博士論文（要約）

Algorithmic Studies on Online Knapsack and Related Problems

(オンラインナップサックと関連する諸問題に対するアルゴリズム論的研究)

河瀬康志

Preface

In classical computational problems such as optimization problems and search problems, we are given entire input at one time, and we then compute a solution for the problem. However, in many practical applications such as routing in communications network, job allocation, and stock trading, we need to choose an action in each step based the current information without knowing the full information which will be completely obtained in the future. Such a problem is called an *online problem* and an algorithm for the problem is called an *online algorithm*. In contrast, a problem with full information on the input is called an *offline problem* and an algorithm for the problem is called an *offline algorithm*. Online algorithms are a natural topic of interest in many fields such as information science, operations research, economics, and learning theory.

Since an online algorithm is forced to make decisions without knowing the entire input, they may later turn out not to be optimal. The quality of an online algorithm is measured by the *competitive ratio*, which is the worst ratio between its performance and an optimal offline algorithm's performance.

The main topic of this thesis is *online knapsack problem*, i.e., online version of the knapsack problem. The *knapsack problem* is one of the most fundamental problems in the field of combinatorial optimization and has a lot of applications in the real world. The knapsack problem is that: given a set of items with values and sizes, we are asked to maximize the total value of selected items in the knapsack satisfying the capacity constraint.

In the online setting of the knapsack problem, the information of the input (i.e., the items) is given gradually, i.e., after a decision is made on the current item, the next item is given. The decisions we have made are irrevocable, i.e., once a decision has been made, it cannot be changed.

In particular, we focus on removable version, i.e., when an item is put into the knapsack, some items in the knapsack are removed if the sum of the sizes of the item and the total size in the current knapsack exceeds the capacity of the knapsack. It may need some cost to remove some items. Removable online problem with removal cost is studied under the name of *buyback problem*. The problem is motivated by the following real scenario in selling advertisements online. A seller allocates a limited inventory to a sequence of potential buyers. The buyers arrive sequentially, submit bids at their arrival time, and the seller must immediately decide to sell or not for their bid. The seller can cancel earlier allocation decision with some cost. Examples of cancellation costs are compensatory payment, paperwork cost, and shipping charge. Compensatory payment is usually constant

rate of the value of canceled bids. On the other hand, paperwork cost and shipping charge usually do not depend on bids values but the number of cancellations.

In this thesis, we provide algorithms for these online problems and conduct competitive analysis. One of the main results of this thesis is on the buyback problem under an unweighted knapsack constraint, where the knapsack problem is called unweighted if the value of each item is proportional to its size. We provide an optimal competitive algorithm for the problem.

Moreover, we introduce optimal composition ordering problem. The input is a set of real-valued functions f_i and a real number c. In maximum total order setting, our goal is to find a composition ordering which maximizes the value of composite function of c. We present a polynomial time algorithm for the problem when all the functions are monotone increasing and linear. We also prove that the problem is NP-hard even if the functions are monotone increasing, convex, and at most 2-piece piecewise linear.

Acknowledgment

I would like to express my sincere gratitude to Professor Kazuhisa Makino of Kyoto University, for his enthusiastic guidance and encouragements. He has taught me how to research, how to write papers. I would also like to thank my supervisor Professor Kazuo Murota of University of Tokyo for his warm support and valuable advice on my research.

I am also thankful to my collaborators. Many parts of this thesis (Chapters 4, 5, 6, and 7) have been completed with Professor Xin Han. He had spent a lot of time having meetings and discussions with me, and I thank him for his thoughtful hospitality when I visited his university, Dalian University of Technology in China. I would like to thank Kento Seimi for his collaboration on work presented in Chapter 8.

I am also very grateful to all the present and past members of Mathematical Informatics 2nd Laboratory for their friendship and encouragements. In particular, I thank Kei Kimura and Hanna Sumita. I thank Hiroshi Hirai, Yusuke Kobayashi, and Naonori Kakimura for their helpful comments in seminars and supports in my life at the laboratory.

I also appreciate the financial support by the Global COE "The Research and Training Center for New Development in Mathematics" from 2011 to 2013, and JST, ERATO, Kawarabayashi Large Graph Project from 2013 to 2014.

Contents

Chapter 1

Introduction

In classical computational problems such as optimization problems and search problems, we are given entire input at one time and we then compute a solution for the problem. However, in many practical applications such as routing in communications network, job allocation, and stock trading, we need to choose an action in each step based the current information without knowing the full information which will be completely obtained in the future. Such a problem is called an *online problem* and an algorithm for the problem is called an *online algorithm*. In contrast, a problem with full information on the input is called an *offline problem* and an algorithm for the problem is called an *offline algorithm*. Online algorithms are a natural topic of interest in many fields such as information science, operations research, economics, and learning theory. We describe a few examples of the online problems.

Ski rental problem (e.g. [16,51,52,78]): Suppose that we will go skiing several times. We can either buy skis and then use them forever, or rent them. Renting skis costs \$1 per day and buying skis costs \$$B$. We must decide whether to rent or buy skis each time without knowing how many times we will go skiing.

If we know in advance how many times we will go skiing, we can choose the optimal strategy. If we will go skiing for more than B times, the best strategy is to buy skis at the first time. On the other hand, if we will go skiing for less than B times, the best strategy is to rent skis every time.

This problem is a fundamental one and has a lot of applications. For example, consider a stream of packets arrive at a destination and are required by the TCP protocol to be acknowledged upon arrival. We can use a single acknowledgment packet to simultaneously acknowledge multiple outstanding packets. Thus we can reduce the overhead of the acknowledgments by waiting over time. On the other hand, delaying acknowledgments too much can interfere with the TCP's congestion control mechanisms. Therefore we should not allow the latency of acknowledgments to increase too much. This problem can be seen as a generalization of the ski rental problem.

Online Scheduling (e.g. [4, 34]): Suppose that we have N machines. We have a sequence of jobs, which arrive one by one and we must allocate each job to a machine immediately without knowing the future jobs. The goal is, for example, minimizing the maximum load on any machine or minimizing total completion time.

Paging problem (e.g. [13, 68, 79]): Suppose that we have a two-level memory system consisting of a small fast memory and a large slow memory. We have a sequence of requests, each of which specifies a page in the memory system. Requests arrive one by one, and the request is served if the corresponding page is in the fast memory. If the page is not in the fast memory, a page fault occurs. Then a page must be removed from the fast memory and the corresponding page must be loaded from the slow memory to the fast memory. The goal is minimizing the number of page faults incurred on the request sequence.

This is an important problem to implement computer operating system. For paging, the random access memory is the small fast memory and the hard-disk drive is the large slow memory. For caching, the CPU cache is the small fast memory and the random access memory is the large slow memory.

k-server problem (e.g. [9, 27, 61, 65]): Suppose that we have k mobile servers, which are located in a metric space. We have a sequence of requests, each of which specifies a point in the space. Requests arrive one by one and we must immediately determine which server to move to the requested point each time, without knowing the future requests. The goal is minimizing the total moving distance of the servers.

The paging problem is the k-server problem when the metric is uniform (all distances are 1) where the servers represent the small fast memory. Another example is that consider a customer support sending technicians to customers when they have trouble with their equipment. If the cost is the total wait time, this problem is the k-server problem when the distance of the metric is the required time where the servers represent the technicians.

Since an online algorithm is forced to make decisions without knowing the entire inputs, they may later turn out not to be optimal. The quality of an online algorithm is usually measured by the *competitive ratio*, which is the worst ratio between the cost of the solution obtained by the online algorithm and the optimal cost.

We can find the roots of online problems in classical combinatorial optimization and in the analysis of data structures. For example, Graham in 1966 [34] analyzed a greedy online algorithm for the problem of scheduling jobs on identical processors. He analyzed how the ordering of jobs effects on the performance of the greedy algorithm. The other example can be found in the amortized analysis on data structures, such as self-balancing binary trees [59, 80, 83], Fibonacci heap [29, 60], and disjoint-set data structure [28, 82, 84].

The concept of the competitive ratio was introduced by Sleator and Tarjan in 1985 [79] as a kind of approximation ratio, and the term "competitive" is introduced by Karlin, Manasse, Rudolph, and Sleator in 1988 [53]. Evaluating a performance by using a ratio is one of the standard way to analyze algorithms or mechanisms in the field of computer science. For instance, the *approximation ratio* for approximation problems, the *optimal robustness factor* for robust optimization problems, and the *price of anarchy* and the *price of stability* in the algorithmic game theory. The approximation ratio measures price of limited computational resources (see [85, 89]). The optimal robustness factor measures price of uncertain inputs and environments e.g., real-time computing environments with uncertain run-time availability (see [42, 50, 69]). The price of anarchy and the price of

stability measure how the efficiency of a system degrades due to selfish behavior of its agents (see [1, 62, 74]).

In this thesis, we consider the online version of knapsack problem and buyback problems described below. For the other online problems, refer to a survey paper and books [14, 46, 72].

In the next section we describe the problems addressed in this thesis. We first present the online knapsack problems. Next we provide the buyback problems. Lastly, we introduce optimal composition ordering problems. In Section 1.2, we outline the contributions of this thesis.

1.1 Problems Addressed in This Thesis

1.1.1 Online Knapsack Problems

The *knapsack problem* is one of the most fundamental problems in the field of combinatorial optimization and has a lot of applications in the real world (see [58]). The (classical) knapsack problem is that: given a set of items e_i $(i = 1, 2, \ldots, n)$ with values $v(e_i)$ and sizes $s(e_i)$, we are asked to maximize the total value of selected items in the knapsack that satisfies the capacity constraint. The ratio $v(e_i)/s(e_i)$ is called the *efficiency* of item e_i. Throughout this thesis, we assume that the capacity of knapsack is 1. Therefore, the knapsack problem can be represented as the following integer linear programming problem:

$$\begin{aligned} \text{maximize} \quad & \textstyle\sum_{i=1}^{n} v(e_i)\, x_i \\ \text{s.t.} \quad & \textstyle\sum_{i=1}^{n} s(e_i)\, x_i \leq 1, \\ & x_i \in \{0, 1\} \qquad (\forall i \in \{1, 2, \ldots, n\}). \end{aligned}$$

It is well-known that the knapsack problem is NP-hard [54] but admits a fully polynomial time approximation scheme (FPTAS) [45]. There are several pseudo-polynomial time algorithms using dynamic programming [10, 77]. Ito, Kiyoshima, and Yoshida [47] presented a constant-time randomized approximation algorithm by using weighted sampling. For other results of the knapsack problem such as approximation algorithms and heuristic algorithms, refer to papers and books such as [44, 57, 58, 60, 67, 85].

In the online setting of knapsack problem, i) the information of the input (i.e., the items) is given gradually, i.e., after a decision is made on the current item, the next item is given; ii) the decisions we have made are irrevocable, i.e., once a decision has been made, it cannot be changed. Given the ith item e_i, which has a value $v(e_i)$ and a size $s(e_i)$, we either accept e_i (i.e., put e_i into the knapsack) or reject it. In the removable setting, when e_i is put into the knapsack, we can remove some items in the knapsack with no cost to make room for e_i. Our goal is to maximize the profit, i.e., the sum of the values of items in the last knapsack.

