathbf{S}_1$ and $\textsc{alg}' = \{u_r, u_s\}$. Since $\mathbf{B}$ is in `Optimum Mode` during the second stage, the algorithm clearly achieves the optimal solution, which is a contradiction. If $m > 2$, then $\mathbf{S}_1$ is of either input sequence type 3 or 4; for otherwise, $|\textsc{alg}'| > 2$. Therefore $\textsc{alg}' = \{u_i, u_m\}$, where $u_i > 0.3$; hence we have

$$\frac{\|\textsc{alg}'\|}{\|\textsc{opt}\|} = \frac{u_i + u_m}{u_r + u_s} > \frac{0.3 + u_m}{2u_m} \geq \frac{0.7}{0.8} > 0.8.$$

Since $\mathbf{B}$ is in `Optimum Mode` during the second stage, by Lemma 12, we have $\|\textsc{alg}'\| < 0.8\|\textsc{opt}\|$, which is contradiction.

(b) If $(u_r, u_s) = (\Diamond, \blacklozenge)$, then u_r must be the largest data item in $\mathbf{S}_1$. This is because $\|\textsc{opt}\| = u_r + u_s < 0.8$. If u_r is not the largest data item in $\mathbf{S}_1$, we can always replace it with the largest data item in $\mathbf{S}_1$; the solution can only increase, and the feasibility is still maintained. If $m = 2$, then $u_r \in \textsc{alg}'$. If $m > 2$, then $\mathbf{S}$ belongs to either input sequence 3 or 4; for otherwise, $|\textsc{alg}'| > 2$. Therefore, we also have $u_r \in \textsc{alg}'$. Since $\mathbf{B}$ is in `Optimum Mode` during the second stage, by Lemma 10, the algorithm achieves the optimal solution, which is also a contradiction.

2. If $|\textsc{opt}| = 3$, then let $\textsc{opt} = \{u_r, u_s, u_t\}$ with $u_r \leq u_s \leq u_t$, there are two possibilities:

(a) If $\textsc{alg}' = \{\nabla, \nabla\}$ or $\textsc{alg}' = \{\nabla, \triangle\}$, then $\mathbf{S}_1$ is of input sequence type 1, 2 or 4, and $\mathbf{S}_1$ contains only two data items. We then have

$$0.8 \leq u_1 + u_2 + u_3 \leq u_r + u_s + u_t \leq 1.$$

Therefore, the terminal condition is satisfied, which is a contradiction.

(b) If $\textsc{alg}' = \{\triangle, \triangle\}$, then $\mathbf{S}_1$ is of either input sequence type 3 or 4, hence $|\textsc{opt}| < 3$, which is a contradiction.

Therefore, $|\textsc{alg}'| \neq 2$. □

Lemma 16. $|\textsc{alg}'| \neq 3$.

Proof. Suppose that $|\textsc{alg}'| = 3$, then $\mathbf{S}_1$ is of either input sequence type 1 or 2; for otherwise, by Lemma 8, either $\textsc{alg}' = \{\triangle, \triangle, \triangle\}$ or $\textsc{alg}' = \{\nabla, \triangle, \triangle\}$, and hence $\|\textsc{alg}'\| > 0.8$, the terminal condition is satisfied, which is a contradiction.

1. If $\textsc{alg}' = \{\triangle, \nabla, \nabla\}$, then $\textsc{alg}' = \{u_1, u_{m-1}, u_m\}$; there are two cases:

(a) If $|\textsc{opt}| < 4$, then by Lemma 11, we have

$$\|\textsc{opt}\| < 0.8.$$

Since we have $\mathbf{B}$ in `Optimum Mode` during the second stage and

$$\|\textsc{alg}'\| = u_1 + u_{m-1} + u_m > 0.7,$$

this contradicts Lemma 12.

(b) If $|\textbf{OPT}| = 4$, then if $\mathbf{B}$ in `Queue Mode` during the first stage, then $|\textbf{ALG}'| = 4$, which is a contradiction. Therefore, $\mathbf{B}$ in `Queue_1 Mode` during the first stage. If $u_1 \in \textbf{OPT}$, then $|\textbf{ALG}'| = 4$, which is a contradiction. Therefore, $u_1 \notin \textbf{OPT}$; by **Observation I**, $\|\textbf{OPT}\| \leq u_2 + u_3 + u_4 + u_5$. Note that at the same time, we have the following three inequalities:

$$u_1 \geq 0.35, \tag{11}$$

$$u_1 + u_2 + u_3 < 0.8, \tag{12}$$

$$u_2 + u_3 \geq u_4 + u_5. \tag{13}$$

Combining (11) and (12), we have $u_2 + u_3 < 0.45$. Therefore, by (13),

$$\|\textbf{OPT}\| \leq u_2 + u_3 + u_4 + u_5 \leq 2(u_2 + u_3) < 0.9.$$

Since we have $\mathbf{B}$ in `Optimum Mode` during the second stage and

$$\|\textbf{ALG}'\| = u_1 + u_{m-1} + u_m > 0.75,$$

this contradicts Lemma 12.

2. If $\textbf{ALG}' = \{\triangledown, \triangledown, \triangledown\}$, then $\textbf{ALG}' = \{u_{m-2}, u_{m-1}, u_m\}$; there are two cases:

(a) If $\mathbf{S}_2$ is empty, then there two possibilities. If $m = 3$, then the algorithm clearly achieves the optimal solution. If $m > 3$, then $\|\textbf{ALG}\| > 0.65$. Since u_{m-2}, u_{m-1} and u_m are the three smallest data items in $\mathbf{S}$, we have $|\textbf{OPT}| < 4$. By Lemma 11, $\|\textbf{OPT}\| < 0.8$. Therefore, the approximation ratio is

$$\rho = \frac{\|\textbf{ALG}\|}{\|\textbf{OPT}\|} > \frac{0.65}{0.8} \geq 0.8,$$

which is a contradiction.

(b) If $\mathbf{S}_2$ is non-empty, then since u_{m-2}, u_{m-1} and u_m are the three smallest data items in $\mathbf{S}$, we have $|\textbf{OPT}| < 4$. By Lemma 11, $\|\textbf{OPT}\| < 0.8$. Since $\mathbf{B}$ in `Stack Mode` during the second stage and $u_m \leq u_{m-1} \leq u_{m-2} \leq 0.3$, we then have $\|\textbf{ALG}\| > 0.7$. Therefore, the approximation ratio is

$$\rho = \frac{\|\textbf{ALG}\|}{\|\textbf{OPT}\|} > \frac{0.7}{0.8} \geq 0.8,$$

which is a contradiction.

Therefore, $|\textbf{ALG}'| \neq 3$. □

Theorem 6 is immediate by Lemmas 13, 14, 15, and 16.