An online knapsack problem was first studied on average case analysis by Marchetti-Spaccamela and Vercellis [66]. They proposed a linear-time algorithm with $O(\log^{3/2} n)$ expected competitive difference, under the condition that the capacity of the knapsack grows proportionally to the number of items n. Lueker [64] improved the expected competitive difference to $O(\log n)$ under a fairly general condition on the distribution.

On the worst case analysis, Marchetti-Spaccamela and Vercellis [66] showed that the online knapsack problem has no constant competitive ratio. Buchbinder and Naor [16] presented an $O(\log(U/L))$-competitive algorithm based on a general online primal-dual framework when the efficiency $v(e_i)/s(e_i)$ of each item e_i is in a known range $[L, U]$, and each size $s(e_i)$ is assumed to be much smaller than the capacity of the knapsack. They also showed an $\Omega(\log(U/L))$ lower bound for the competitive ratio for the case. Zhou, Chakrabarty, and Lukose [91] showed $\Omega(\log(U/L))$ is also a lower bound for the randomized case, which implies that the online knapsack problem has no constant randomized competitive ratio.

Iwama and Taketomi [48] studied the *removable* online knapsack problem. They obtained a $(1+\sqrt{5})/2 \approx 1.618$-competitive algorithm for the *unweighted online knapsack* when the removable condition is allowed, where the knapsack problem is called *unweighted* if the value of each item is proportional to its size, i.e., all items have the same efficiency. They also showed that this is the best possible by providing a lower bound $(1+\sqrt{5})/2$ for the case. We remark that the problem has unbounded competitive ratio, if at least one of the removal and unweighted conditions is not satisfied [48, 49]. For the randomized and general weighted case, Babaioff, Hartline, and Kleinberg [6] provided a lower bound 5/4.

There are several previous works on the removable online problems. Han and Makino [41] considered the online knapsack with limited cuts, i.e., the removable online knapsack problem with the condition that item are allowed to be cut at most k (≥ 1) times. Han and Makino [40] studied the removable online minimization knapsack problem and they provide the optimal competitive ratio. Han, Iwama, and Zhang [36] considered removable version of online square packing.

Removable online problem with removal cost is studied under the name of *buyback problem*. We describe the removable online knapsack problem with removal cost in the next subsection.

1.1.2 Buyback Problems

The buyback problem was first defined and studied by Babaioff, Hartline, and Kleinberg [5] and Constantin, Feldman, Muthukrishnan, and Pál [23]. The problem is motivated by the following real scenario in selling advertisements online. A seller allocates a limited inventory to a sequence of potential buyers. The buyers arrive sequentially, submit bids at their arrival time, and the seller must immediately decide to sell or not for their bid. The seller can cancel earlier allocation decision with some cost. Examples of cancellation costs are compensatory payment, paperwork cost, and shipping charge. Compensatory payment is usually proportional to the value of canceled items. On the other hand, paperwork cost and shipping charge usually do not depend on the value of items but on the number of

items.

More formally, the input for the buyback problem is a sequence of elements $e_1, e_2, \ldots, e_n$, each of which has a weight $w(e_i)$. Let $(E = \{e_1, \ldots, e_n\}, \mathcal{I})$ be an independence system, i.e., $\mathcal{I}$ is a family of subsets of E, and if $J \subseteq I \in \mathcal{I}$ then $J \in \mathcal{I}$. Then we want to find an independent set with maximum total weight. Given the ith element e_i, we either accept e_i or reject it with no cost where the set of accepted elements must be independent. When we accept an element e_i, we can cancel some of the previously accepted elements with some cost. Let B_i be the set of selected elements at the end of the ith round. Then $B_i \subseteq B_{i-1} \cup \{e_i\}$ and $B_i \in \mathcal{I}$. An algorithm must run based only on the weights $w(e_i)$ $(1 \leq i \leq k)$ and the feasibility of subsets $T \subseteq \{e_1, \ldots, e_k\}$. Our goal is to maximize the profit, i.e., the sum of the weights of elements accepted (and not canceled) minus the total cancellation cost occurred.

In this thesis we consider two types of cancellation costs: *proportional cost* and *unit cost*. Let $B = B_n$ be the final set held by an algorithm and $R = (\bigcup_i B_i) \setminus B$ be the set of elements canceled. In the proportional cost model, the utility of the algorithm is defined as $\sum_{e \in B} w(e) - f \cdot \sum_{e \in R} w(e)$ where $f > 0$ is a fixed given constant called the *buyback factor*. In the unit cost model, the utility of the algorithm is defined as $\sum_{e \in B} w(e) - c \cdot |R|$ where $c > 0$ is a fixed cost for each element.

The buyback problem with proportional cost was studied in [2, 3, 5, 6, 12, 23]. Babaioff *et al.* [5] and Constantin *et al.* [23] showed that the problem is $1 + 2f + 2\sqrt{f(1+f)}$ competitive for the single element constraint. Babaioff *et al.* [5] also showed that the problem has a competitive ratio $1+2f+2\sqrt{f(1+f)}$ for a matroid constraint. Ashwinkumar [2] extended their results and showed that the buyback problem with the constraint of k matroid intersection is $k(1+f)(1+\sqrt{1-\frac{1}{k(1+f)}})^2$ competitive. Babaioff *et al.* [5, 6] also studied the buyback problem with the weighted knapsack constraint. They showed that if the largest element is of size at most γ, where $0 < \gamma < 1/2$, then the competitive ratio is $1 + 2f + 2\sqrt{f(1+f)}$ with respect to the optimum solution for the knapsack problem with capacity $(1-2\gamma)$. They also proposed a randomized $3(1+2f+2\sqrt{f(1+f)})$-competitive algorithm for this problem. Ashwinkumar and Kleinberg [3] showed that the buyback problem for a matroid constraint is randomized $-W(\frac{-1}{e(1+f)})$ competitive against an oblivious adversary. Here W denote Lambert's W function, defined as the inverse of the function $f(x) = ze^z$. Since Lambert's W function is multivalued, we restrict to the case where $W(\frac{-1}{e(1+f)}) \leq -1$.

1.1.3 Optimal Composition Ordering Problems

We introduce *optimal composition ordering problems*. The input is n real functions $f_1, \ldots, f_n : \mathbb{R} \to \mathbb{R}$ and a constant $c \in \mathbb{R}$. We consider two settings: total composition and partial composition setting. The maximum total composition ordering problem is to compute a permutation $\sigma : [n] \to [n]$ which maximizes $f_{\sigma(n)} \circ f_{\sigma(n-1)} \circ \cdots \circ f_{\sigma(1)}(c)$. where $[n] = \{1, \ldots, n\}$, and the maximum partial composition ordering problem is to compute a permutation $\sigma : [n] \to [n]$ and a nonnegative integer k $(0 \leq k \leq n)$ which

maximize $f_{\sigma(k)} \circ f_{\sigma(k-1)} \circ \cdots \circ f_{\sigma(1)}(c)$. We similarly consider the minimization problems.

For example, if the input is $((f_1(x) = 2x - 6,\ f_2(x) = \frac{1}{2}x + 2,\ f_3(x) = x + 2),\ c = 2)$, the optimal value for the maximum total composition ordering problem is $f_1 \circ f_3 \circ f_2(c) = f_1(f_3(f_2(c))) = f_1(f_3(c/2 + 2)) = f_1(c/2 + 4) = c + 2 = 4$.

Considering composition ordering is a natural and fundamental problem. In fact, the composition ordering problems include single machine time-dependent scheduling problems and a kind of secretary problem as follows.

Time-dependent scheduling

Some machine scheduling problems with time-dependent processing times, *time-dependent scheduling* [22,32] can be represented as the optimal composition ordering problems. Given the start time $t_0 = 0$, and a set of jobs J_i $(i = 1, \ldots, n)$ with a ready time r_i, a deadline d_i, and processing time $p_i : \mathbb{R} \to \mathbb{R}$, consider the single machine scheduling problem to minimize the makespan. while trying to minimize the makespan. Here the makespan denotes the time when all the jobs have finished processing. We assume that the machine can handle one job at a time and preemption is not allowed.

Different from the classical setting, the processing time p_i is *not* constant, which depends on the *starting* time of job J_i. The models have studied to consider learning and deteriorating effects. Here each p_i is assumed to satisfy $p_i(t) \leq s + p_i(t + s)$ for any $t \geq t_0$ and $s \geq 0$, since we can earlier finish processing the job J_i if it is earlier started processing.

For simplicity, let we first consider the case in which each job has neither the ready time r_i nor the deadline d_i. Define $f_i(t) := t + p_i(t)$ for $i \in [n]$, and consider the minimum total composition ordering problem. Note that job J_i has been finished processing at time $f_i(t)$ if it is started processing at time t. This implies that $f_{\sigma(n)} \circ f_{\sigma(n-1)} \circ \cdots \circ f_{\sigma(1)}(c)$ denotes the makespan of the scheduling problem when we fix the ordering σ.

More generally, even if job J_i has both the ready time r_i, and the deadline d_i $(d_i \geq r_i)$, the problem can be reduced to the minimum total composition ordering problem defined as $c = t_0$ and

$$f_i(t) = \begin{cases} r_i + p_i(r_i) & (t \leq r_i), \\ t + p_i(t) & (r_i < t,\ t + p_i(t) \leq d_i), \\ \infty & (d_i < t + p_i(t)). \end{cases}$$

There exist many models of time-dependent scheduling problem as a restriction of functions $p_i(t)$ such as linear deterioration and linear shortening models.

In the linear deterioration model, the job processing times are restricted to be increasing linear functions that satisfy $p_i(t) = a_i t + b_i$ with two positive constants $a_i, b_i > 0$. a_i and b_i are respectively called the *deterioration rate* and the *basic processing time* of job J_i. Gawiejnowicz and Pankowska [33], Gupta and Gupta [35], Tanaev *et al.* [81], and Wajs [87] obtained the result that time-dependent scheduling problem of this model is solvable in $O(n \log n)$ time by scheduling jobs in the nonincreasing ordering of ratios b_i/a_i. Monsheiov [71] considered the proportional deterioration model, i.e., $b_i = 0$ $(\forall i \in [n])$, and

he showed the makespan is constant, i.e., does not depend on processing ordering. Cheng and Ding [19] provided an $O(n^5)$-time algorithm for the model $p_i(t) = at + b_i$ $(a, b_i > 0)$ with deadline d_i.

Another model is called the linear shortening model introduced by Ho *et al.* [43]. In this model, the job processing times are restricted to be nonincreasing linear functions that satisfy $p_i(t) = -a_i t + b_i$ with two constants $1 > a_i > 0,\ b_i > 0$ and $a_j\ (\sum_{i=1}^{n} b_i - b_j) < b_j$. They showed that the time-dependent scheduling problem of this model is also solvable in $O(n \log n)$ time by scheduling jobs in the nonincreasing ordering of ratios b_i/a_i.

Hardness results for time-dependent scheduling are as follows. Gawiejnowicz [31] showed that the problem of the proportional deterioration model with the ready time and the deadline is strongly NP-hard. Cheng and Ding [19] presented that the linear deterioration model with deadline is strongly NP-hard. Cheng and Ding [18] showed relationships between the linear deterioration model with the deadlines and the linear shortening model with the ready times, and the linear deterioration model with the ready times and the linear shortening model with deadlines. They also showed both the linear deterioration model with ready times and the linear shortening model with deadlines are strongly NP-hard.

The current status on the time complexity of the single-machine time-dependent scheduling problem are summarized in Table 1.1.

Table. 1.1. The current status on time complexity of single-machine time-dependent scheduling problem.

Model	Complexity	References
$p_j = b_j t$ †	$O(n)$	[71]
$p_j = a_j + b_j t$ *†	$O(n \log n)$	[33, 35, 81, 87]
$p_j = a_j - b_j t$ *‡	$O(n \log n)$	[43]
$p_j = a + b_j t, b_j \in \{B_1, B_2\}, d_j$ *†	$O(n \log n)$	[20]
$p_j = a_j + b_j \max\{t - t_0, 0\}$ *†	$O(n \log n)$	[17]
$p_j = a_j + f(t)$ **	$O(n \log n)$	[70]
$p_j = a_j + bt, d_j$ *†	$O(n^5)$	[19]
$p_j = a_j - bt, r_j$ *‡	$O(n^6 \log n)$	[18]
$p_j = a_j + bt, r_j$ *†	NP-hard	[18]
$p_j = 1 + b_j t, d_j$ *†	NP-hard	[20]
$p_j = 1 - b_j t, d_j$ *‡	NP-hard	[20]
$p_j = \max\{a_j - b_j t,\ a_j - b_j T\}$ *‡	NP-hard	[21]
$p_j = b_j t, r_j \in \{R_1, R_2\}, d_j \in \{D_1, D_2\}$ †	NP-hard	[31]
$p_j = b_j t, r_j, d_j$ †	Strongly NP-hard	[31]
$p_j = a_j + b_j t, r_j$ *†	Strongly NP-hard	[18]

* $a_j > 0$, † $b_j > 0$, ‡ $1 > b_j > 0$, * $f(t) : \mathbb{R}_+ \to \mathbb{R}_+$, nondecreasing function,

Free-order Secretary problem

Another application of the optimal composition ordering problems is the *free-order secretary problem*, which is closely related to the full-information secretary problem [26], knapsack and matroid secretary problems [7, 8, 75] and stochastic knapsack problems [24, 25]. Imagine an administrator willing to hire the best secretary out of n applicants for the position. Each applicant i has a nonnegative independent random variable X_i as his value. Here $X_1, \ldots, X_n$ are not necessarily the same probability distribution, and assume that the administrator knows the probability distributions of the random variables in advance. The applicants are interviewed one-by-one. A decision on each particular applicant is to be made immediately after the interview. Once rejected, the applicant cannot be hired. After the interview of applicant i, the administrator can observe the value X_i. The objective is to find the optimal strategy, i.e., find the interview ordering and the stopping rule to maximize the expected value.

Let $f_i(x) = \mathbf{E}[\max\{X_i, x\}]$. Then, by backward induction, the optimal value for an order (permutation) $\sigma : [n] \to [n]$ is $f_{\sigma(n)} \circ \cdots \circ f_{\sigma(1)}(0)$.

Thus this problem is reduced to the maximum total and partial composition ordering problems of $((f_i)_{i\in[n]}, 0)$. Furthermore, if X_i is a k-valued random variable with possible values $\{a_i^1, \ldots, a_i^k\}$ $(a_i^1 \geq \cdots \geq a_i^k > 0)$, and with probability that the variable takes the value a_i^j is p_i^j $(j = 1, \ldots, k)$, then we have the following $(k+1)$-piece piecewise linear function:

$$
\begin{aligned}
f_i(x) &= \sum_{j=1}^{k} p_i^j \max\{a_i^j, x\} \\
&= \begin{cases}
x & \left(x \geq a_i^1\right), \\
\vdots & \\
\sum_{j=1}^{l} p_i^j a_i^j + \sum_{j=l+1}^{k} p_i^j x & \left(a_i^l \geq x \geq a_i^{l+1}\right), \\
\vdots & \\
\sum_{j=1}^{k} p_i^j a_i^j & \left(a_i^k \geq x\right)
\end{cases} \\
&= \max_{l=0}^{k} \left\{ \sum_{j=1}^{l} p_i^j a_i^j + \sum_{j=l+1}^{k} p_i^j x \right\}.
\end{aligned}
$$

1.2 Contribution of This Thesis

The main results in this thesis are summarized as follows.

Table. 1.2. The current status on competitive ratios for online knapsack problems under convex functions, where our results are written in bold letters.

$f(x)$	linear	convex	specific properties
upper bound	$\frac{1+\sqrt{5}}{2}$ [48]	$\mathbf{\frac{5}{3}}$ [Theorem 5.15]	$\mathbf{\frac{1+\sqrt{5}}{2}}$ [Theorem 5.16]
lower bound	$\frac{1+\sqrt{5}}{2}$ [48]	$\mathbf{\frac{1+\sqrt{5}}{2}}$ [Theorem 5.18]	$\mathbf{\frac{1+\sqrt{5}}{2}}$ [Theorem 5.18]

1.2.1 Online Knapsack Problem

We consider randomized algorithms for online knapsack problem and deterministic algorithm for online knapsack problem under convex functions.

Randomized Algorithms for Online Knapsack Problem

We study the worst case analysis of randomized algorithms for online knapsack problems against an oblivious adversary.

We first provide a randomized 2-competitive algorithm for the unweighted non-removable online knapsack problem, and show that it is the best possible.

For the unweighted removable case, we propose a randomized 10/7-competitive algorithm. Our algorithm divides all the items into three groups: *small*, *medium* and *large*. If a large item comes, our algorithm accepts it and cancels all the items in the knapsack. Otherwise the algorithm first handles medium items, then applies a greedy algorithm for the small items. For medium items, it randomly selects the one among two deterministic subroutines. We also show that there exists no randomized online algorithm with competitive ratio less than 5/4 for the unweighted removable case.

For the general removable case, we present a simple randomized 2-competitive algorithm, which is an extension of the famous 2-approximation greedy algorithm for the offline knapsack problem. As a lower bound, we show that there exists no randomized online algorithm with competitive ratio less than $1 + 1/e$ for the general weight removable online knapsack problem.

Online Knapsack Problem under Convex Functions

We consider an online knapsack problem under a convex size-value function, i.e., the larger item has a higher efficiency. We first give a greedy online algorithm with a competitive ratio 2. Then we propose an improved online algorithm with a competitive ratio 5/3. We also prove that when the convex function has a specific property, our improved online algorithm is $(1 + \sqrt{5})/2$-competitive, which is optimal. Finally, we prove that the lower bound of this problem is $(1 + \sqrt{5})/2$. We summarize the current status on competitive ratios for the online knapsack problem in Table 1.2, where our results are written in bold letters.

1.2.2 Buyback Problem

We consider proportional cost and unit cost models for the buyback problem.

Proportional Cost Buyback Problem

We study proportional cost buyback problem with the single element constraint, a matroid constraint, or the unweighted knapsack constraint. Let $f > 0$ be a buyback factor, i.e., cancellation cost of an element e_i is $f \cdot w(e_i)$.

For the single element and the matroid cases, we consider the problem with upper and lower bounds of weights, i.e., each element e_i has a weight such that $l \leq w(e_i) \leq u$. We construct an optimal online algorithm and prove that this is the best possible. The competitive ratio is $\nu(l, u, f)$ which is described in Chapter 6.

For the unweighted knapsack case, we deal with the problem with lower bounds of weights, i.e., each element e_i has a weight such that $l \leq w(e_i) \leq 1$. We also construct an optimal online algorithm for the case and prove that this is the best possible. The competitive ratio is $\zeta(l, f)$. See Chapter 6 for details. The main ideas of the algorithm are: i) it rejects elements (with no cost) many times, but in at most one round, it removes some elements from the knapsack. ii) some elements are removed from the knapsack, only when the total value in the resulting knapsack gets high enough to guarantee the optimal competitive ratio.

Unit Cost Buyback Problem

We study unit cost buyback problem with a matroid constraint, or the unweighted knapsack constraint. Let $c > 0$ be the cancellation cost of each element.

For the matroid case, we consider the problem with upper and lower bounds of weights, i.e., each element e_i has a weight such that $l \leq w(e_i) \leq u$. We construct an optimal online algorithm and prove that this is the best possible. The competitive ratio is $\lambda(l, u, c)$ which is described in Chapter 7.

For the unweighted knapsack case, we deal with the problem with lower bounds of weights, i.e., each element e_i has a weight such that $l \leq w(e_i) \leq 1$. The competitive ratio is $\mu(l, c)$. See Chapter 7 for details. The main ideas of the algorithm are the same as the ones for the proportional cost model.

We summarize current status on competitive ratios for removable online knapsack problems in Table 1.3 and for buyback problems in Table 1.4, where our results are written in bold letters.

1.2.3 Optimal Composition Ordering Problems

We first show that the the maximum total composition ordering problem and the minimum total composition ordering problem are mutually reducible to one another, and the maximum partial composition ordering problem and the minimum partial composition ordering problem are also mutually reducible. Thus, we only consider the maximum total

Table. 1.3. The current status on competitive ratios for buyback problem, where our results are written in bold letters.

		unweighted		general	
		lower bound	upper bound	lower bound	upper bound
non-removable	det.	∞ [48]		∞ [66]	
	rand.	**2** [Thm. 4.1, 4.2]		∞ [91]	
no cost	det.	$\frac{1+\sqrt{5}}{2}$ [48]		∞ [49]	
	rand.	**5/4** [Thm. 4.8]	**10/7** [Thm. 4.6]	$\mathbf{1+\frac{1}{e}}$ [Thm. 4.13]	**2** [Thm. 4.10]
prop. cost	det.	$\boldsymbol{\zeta(l,f)}$ [Thm. 6.21, 6.29]		∞ [49]	
unit cost	det.	$\boldsymbol{\mu(l,c)}$ [Thm. 7.20, 7.28]		∞ [49]	

Table. 1.4. The current status on competitive ratios for buyback problems with upper and lower bounds of weights, i.e., each element e_i has weight $l \leq w(e_i) \leq u$, where our results are written in bold letters.

	single element, matroid	unweighted knapsack
prop. cost	$1+2f+2\sqrt{f(1+f)}$ [5, 7, 23] $(l=0, u=\infty)$ $\boldsymbol{\nu(l,u,f)}$ [Thm. 6.18, 6.19, and 6.20]	$\boldsymbol{\zeta(l,f)}$ [Thm. 6.21, 6.29]
unit cost	$\boldsymbol{\lambda(l,u,c)}$ [Thm. 7.7 and 7.9]	$\boldsymbol{\mu(l,c)}$ [Thm. 7.20, 7.28]

composition ordering problem and the maximum partial composition ordering problem. In addition, we show that the maximum partial composition ordering problem and the minimum partial composition ordering problem are respectively reducible to the maximum total composition ordering problem and the minimum total composition ordering problem.

We present a polynomial time algorithm for the maximum total composition ordering problem and the maximum partial composition ordering problem when the functions are monotone increasing and linear. Thus, we can solve time-dependent scheduling problem with both linear shortening and linear deterioration jobs in polynomial time.

We also propose a polynomial time algorithm for the maximum partial composition ordering problem when the functions are piecewise increasing, i.e., $f_i(x) = \max\{a_i x + b_i, c_i\}$ $(a_i > 0)$. This result implies a polynomial time algorithm for two-valued free-order secretary problem.

For negative results, we prove that the optimal composition ordering problems are NP-hard even if the functions are monotone increasing, convex (concave), and at most 2-piece piecewise linear.

We summarize the current status on the time complexity for the maximum total composition ordering problem in Table 1.5.

Table. 1.5. The current status on the time complexity of the maximum total composition ordering problem.

Functions	Complexity	Reference
$f_i(x) = a_i x \ (a_i > 1)$	$O(n)$	[71]
$f_i(x) = \begin{cases} ax - b_i & (x \geq -d_i) \\ -\infty & (x < -d_i) \end{cases} \quad (a > 1,\ b_i > 0)$	$O(n^5)$	[19]
$f_i(x) = a_i x - b_i \quad (a_i > 1,\ b_i > 0)$	$O(n \log n)$	[33, 35, 81, 87]
$f_i(x) = a_i x - b_i \quad (1 > a_i \geq 0,\ b_i > 0)$	$O(n \log n)$	[43]
$f_i(x) = \min\{ax - b_i, -r_i\} \quad (a > 1,\ b_i > 0)$	NP-hard	[18]
$f_i(x) = \begin{cases} a_i x - 1 & (x \geq -d_i) \\ -\infty & (x < -d_i) \end{cases} \quad (a_i > 1)$	NP-hard	[20]
$f_i(x) = \begin{cases} a_i x - 1 & (x \geq -d_i) \\ -\infty & (x < -d_i) \end{cases} \quad (1 > a_i > 0)$	NP-hard	[20]
$f_i(x) = \begin{cases} a_i t_i - b_i & (x > t_i) \\ a_i x - b_i & (t_i \geq x \geq s_i) \\ -\infty & (x < s_i) \end{cases}$	SNP-hard	[31]
$f_i(x) = \min\{a_i x + b_i, c_i\} \quad (a_i > 1)$	SNP-hard	[18]
$f_i(x) = a_i x + b_i \quad (a_i \geq 0)$	$O(n \log n)$	[Theorem 8.21]
$f_i(x) = \max\{x, a_i x + b_i\} \quad (a_i \geq 0)$	$O(n \log n)$	[Theorem 8.15]
$f_i(x) = \max\{x, a_i x + b_i, c_i\} \quad (a_i \geq 0)$	$O(n^2)$	[Theorem 8.19]
$f_i(x) = \max\{a_i^1 x + b_i^1, a_i^2 x + b_i^2\} \quad (a_i^1, a_i^2 > 0)$	NP-hard	[Theorem 8.31]
$f_i(x) = \max\{x, \min\{a_i^1 x + b_i^1, a_i^2 x + b_i^2\}\} \quad (a_i^1, a_i^2 > 0)$	NP-hard	[Theorem 8.29]

1.3 Organization of This Thesis

This thesis is organized as follows. Our results are presented in Chapters 4–8.

In Chapter 2, we give preliminaries which will be used in the rest of the thesis. In Section 2.2, we show a formal definition and properties of matroid. In Section 2.3, we formally define the online problems as request answer games [11]. In Chapter 3, we present previously known results for the online knapsack problem. Section 3.1 gives results in Marchetti-Spaccamela and Vercellis [66]. Section 3.2 describes the results in Iwama and Taketomi [48]. Section 3.3 provides results in Zhou, Chakrabarty, and Lukose [91]. Section 3.4 presents results in Iwama and Zhang [49]. In Chapter 4, we consider randomized algorithms for online knapsack problem. In Chapter 5, we study an online knapsack problem under a convex size-value function. In Chapters 6 and 7, we treat the buyback problem. Chapters 6 and 7 discuss the proportional and the unit cost cases. In Chapter 8, we consider the optimal composition ordering problems. Finally, we conclude this thesis by summarizing the obtained results and discussing open problems in Chapter 9.

Let us mention here the relation between our publications and the contents of this thesis. The results in Chapter 4 are given in [38], and those in Chapter 5 are presented

in [39]. The results in Chapter 6 and 7 are based on [37] and the results in Chapter 7 are due to [55]. The results in Chapter 8 are given in [56].

Chapter 2

Preliminaries

2.1 Notations

Throughout this thesis, we will use the following symbols and notations:

$\mathbb{Z}_+$	the set of all nonnegative integers.
$\mathbb{Z}_{++}$	the set of all positive integers.
$\mathbb{R}$	the set of all real numbers.
$\mathbb{R}_+$	the set of all nonnegative real numbers.
$[n]$	the set of the first n positive integers, i.e., $\{1, 2, \ldots, n\}$.
$f \circ g$	the composition of functions f and g, i.e., $f \circ g(x) := f(g(x))$ for any x.
$\bar{z}$	complex conjugate of the complex number z.
$\arg(z)$	argument of the complex number $z \neq 0$ $(-\pi < \arg(z) \leq \pi)$.
$\mathrm{Re}(z)$	real part of the complex number z.

2.2 Online Problem

In this section, we define an online problem as a *request-answer game*. Most of online problems can be naturally modeled as the request answer game, which is introduced by Ben-David, Borodin, Karp, Tardos, and Wigderson [11].

2.2.1 Request Answer Game

We view the online problem as a game between an online player and a malicious adversary. The adversary construct an input and the online player construct an output one after the other. The adversary try to construct the worst input for the online player based on the knowledge of the behavior of the online player.

Definition 2.1 (Request Answer Game)**.** A *request-answer game* $(R, \mathcal{A}, \mathcal{C})$ consists of a

request set R, a sequence of finite nonempty *answer set* $A_1, A_2, \ldots$, and a sequence of *cost functions* $C_1, C_2, \ldots$ where $C_n : R^n \times A_1 \times \cdots \times A_n \to \mathbb{R}_+ \cup \{\infty\}$.

Definition 2.2 (Deterministic Online Algorithm)**.** A *deterministic online algorithm* ALG for the request-answer game $(R, \mathcal{A}, \mathcal{C})$ is a sequence of functions $g_i : R^i \to A_i$ $(i = 1, 2, \ldots)$. Given an online algorithm $ALG = \{g_i\}$ and a request sequence $\sigma = (r_1, \ldots, r_n) \in R^n$, the output is an answer sequence

$$ALG[\sigma] = (a_1, \ldots, a_n) \in A_1 \times \cdots \times A_n$$

where $a_i = g_i(r_1, \ldots, r_i)$ for $i = 1, \ldots, n$. The *cost* incurred by ALG on σ, denoted by $ALG(\sigma)$ is defined as

$$ALG(\sigma) = C_n(\sigma, ALG[\sigma]).$$

The performance of an online algorithm is measured by its competitive ratio, the ratio between its value and the optimal value for the worst request sequence. The competitive ratio for online algorithms was introduced by Sleator and Tarjan in 1985 [79].

Definition 2.3 (Deterministic Competitive Ratio)**.** Given a request sequence $\sigma \in R^n$, the *optimal offline cost* on σ is defined as

$$OPT(\sigma) = \max\{C_n(\sigma, a) \,:\, a \in A_1 \times \cdots \times A_n\}.$$

An online algorithm ALG is deterministic ρ-competitive if

$$\sup_{\sigma \in A_1 \times \cdots \times A_n} \frac{OPT(\sigma)}{ALG(\sigma)} = \rho$$

where we define $0/0 = 1$. We denote the competitive ratio ρ as $\overline{\mathcal{R}}_{DET}(ALG)$.

The value of the competitive ratio is at least 1 and smaller is better.

Next, we define a randomized online algorithm and its competitive ratio.

Definition 2.4 (Randomized Online Algorithm)**.** A *randomized online algorithm* $RALG$ for the request-answer game $(R, \mathcal{A}, \mathcal{C})$ is a probability distribution over the set of all deterministic online algorithms ALG_x (we think of x as the random string that selects the deterministic algorithm). Given a randomized online algorithm $RALG$ and a request sequence σ, the output and the cost incurred by $RALG$ are random variables.

For randomized online algorithms, there are three different definitions of adversaries, i.e., *oblivious*, *adaptive-online*, and *adaptive-offline* adversaries.

Definition 2.5 (Adversaries)**.** An adversary is defined as a pair (Q, S), where Q is the requesting component, and S is the serving component.

For oblivious adversary, the requesting component Q is a sequence of requests $\sigma(Q) = (r_1, \ldots, r_{d_Q}) \in R^{d_Q}$.

In contrast, for adaptive adversary, the requesting component Q is a sequence of functions $q_i : A_1 \times \cdots \times A_{i-1} \to R \cup \{\text{STOP}\}$, $i = 1, 2, \ldots, d_Q$. In particular, d_Qth function q_{d_Q} only takes the value STOP. The index d_Q is a constant which means the maximum number of requests of the adversary. For an adversary (Q, S) and a deterministic algorithm ALG,

Let $\sigma(ALG, Q) = (r_1, \ldots, r_n)$ be the request sequence, $a(ALG, Q) = (a_1, \ldots, a_n)$ be the answer sequence, and $n = n(ALG, Q)$ be the length of the sequences for the adversary (Q, S) and the deterministic algorithm ALG. Then $q_i(a_1, \ldots, a_{i-1}) = r_i$ for $i = 1, \ldots, n$ and $q_n(a_1, \ldots, a_n) = \text{STOP}$.

For offline adversary, the serving component S is a sequence of answer $(a_1, \ldots, a_n) \in A_1 \times \cdots \times A_n$, where $n = n(ALG, Q)$.

In contrast, for online adversary, the serving component S is a sequence of functions $p_i : A_1 \times \cdots \times A_{i-1} \to A_i$, $i = 1, 2, \ldots, d_Q$. We denote the answer sequence of S for a deterministic algorithm ALG and an adversary (Q, S) by $b(ALG, (Q, S)) \in A_1 \times \cdots \times A_n$, where $n = n(ALG, Q)$.

Definition 2.6 (Randomized Competitive Ratio Against an Oblivious Adversary)**.** A randomized online algorithm $RALG$ is randomized ρ-competitive against oblivious adversary if

$$\sup_{(Q,S):\substack{\text{Oblivious}\\\text{Adversary}}} \frac{OPT(\sigma(Q))}{\mathbf{E}_x[ALG_x(\sigma(Q))]} = \rho$$

where we define $0/0 = 1$, and we abuse the notation $\mathbf{E}_x$ as the expectation with respect to the distribution over the set $\{ALG_x\}$, which defines $RALG$. We denote the competitive ratio ρ as $\overline{\mathcal{R}}_{OBL}(RALG)$.

Definition 2.7 (Randomized Competitive Ratio Against an Adaptive Online Adversary)**.** A randomized online algorithm $RALG$ is randomized ρ-competitive against adaptive online adversary if

$$\sup_{(Q,S):\substack{\text{Adaptive}\\\text{Online}\\\text{Adversary}}} \frac{\mathbf{E}_x[C_n(\sigma(ALG_x, Q), b(ALG_x, (Q, S)))]}{\mathbf{E}_x[ALG_x(\sigma(ALG_x, Q))]} = \rho$$

where we define $0/0 = 1$. We denote the competitive ratio ρ as $\overline{\mathcal{R}}_{AON}(RALG)$.

Definition 2.8 (Randomized Competitive Ratio Against an Adaptive Offline Adversary)**.** A randomized online algorithm $RALG$ is randomized ρ-competitive against adaptive offline adversary if

$$\sup_{(Q,S):\substack{\text{Adaptive}\\\text{Offline}\\\text{Adversary}}} \frac{\mathbf{E}_x[OPT(\sigma(ALG_x, Q))]}{\mathbf{E}_x[ALG_x(\sigma(ALG_x, Q))]} = \rho$$

where we define $0/0 = 1$. We denote the competitive ratio ρ as $\overline{\mathcal{R}}_{AOFF}(RALG)$.

Let us denote by $\overline{\mathcal{R}}_{DET}$ the supremum of the deterministic competitive ratio for any deterministic online algorithm, i.e, $\sup_{ALG} \overline{\mathcal{R}}_{DET}(ALG)$. Let $\overline{\mathcal{R}}_{OBL}$, $\overline{\mathcal{R}}_{AON}$, and $\overline{\mathcal{R}}_{AOFF}$ respectively denote the supremum of the competitive ratios against the oblivious adversary, the adaptive online adversary, and the adaptive offline adversary for any randomized online algorithm.

2.2.2 Relating the Adversaries

In this section, we consider relationships between the adversaries. By the definitions of the adversaries, we have the following Propositions.

Proposition 2.9. Given a request-answer game and a randomized online algorithm $RALG$, we have

$$\overline{\mathcal{R}}_{OBL}(RALG) \leq \overline{\mathcal{R}}_{AON}(RALG) \leq \overline{\mathcal{R}}_{AOFF}(RALG).$$

Proposition 2.10. Given a request-answer game, we have

$$\overline{\mathcal{R}}_{OBL} \leq \overline{\mathcal{R}}_{AON} \leq \overline{\mathcal{R}}_{AOFF} \leq \overline{\mathcal{R}}_{DET}.$$

Ben-David *et al.* [11] provided more relationships as follows.

Theorem 2.11 (Ben-David *et al.* [11])**.** If there is a randomized algorithm that is α-competitive against adaptive offline adversary, then there exists an α-competitive deterministic algorithm.

This results implies $\overline{\mathcal{R}}_{DET} = \overline{\mathcal{R}}_{AOFF}$.

Theorem 2.12 (Ben-David *et al.* [11])**.** Suppose ALG is an α-competitive randomized algorithm against adaptive online adversary, and there exists a β-competitive randomized algorithm against oblivious adversary, then ALG is at least $(\alpha\beta)$-competitive against adaptive offline adversary.

This results implies $\overline{\mathcal{R}}_{AOFF} \leq \overline{\mathcal{R}}_{OBL} \cdot \overline{\mathcal{R}}_{AON}$.

2.2.3 Yao's Principle

In this subsection, we study Yao's principle, which is a game-theoretic technique to proving lower bounds on the performance of randomized algorithms.

Let $S_k := \{x \in \mathbb{R}^k : \sum_{i=1}^{k} x_i = 1,\ x \geq 0\}$.

Theorem 2.13 (von Neumann's Minimax Theorem [86])**.** Let M be a real $m \times n$ matrix. Then we have

$$\max_{p \in S_m} \min_{q \in S_n} p^T M q = \min_{q \in S_n} \max_{p \in S_m} p^T M q.$$

Let e_k denote a unit vector with a 1 in the kth position and 0s elsewhere.

Theorem 2.14 (Loomis' Theorem [63])**.** Let M be a real $m \times n$ matrix, we have

$$\max_{p \in S_m} \min_{j \in [n]} p^T M e_j = \min_{q \in S_n} \max_{i \in [m]} e_i^T M q.$$

This theorem implies that for any $p \in S_m$

$$\min_{j \in [n]} p^T M e_j \leq \min_{q \in S_n} \max_{i \in [m]} e_i^T M q.$$

Applying this inequality to the competitive ratio against the oblivious adversary, we have the following theorem.

Theorem 2.15 (Yao's principle [90])**.** Let G be any finite request answer game. Let $RALG$ be any online randomized algorithm for G, and let σ be any probability distribution over request sequences σ_y. Then we have

$$\overline{\mathcal{R}}_{OBL}(RALG) \geq \min_x \frac{\mathbf{E}_y[OPT(\sigma_y)]}{\mathbf{E}_y[ALG_x(\sigma_y)]}$$

where $\mathbf{E}_y$ is the expectation with respect to the distribution over the set $\{\sigma_y\}$.

2.3 Matroids

In this section, we show some basic properties of matroids, which is introduced by Whitney in 1935 [88]. The concept of a matroid is a combinatorial abstraction of linear independence in matrices.

A *matroid* is a set system $(E, \mathcal{I})$, i.e. E is a finite set and $\mathcal{I}$ is a family of subsets of E, with the following properties:

(I1) $\emptyset \in \mathcal{I}$,
(I2) $J \subseteq I \in \mathcal{I} \Rightarrow J \in \mathcal{I}$,
(I3) $I, J \in \mathcal{I},\ |J| < |I| \Rightarrow \exists v \in I \setminus J$ such that $J \cup \{v\} \in \mathcal{I}$.

A set system only with the properties (I1) and (I2) is called *independence system*.

Given a matroid $M = (E, \mathcal{I})$, a subset I of E is called *independent set* if I belongs to $\mathcal{I}$, and an inclusionwise maximal independent set is called a *base*.

The maximum size of an independent subset of $T \subseteq E$ is called the *rank* of T, denoted by $r(T) := \max\{|I| : I \in \mathcal{I},\ I \subseteq T\}$. If $r(T) = r(T \cup \{e\})$ for $e \in E$ and $T \subseteq E$, we say that T *spans* e. The set $\mathrm{cl}(T) := \{e \in E : T \text{ spans } e\}$ is called the *closure* of T.

2.3.1 Examples of Matroids

Here are some examples of matroids:

- **Explicit Example**. Let $E = \{1, 2, 3, 4\}$ and $\mathcal{I} = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}\}$. Then the set system $(E, \mathcal{I})$ is a matroid.

- **Uniform Matroids**. Let E be a finite set and k be a nonnegative integer. Then the set system $U_n^k := (E,\ \{I\ :\ I \subseteq E,\ |I| \leq k\})$ is a matroid, that is called a *k-uniform matroid* where $n := |E|$.
- **Partition Matroids**. Let E_i be finite sets and k_i be nonnegative integers ($i = 1, 2, \ldots, m$). Then the set system $(\bigcup_i E_i, \{\bigcup_i I_i\ :\ I_i \subseteq E_i,\ |I_i| \leq k_i\})$ is a matroid, that is called a *partition matroid*.
- **Linear Matroids**. Let A be an $m \times n$ matrix. Let $E = \{1, 2, \ldots, n\}$ and $\mathcal{I}$ be the set of all subsets I of E such that the columns of A with index in I are linearly independent. Then $(E, \mathcal{I})$ is a matroid, that is called a *linear matroid*.
- **Graphic Matroids**. Let $G = (V, E)$ be an undirected graph, and $\mathcal{I}$ be the set of all subsets I of E such that I is a forest in the graph G. Then $(E, \mathcal{I})$ is a matroid, that is called a *graphic matroid*.
- **Transversal Matroid**. Let $G = (U, V, E)$ be a bipartite graph, and $\mathcal{I}$ be the set of all subsets I of U such that I is sets of endpoints of matchings of the graph. Then $(U, \mathcal{I})$ is a matroid that is called a *transversal matroid*.

2.3.2 Greedy Algorithms

One important property of matroids is that the greedy algorithm works for them.

Let $(E, \mathcal{I})$ be a matroid and each element $e \in E$ has a nonnegative weight $w(e)$. Then we can find the maximum weight independent set $I \in \mathcal{I}$ with Algorithm 1.

Algorithm 1 Matroid Greedy Algorithm

```
1: sort E = {e_1, e_2, ..., e_n} such that w(e_1) ≥ w(e_2) ≥ ··· ≥ w(e_n).
2: initialize I_0 := ∅.
3: for i = 1 to n do
4:    if I_{i-1} ∪ {e_i} ∈ I then I_i := I_{i-1} ∪ {e_i}.
5: end for
6: return I_n
```

Theorem 2.16 (Oxley [76] Lemma 1.8.3)**.** Algorithm 1 outputs a maximum weight independent set.

In this thesis, we use another greedy algorithm.

Theorem 2.17. Algorithm 2 outputs a maximum weight independent set.

Proof. For $T \subseteq E$ and $\theta \geq 0$, let

$$T(\theta) = \{t \in T\ :\ w(t) > \theta\} \quad \text{and} \quad \mathrm{cl}_\theta(T) = \mathrm{cl}(T(\theta)).$$

We first prove that $\mathrm{cl}_\theta(I_k) = \mathrm{cl}_\theta(\{e_1, \ldots, e_k\})$ for any $\theta \geq 0$ and $1 \leq k \leq n$.

Since $I_k \subseteq \{e_1, \ldots, e_k\}$, it holds that $\mathrm{cl}_\theta(I_k) \subseteq \mathrm{cl}_\theta(\{e_1, \ldots, e_k\})$.

Algorithm 2 Matroid Greedy Algorithm without Sorting

1: let $E = \{e_1, e_2, \ldots, e_n\}$
2: initialize $I_0 := \emptyset$.
3: **for** $i = 1$ to n **do**
4: **if** $I_{i-1} \cup \{e_i\} \in \mathcal{I}$ **then** $I_i := I_{i-1} \cup \{e_i\}$.
5: **else** let e_j be the smallest element such that $I_{i-1} \cup \{e_i\} \setminus \{e_j\} \in \mathcal{I}$
6: **if** $w(e_i) > w(e_j)$ **then** $I_i := I_{i-1} \cup \{e_i\} \setminus \{e_j\}$
7: **end for**
8: return I_n

We prove $\mathrm{cl}_\theta(I_k) \supseteq \mathrm{cl}_\theta(\{e_1, \ldots, e_k\})$ by induction. $\mathrm{cl}_\theta(I_1) = \mathrm{cl}_\theta(\{e_1\})$ is obvious. Assume that $\mathrm{cl}_\theta(I_k) \supseteq \mathrm{cl}_\theta(\{e_1, \ldots, e_k\})$. Then it is sufficient to prove that $\mathrm{cl}_\theta(I_{k+1}) \supseteq \mathrm{cl}_\theta(\{e_1, \ldots, e_k, e_{k+1}\})$. If $w(e_{k+1}) \leq \theta$, then $\{e_1, \ldots, e_k\}(\theta) = \{e_1, \ldots, e_k, e_{k+1}\}(\theta)$ and $I_k(\theta) = I_{k+1}(\theta)$. Thus $\mathrm{cl}_\theta(I_{k+1}) \supseteq \mathrm{cl}_\theta(\{e_1, \ldots, e_k, e_{k+1}\})$ holds. We then consider the case $w(e_{k+1}) > \theta$. If $I_{k+1} = I_k$, then $e_{k+1} \in \mathrm{cl}_\theta(I_k)$ and $\mathrm{cl}_\theta(I_{k+1}) \supseteq \mathrm{cl}_\theta(\{e_1, \ldots, e_k, e_{k+1}\})$. On the other hand, if $I_{k+1} = I_k \cup \{e_{k+1}\} \setminus \{e'\}$, then $w(e') \leq \theta$ and $e' \notin \mathrm{cl}_\theta(\{e_1, \ldots, e_{k+1}\})$, or $w(e') > \theta$ and $e' \in \mathrm{cl}_\theta(I_{k+1})$. Therefore $\mathrm{cl}_\theta(I_{k+1}) \supseteq \mathrm{cl}_\theta(\{e_1, \ldots, e_k, e_{k+1}\})$.

Let the output $I_n = \{b_1, \ldots, b_k\}$ such that $w(b_1) \geq \cdots \geq w(b_k)$ and let a maximum weight independent set $OPT = \{b_1^*, \ldots, b_k^*\}$ such that $w(b_1^*) \geq \cdots \geq w(b_k^*)$.

We prove I_n is a maximum weight independent set by contradiction. Assume that I_n is not a maximum weight independent set. Let l be the smallest integer such that $w(b_l) < w(b_l^*)$. $I' = \{b_1, b_2, \ldots, b_{l-1}\}$ and $OPT' = \{b_1^*, b_2^*, \ldots, b_l^*\}$. There exists an element b_j^* $(1 \leq j \leq l)$ such that $I' \cup \{b_j^*\} \in \mathcal{I}$. This contradicts $\mathrm{cl}_{w(b_l)}(I_n) = \mathrm{cl}_{w(b_l)}(\{e_1, \ldots, e_n\})$ since $b_j^* \notin \mathrm{cl}(I') = \mathrm{cl}_{w(b_l)}(I_n)$. □

Chapter 3

Online Knapsack Problems

In this chapter we study deterministic algorithms for online knapsack problem. We consider four cases, depending on whether unweighted or general weight, and removable or non-removable.

3.1 Unweighted Non-removable Online Knapsack Problem

Marchetti-Spaccamela and Vercellis [66] showed that the competitive ratio of the unweighted non-removable online knapsack problem is infinite.

Theorem 3.1 (Marchetti-Spaccamela and Vercellis [66,91])**.** There exists no deterministic online algorithm with constant competitive ratio for the unweighted non-removable online knapsack problem.

Proof. Let (s, v) denote an item whose size and value are s and v, respectively. Let A denote an online algorithm chosen arbitrarily. We consider two sequences of input items:

$$(1, \varepsilon), \tag{3.1}$$

$$(1, \varepsilon),\ (1, 1) \tag{3.2}$$

where ε is a sufficiently small positive number.

If A rejects the first item, the competitive ratio becomes infinite for the input sequence (3.1). Otherwise, A accepts the first item, and the competitive ratio approaches infinite for the input sequence (3.2) as $\varepsilon \to 0$. □

3.2 Unweighted Removable Online Knapsack Problem

Iwama and Taketomi [48] studied the unweighted removable online knapsack problem. They obtained a $(1 + \sqrt{5})/2 \approx 1.618$-competitive algorithm for this problem, and showed that this is the best possible by providing a lower bound $(1 + \sqrt{5})/2$.

Let e_i be the item given in the ith round. Let B_i be the set of selected items by their Algorithm 3 at the end of the ith round. We denote by $s(B_i)$ the total size of items in B_i. Algorithm 3 partitions all the items into three groups, *small*, *medium* and *large* where an item e is called *small*, *medium*, and *large* if $s(e) \leq (3-\sqrt{5})/2$, $(3-\sqrt{5})/2 < s(e) < (\sqrt{5}-1)/2$, and $s(e) \geq (\sqrt{5}-1)/2$, respectively. Let S, M, and L respectively denote the sets of small, medium, and large items (see Figure 3.1).

Figure 3.1. Item partition for Algorithm 3.

Their algorithm is briefly described as follows. If a large item comes, the algorithm keeps it in the knapsack (by removing all the items we have chosen), since it ensures $(1+\sqrt{5})/2$-competitivity of the algorithm. Otherwise, the algorithm chooses the medium items from the smallest to the largest, and the small items from the largest to the smallest.

Algorithm 3 Iwama and Taketomi [48]

```
1: B_0 := ∅
2: for each item e_i in order of arrival do
3:    if s(B_{i-1}) ≥ (√5−1)/2 then
4:       B_i := B_{i-1}
5:    else
6:       choose the largest L-item from B_{i-1} ∪ {e_i}.
7:       choose the M-items among B_{i-1} ∪ {e_i} from the smallest to the largest.
8:       choose the S-items among B_{i-1} ∪ {e_i} from the largest to the smallest.
9:    end if
10: end for
```

Theorem 3.2 (Iwama and Taketomi [48])**.** Algorithm 3 is $(1+\sqrt{5})/2$-competitive for the unweighted removable online knapsack problem.

Proof. Let *OPT* be the (offline) optimal value for the problem. If the condition in the line 3 of Algorithm 3 is satisfied in some round, then the competitive ratio is at most

$$\frac{1}{\frac{\sqrt{5}-1}{2}} = \frac{1+\sqrt{5}}{2}$$

since the optimal value is at most 1. Thus we assume that the condition in the line 3 of Algorithm 3. is not satisfied before some large item arrives.

If a large item arrives, the algorithm keeps a large item, which implies $s(B_n) \geq (\sqrt{5}-1)/2$.

Assume that no large item arrives. We then consider the following three cases.

Case 1: A small item is removed by Algorithm 3. Let Algorithm 3 removes some small items in ith round, and a removed small item be e_j. Then we have $s(B_i) + s(e_j) > 1$ and

$$s(B_i) > 1 - s(e_j) \geq 1 - \frac{3-\sqrt{5}}{2} = \frac{\sqrt{5}-1}{2}.$$

Case 2: The sum of the sizes of two smallest medium items is at most 1. In this case, the algorithm keeps two medium items in some round i, which implies

$$s(B_i) > 2 \cdot \frac{3-\sqrt{5}}{2} = 3 - \sqrt{5} > \frac{\sqrt{5}-1}{2}.$$

Case 3: Otherwise, i.e., no small item is removed by Algorithm 3 and the sum of sizes of any two medium items is larger than 1. If no medium item arrives, it is easy to see $OPT = s(B_n)$ and the competitive ratio is 1. Otherwise, let m^* and m_* respectively be the largest and smallest medium items and let s be the sum of sizes of all small items in the input sequence. Then the competitive ratio is

$$\frac{OPT}{s(B_n)} \leq \frac{s(m^*)+s}{s(m_*)+s} \leq \max\left\{\frac{s(m^*)}{s(m_*)}, 1\right\} \leq \frac{1+\sqrt{5}}{2}.$$

□

Theorem 3.3 (Iwama and Taketomi [48])**.** There exists no deterministic online algorithm with competitive ratio less than $(1+\sqrt{5})/2$ for the unweighted removable online knapsack problem.

Proof. We consider the following two input sequences:

$$\frac{3-\sqrt{5}}{2},\ \frac{\sqrt{5}-1}{2}+\varepsilon, \tag{3.3}$$

$$\frac{3-\sqrt{5}}{2},\ \frac{\sqrt{5}-1}{2}+\varepsilon,\ \frac{\sqrt{5}-1}{2} \tag{3.4}$$

where we identify the items with their size (value) and ε is a sufficiently small positive number. We note that the first and the second items do not in the knapsack together.

Let A denote an online algorithm chosen arbitrarily. At the end of second round, if A keeps the first item, then the competitive ratio is

$$\frac{\frac{\sqrt{5}-1}{2}}{\frac{3-\sqrt{5}}{2}} = \frac{1+\sqrt{5}}{2}$$

for the input sequence (3.3). Otherwise, i.e., A keeps the second item at the end of the second round, and the competitive ratio is at least

$$\frac{1}{\frac{\sqrt{5}-1}{2}+\varepsilon} \to \frac{1+\sqrt{5}}{2}$$

as $\varepsilon \to 0$ for the input sequence (3.4). □

3.3 General Non-removable Online Knapsack Problem

The competitive ratio of the general non-removable online knapsack problem is also infinite by Theorem 3.1. On the other hand, Zhou, Chakrabarty, and Lukose [91] presented $\ln(Ue/L)$-competitive algorithm when the size of each item is very small and the efficiency of each item is bounded by two positive constants L and U.

Let e_i be the item given in the ith round. We denote by B_i the set of selected items by Algorithm 4 at the end of the ith round. Let $s(B_i)$ and $v(B_i)$ respectively be the total size and value of items in B_i. Let $\Psi(z) = (Ue/L)^z(L/e)$.

Algorithm 4 Zhou *et al.* [91]

1: $B_0 := \emptyset$
2: **for each** item e_i in order of arrival **do**
3: **if** $s(B_i) + s(e_i) \le 1$ and $v(e_i)/s(e_i) \ge \Psi(s(B_{i-1}))$ **then** $B_i := B_{i-1} \cup \{e_i\}$
4: **else** $B_i := B_{i-1}$
5: **end for**

Theorem 3.4 (Zhou *et al.* [91])**.** Algorithm 4 is $\ln(Ue/L)$-competitive for the general non-removable online knapsack problem when the size of each item is very small and the efficiency of each item is lower and upper bounded by two positive constants L and U.

Proof. Let OPT be an optimal (offline) solution, and let $S = \sum_{e\in(B_n\cap OPT)} s(e)$ and $V = \sum_{e\in(B_n\cap OPT)} v(e)$. Since B_n contains every item with value $\Psi(s(B_n))$, we have

$$v(OPT) \le V + \Psi(s(B_n))(1 - W).$$

As each item e_j picked by the algorithm have efficiency at least $\Psi(z_j)$ where $z_j = w(B_{j-1})$, we have

$$V \ge \sum_{e_j\in B_n\cap OPT} \Psi(z_j)w(e_j),$$
$$v(B_n \setminus OPT) \ge \sum_{e_j\in B_n\setminus OPT} \Psi(z_j)w(e_j).$$

Thus we have

$$\begin{aligned}\frac{v(OPT)}{v(B_n)} &\le \frac{V + \Psi(s(B_n))(1 - W)}{V + v(B_n \setminus OPT)} \\ &\le \frac{\sum_{e_j\in B_n\cap OPT} \Psi(z_j)w(e_j) + \Psi(s(B_n))(1 - W)}{\sum_{e_j\in B_n\cap OPT} \Psi(z_j)w(e_j) + v(B_n \setminus OPT)}\end{aligned}$$

$$\leq \frac{\Psi(s(B_n))W + \Psi(s(B_n))(1-W)}{\sum_{e_j \in B_n \cap OPT} \Psi(z_j)w(e_j) + \sum_{e_j \in B_n \setminus OPT} \Psi(z_j)w(e_j)}$$
$$\leq \frac{\Psi(s(B_n))}{\sum_{e_j \in B_n} \Psi(z_j)w(e_j)}.$$

Based on the assumption that the sizes are very small, we have

$$\begin{aligned}\sum_{e_j \in B_n} \Psi(z_j)w(e_j) &\approx \int_0^{s(B_n)} \max\{L,\ \Psi(z)\}dz \\ &= \int_0^{\frac{1}{\ln(Ue/L)}} L dz + \int_{\frac{1}{\ln(Ue/L)}}^{s(B_n)} \Psi(z)dz \\ &= \frac{L}{\ln(Ue/L)} + \frac{L}{e}\frac{(Ue/L)^{s(B_n)} - (Ue/L)^{\frac{1}{\ln(Ue/L)}}}{\ln(Ue/L)} \\ &= \frac{L}{e}\frac{(Ue/L)^{s(B_n)}}{\ln(Ue/L)} = \frac{\Psi(s(B_n))}{\ln(Ue/L)}.\end{aligned}$$

Therefore, the competitive ratio is

$$\frac{v(OPT)}{v(B_n)} \leq \ln(Ue/L).$$

□

Zhou *et al.* [91] also showed that the competitive ratio in Theorem 3.4 is tight.

Theorem 3.5 (Zhou *et al.* [91])**.** There exists no online algorithm with competitive ratio $\ln(Ue/L)$ for the general non-removable online knapsack problem when the size of each item is very small and the efficiency of each item is lower and upper bounded by two positive constants L and U.

Proof. Let η be a sufficiently small positive number and let n be a sufficiently large positive integer. Let k be a largest integer such that $(1+\eta)^k \leq U/L$, i.e., $k = \lfloor \frac{\ln(U/L)}{\ln(1+\eta)} \rfloor$. Let (s, v) denote an item whose size and value are s and v, respectively. For a nonnegative integer i $(0 \leq i \leq k)$, we consider the following sequences with $n(i+1)$ items:

$$\begin{aligned}&\underbrace{(1/n, (1+\eta)^0 L/n), \ldots, (1/n, (1+\eta)^0 L/n)}_{n \text{ items}}, \\ &\underbrace{(1/n, (1+\eta)^1 L/n), \ldots, (1/n, (1+\eta)^1 L/n)}_{n \text{ items}}, \\ &\quad\vdots \\ &\underbrace{(1/n, (1+\eta)^i L/n), \ldots, (1/n, (1+\eta)^i L/n)}_{n \text{ items}}.\end{aligned} \tag{3.5}$$

Let A denote an online algorithm chosen arbitrarily. We specify the algorithm by the vector $(f_0, f_1, \ldots, f_k)$, where f_i is the number of item with $(1/n, (1+\eta)^i L/n)$ that A picks.

By the knapsack constraint, we have $\sum_{i=0}^{k} f_i \leq n$.

Let OPT_i be the optimal value, and A_i be the value by A for the input sequence (3.5). Then the competitive ratio is

$$\begin{aligned}
\max_{0\leq i\leq n} \frac{OPT_i}{A_i} &= \max_{0\leq i\leq n} \frac{(1+\eta)^i L}{\sum_{j=0}^{i}(1+\eta)^j L f_j/n} \\
&= \frac{n}{\min_{0\leq i\leq n}\sum_{j=0}^{i}(1+\eta)^{j-i} f_j} \\
&\geq \frac{((k+1)\eta+1)n}{\sum_{i=0}^{k}((1+\eta)^{i-k} f_k + \eta\sum_{j=0}^{i}(1+\eta)^{j-i} f_j)} \\
&= \frac{((k+1)\eta+1)n}{\sum_{j=0}^{k}((1+\eta)^{j-k} f_k + \eta\sum_{i=j}^{k}(1+\eta)^{j-i} f_j)} \\
&= \frac{((k+1)\eta+1)n}{\sum_{j=0}^{k}((1+\eta)^{j-k} f_k + \eta\frac{1-(1+\eta)^{-k+i-1}}{1-(1+\eta)^{-1}} f_j)} \\
&= \frac{((k+1)\eta+1)n}{\sum_{j=0}^{k}(1+\eta) f_j} \\
&\geq \frac{(k+1)\eta+1}{1+\eta} \\
&\geq \frac{(\ln(U/L)/\ln(1+\eta))\eta+1}{1+\eta} \to \ln(U/L)+1 = \ln(Ue/L).
\end{aligned}$$

□

3.4 General Removable Online Knapsack Problem

Iwama and Zhang [49] showed that we cannot get constant competitive algorithm for the unweighted version of the removable online knapsack problem.

Theorem 3.6 (Iwama and Zhang [49])**.** There exists no deterministic online algorithm with constant competitive ratio for the general removable online knapsack problem.

Proof. Let (s, v) denote an item whose size and value are s and v, respectively. Let A denote an online algorithm chosen arbitrarily. For a positive integer n, our adversary requests the sequence of items

$$(1,1), \left(\frac{1}{n^2}, \frac{1}{n}\right), \ldots, \left(\frac{1}{n^2}, \frac{1}{n}\right)$$

until A rejects or removes the first item or rejects n^2 items.

We first note that algorithm A must take the first item, since otherwise the competitive ratio of A becomes infinity.

If A removes the first item, the competitive ratio is at least n. Otherwise, i.e., A rejects

all the items $(1/n^2, 1/n)$, the competitive ratio is at least

$$\frac{(1/n) \cdot n^2}{1} = n.$$

Therefore, the competitive ratio is greater than any integer. □

Chapter 4: Randomized Algorithms for Online Knapsack Problems は論文投稿中のため公表を延期します．5 年以内に出版予定です．

Chapter 5

Online Knapsack Problem under Convex Functions

In this chapter, we consider the online knapsack problem with a convex function. We first propose a simple greedy online algorithm, in which the larger item has a higher priority. We prove that the online algorithm is 2-competitive. Observe that the optimal value for the problem with a convex function can be estimated by the largest item in the input. Using this fact, we improve the greedy online algorithm by the following approach: i) divide all the items into three groups, *large*, *medium* and *small*, ii) for large and small items, we select them in the way: *the largest the first*, iii) for medium items, we select them in the way: *the smallest the first*. We prove that the improved algorithm is 5/3-competitive. Another result is that: if the convex function has a specific property, the improved online algorithm is $(1+\sqrt{5})/2$-competitive, which extends the result in Iwama and Taketomi [48]. For example, for any $1 \leq c \leq 1.3884$, the function $f(x) = x^c$ satisfies the property. Finally, we prove that the lower bound for the problem is $(1+\sqrt{5})/2$.

5.1 Knapsack Problem under Convex Function

Knapsack Problem with Convex Function: The input is a unit size of knapsack and a set of items associated with a size-value function $f(\cdot)$, where function $f(\cdot)$ is convex. The output is to select a subset of items to maximize the total value of all the selected items without exceeding the capacity of the knapsack.

Online Knapsack Problem under Convex Function: In this chapter, we study an online knapsack problem under a convex size-value function. The capacity of the knapsack and the function $f(\cdot)$ are known before packing. The word "Online" means that i) items are given one by one over time, i.e., after a decision is made on the current item, then the next one is known, ii) in order to accept a new item, it is allowed to remove old items in the knapsack. During selection, in order to accept a new item, some old items are allowed to be discarded or removed. The objective of the online knapsack is the same as the offline version, i.e., to maximize the value under the capacity constraint. Let $f(\cdot)$ be the convex function, i.e., for each item with size x, its value is $f(x)$. Here we require the

convex function to satisfy two conditions: i) $f(0) = 0$, ii) $f(x) > 0$ for any $0 < x \leq 1$.

Next, we outline several properties of a convex function [15]. If $f(\cdot)$ is convex, then for any $0 \leq \theta \leq 1$

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y), \tag{5.1}$$

by the definition of the convex, we have

$$f(x_1) + f(y_1) \geq f(x_2) + f(y_2), \tag{5.2}$$

where $x_1 + y_1 = x_2 + y_2$ and $x_1 \geq x_2 \geq y_2 \geq y_1$. By the above properties, it is not difficult to prove the following two Lemmas [15].

Lemma 5.1. Given a convex function $f(\cdot)$, if $f(0) = 0$, then $f(x)/x$ is nondecreasing for all $x > 0$.

Proof. For any $x_1 \geq x_2 > 0$, if we can prove that

$$\frac{f(x_1)}{x_1} \geq \frac{f(x_2)}{x_2},$$

then it is done. By equation (5.1), setting $\theta = x_2/x_1$, $x = x_1$ and $y = 0$, we have

$$\frac{x_2}{x_1} f(x_1) + \frac{x_1 - x_2}{x_1} f(0) \geq f(x_2) \quad \Rightarrow \quad \frac{x_2}{x_1} f(x_1) \geq f(x_2),$$

since $f(0) = 0$. Hence we have this lemma. □

Lemma 5.2. Given real numbers $x_1, x_2, \ldots, x_n$, $\sum_{i=1}^{n} f(x_i) \leq f\left(\sum_{i=1}^{n} x_i\right)$.

Proof. By Lemma 5.1, for any $1 \leq j \leq n$, we have $f(x_j) \leq x_j \cdot \frac{f(\sum_{i=1}^{n} x_i)}{\sum_{i=1}^{n} x_i}$. Hence

$$\sum_{j=1}^{n} f(x_j) \leq \sum_{j=1}^{n} x_j \cdot \frac{f\left(\sum_{i=1}^{n} x_i\right)}{\sum_{i=1}^{n} x_i} = f\left(\sum_{i=1}^{n} x_i\right).$$

□

Lemma 5.3. Given a convex function $f(\cdot)$ with $f(0) = 0$, if $f(x) > 0$ for any $x > 0$, then $f(\cdot)$ is a monotonically increasing function.

Proof. For any $x_1 > x_2$, if $x_2 = 0$, then $f(x_1) > 0 = f(x_2)$, else $x_2 > 0$, by Lemma 5.1, $f(x_1) \geq \frac{x_1}{x_2} \cdot f(x_2) > f(x_2)$. □

By Lemmas 5.2 and 5.3, we have the convexity of the value function induces that the largest item has the best ratio of the value to the size. Then we have the following lemma.

Lemma 5.4. Consider the knapsack problem under the value function $f(\cdot)$. If the largest size in the input is $\alpha \geq 0.5$, then the optimal value is at most $f(\alpha) + f(1-\alpha)$.

Proof. Let $x_1, x_2, \ldots, x_n$ be the sizes of items in an optimal solution such that $x_1 \geq x_2 \geq \cdots \geq x_n$, where n is the number of the items in the optimal solution. Then

$OPT = \sum_{i=1}^{n} f(x_i)$. The claim holds if $\sum_{i=1}^{n} x_i \le \alpha$ by Lemma 5.2. So, we assume there exists an integer k such that

$$\sum_{i=1}^{k} x_i \le \alpha \quad \text{and} \quad \sum_{i=1}^{k+1} x_i > \alpha.$$

Since $\sum_{i=1}^{n} x_i \le 1$, we have $\sum_{i=k+2}^{n} x_i < 1 - \alpha$. By Lemma 5.2, we have

$$\sum_{i=1}^{k} f(x_i) \le f\left(\sum_{i=1}^{k} x_i\right) \quad \text{and} \quad \sum_{i=k+2}^{n} f(x_i) \le f\left(\sum_{i=k+2}^{n} x_i\right).$$

Then

$$\begin{aligned} f\left(\sum_{i=1}^{k} x_i\right) + f(x_{k+1}) + f\left(\sum_{i=k+2}^{n} x_i\right) &\le f(\alpha) + f\left(\sum_{i=1}^{k+1} x_i - \alpha\right) + f\left(\sum_{i=k+2}^{n} x_i\right) \\ &\le f(\alpha) + f(1-\alpha), \end{aligned}$$

where the last inequality holds by Lemma 5.2. Hence we have $\sum_{i=1}^{n} f(x_i) \le f(\alpha) + f(1-\alpha)$. □

5.2 A Simple Online Algorithm

In this section, we first give a simple greedy online algorithm and prove that it is 2-competitive. By Lemma 5.1, when the convex function passes through the origin, we have the efficiency (the value divided by the size) is a nondecreasing function with respect to the size. Then the idea of the greedy algorithm is that the larger item has a higher priority, i.e., when a new large item arrives, if necessary, remove the smallest item first until the new item can be accommodated.

Online algorithm ALG_1: when a new item m is given, our online algorithm works as below:

(a) Sort all the items in the knapsack including item m in nonincreasing order of sizes.
(b) Remove the smallest one until the total size of items in the knapsack is at most 1.

Let $ALG_1(t)$ and $OPT(t)$ be the values by online algorithm ALG_1 and an optimal algorithm after time step $t \ge 1$ respectively. Let $\beta(t)$ be the value of the largest item in all the discarded items at or before time t.

Lemma 5.5. For any time step $t \ge 1$, we have $OPT(t) \le ALG_1(t) + \beta(t)$.

Proof. Let x_i be the ith largest item in the input after time t, where $1 \le i \le t$. If $\sum_{i=1}^{t} s(x_i) \le 1$, then our online algorithm does not need to discard any item, i.e., $OPT(t) = ALG_1(t)$. Otherwise, there exists an integer $k < t$ such that $\sum_{i=1}^{k} s(x_i) \le 1 < \sum_{i=1}^{k+1} s(x_i)$. By Lemma 5.1, the larger item has a higher efficiency. Then the optimal

value of the fractional Knapsack problem is

$$\sum_{i=1}^{k} f(s(x_i)) + \frac{1-\sum_{i=1}^{k} s(x_i)}{s(x_{k+1})} f(s(x_{k+1})) \leq \sum_{i=1}^{k+1} f(s(x_i)).$$

Since the optimal value is bounded from above by the fractional optimal value, we have

$$OPT(t) \leq \sum_{i=1}^{k+1} f(s(x_i)).$$

In our online algorithm, the discarding policy used is the smaller the first, thus all the largest k items are kept in the knapsack, i.e., $ALG_1(t) = \sum_{i=1}^{k} f(s(x_i))$. Since $\beta(t)$ is the $(k+1)$th largest item, we have $OPT(t) \leq \sum_{i=1}^{k+1} f(s(x_i)) = ALG_1(t) + \beta(t)$. □

It is not difficult to see that $ALG_1(t) \geq \beta(t)$, then by Lemma 5.5, we have the following theorem and lemma.

Theorem 5.6. The online algorithm ALG_1 is 2-competitive.

Lemma 5.7. If the largest item has size at most $1/k$, where k is a positive integer, algorithm ALG_1 is $\frac{k+1}{k}$-competitive.

5.3 Improved Upper Bounds

In this section, we first observe that the optimal value can be estimated by the maximal size of items in the input and the size-value function $f(\cdot)$. Then combining with some techniques, we improve the greedy algorithm and prove that its competitive ratio is 5/3. We then prove that the improved algorithm is optimal with respect to the competitive ratio if the size-value convex function satisfies a certain condition.

Definition 5.8. θ-point: given two variables $0.5 < x \leq 1$ and $\theta > 1$, a convex function $f(\cdot)$, if

$$\frac{f(x)}{f(1-x)} = \theta,$$

then x is θ-point of function $f(\cdot)$.

Lemma 5.9. Given $\theta > 1$ and a convex function $f(\cdot)$ with domain $[0,1]$, if $f(0) = 0$ and $f(x) > 0$ for any $x > 0$, then θ-point x_0 of $f(\cdot)$ exists in $\left(0.5, \frac{\theta}{1+\theta}\right]$ and is unique.

Proof. Define a new function $F(x)$ for $x \in [0,1]$ as below, where $\theta > 1$:

$$F(x) = f(x) - \theta f(1-x).$$

A convex function must be continuous, so $f(x)$ and $F(x)$ are continuous. And we have

$$F\left(\frac{\theta}{1+\theta}\right) = f\left(\frac{\theta}{1+\theta}\right) - \theta f\left(\frac{1}{1+\theta}\right) \geq 0,$$

where the last inequality holds by Lemma 5.1. And $F(0.5) = f(0.5) - \theta f(0.5) < 0$ since $f(x) > 0$ for any $x > 0$ and $\theta > 1$. We know function $F(x)$ is continuous, then there exists x in $\left(0.5, \frac{\theta}{1+\theta}\right]$ such that $F(x) = 0$, i.e., $x_0 \in \left(0.5, \frac{\theta}{1+\theta}\right]$.

By Lemma 5.3, functions $f(x)$ and $-f(1-x)$ are monotonically increasing. Then $F(x)$ is also monotonically increasing. Hence there is a unique solution for $F(x) = 0$ in interval $\left(0.5, \frac{\theta}{1+\theta}\right]$. □

5.3.1 An Improved Online Algorithm

Observe that if all the items are very large then it will be good enough to have the largest one in the knapsack; if all the items are very small, then it will be good enough to call the greedy algorithm. To have a good competitive ratio, the point is how to handle the medium item. Our strategies are: i) divide items into three groups, *large*, *medium* and *small*, ii) for large and small items, the larger item has a higher priority, iii) for medium items, the smaller item has a higher priority. Combining with some techniques to estimate the optimal value, we propose a refined online algorithm in this subsection.

Let $\theta = 1.5$. Define a θ-point x_0 as below:

$$\frac{f(x_0)}{f(1-x_0)} = 1.5.$$

Grouping: we divide the interval $[0, 1]$ into three sub-intervals,

$$I_0 = [x_0, 1], \quad I_1 = [1 - x_0, x_0), \quad I_2 = [0, 1 - x_0).$$

By Lemma 5.9, x_0 exists and is unique. Given an item with size s, if $s \in I_i$, then the item belongs to type-i, where $0 \le i \le 2$.

Online algorithm ALG_2: when a new item m is given, our algorithm works as below:

1. $s(m) \in I_0$: accept the largest item in the knapsack including the current item, discard all the others.
2. $s(m) \in I_1$:
   (*a*) If the total value in the knapsack is at least $f(x_0)$, then discard m.
   (*b*) Else if there is an item q with $s(q) \in I_1$ in the knapsack,
      (*i*) If $s(q) + s(m) \le 1$ then accept items q and m, discard all the others.
      (*ii*) Else accept the *smaller* one of the two items, discard the larger one.
   (*c*) Else accept item m, if necessary, discard items in a way: the smallest the first.
3. $s(m) \in I_2$:
   (*a*) If the total value in the knapsack is at least $f(x_0)$ then discard item m.
   (*b*) Else call the greedy algorithm ALG_1 to handle item m, i.e., if necessary use the policy of the smallest the first to discard items.

Pattern: define a pattern of packing in the knapsack as a vector $v = \{v_0, v_1, v_2\}$ of three

components, where v_i is the number of type-i items in the knapsack, where $0 \leq i \leq 2$. It is not difficult to see the following results by referring to Steps $1., 2.(a), 3.(a)$ of our online algorithm.

Observation 5.10. If some type-0 items have been given, the largest one must be accepted.

Observation 5.11. If the packing pattern is one of $(1, 0, 0)$, $(0, 2, *)$, where $*$ denotes any feasible integer, then the total value in the knapsack is at least $f(x_0)$.

Observation 5.12. Once the total value by our online algorithm is at least $f(x_0)$ at the current time step, then it never goes down below $f(x_0)$ in the future.

Observation 5.13. If the execution passes through Step $2.(b)(ii)$ at time t, we have the following results:

- the largest item has size less than x_0;
- the minimal type-1 item is selected in the knapsack;
- in the input, there is only one type-1 item or any two type-1 items cannot be packed together;
- the largest type-2 item must be packed if it exists.

Before proving the main result: our algorithm is 5/3-competitive, we need the following lemma first.

Lemma 5.14. Assume in the input any two type-1 items cannot be packed together in the knapsack. Let γ_i be the ith largest type-2 item, where $i \geq 1$. Let β_1 be the largest item, which is type-1 item. If $s(\beta_1) + \sum_{i=1}^{k} s(\gamma_i) \geq 1$, where k is the number of type-2 item in the input, then the optimal value is at most $f(s(\beta_1)) + f(\sum_{i=1}^{k} s(\gamma_i))$.

Proof. Define a set $O = \{\beta_1, \gamma_1, \ldots, \gamma_k\}$. Let O^* be the set of items in an optimal solution. By the assumption, there is at most one type-1 item in O^*. Observe that i) the larger item has a larger efficiency by Lemma 5.1; ii) $s(\beta_1) + \sum_{i=1}^{k} s(\gamma_i) \geq 1$ and γ_i is the ith largest type-2 item in the input. We claim the average efficiency in O is not less than the one in O^*.

Hence we have $f(O^*) \leq f(O) = f(s(\beta_1)) + f\left(\sum_{i=1}^{k} s(\gamma_i)\right)$. □

Analysis: let $ALG(t)$ and $OPT(t)$ be the values by our online algorithm and an optimal algorithm after time step $t \geq 1$ respectively. Next we prove that for any integer $t \geq 1$, $\frac{OPT(t)}{ALG(t)} \leq 5/3$, i.e., the online algorithm is 5/3-competitive. We use induction to prove the result, namely, at some time $t_0 \geq 1$, for any input, if we have $OPT(t_0) \leq 5/3 \cdot ALG(t_0)$, we need to prove the claim still holds for the next time step.

Theorem 5.15. The online algorithm ALG_2 is 5/3-competitive.

Proof. It is not difficult to see that, After the first time step, we have $OPT(1) = ALG(1)$, we have $\frac{OPT(1)}{ALG(1)} \leq 5/3$. Assume $\frac{OPT(t_0)}{ALG(t_0)} \leq 5/3$ holds for any time step $t_0 \geq 1$. Next we prove that $\frac{OPT(t_1)}{ALG(t_1)} \leq 5/3$, where t_1 is the next time step.

Let m be the item given at time t_1 and $s(m)$ be its size. Consider the execution route when item m is processed. There are two cases: i) no items are discarded after time step t_1; ii) some items are discarded after time step t_1. For each case, we prove that $\frac{OPT(t_1)}{ALG(t_1)} \leq 5/3$ holds.

Case 1: It is not difficult to see that $ALG(t_1) = ALG(t_0) + f(s(m))$ and $OPT(t_1) \leq OPT(t_0) + f(s(m))$. By the assumption $\frac{OPT(t_0)}{ALG(t_0)} \leq 5/3$, we have $\frac{OPT(t_1)}{ALG(t_1)} \leq 5/3$.

Case 2: Let α be the size of the largest item in the input. There are two subcases.

Case 2.1: $\alpha \geq x_0$. By Observation 5.10 the item α must have been selected in the knapsack. Then $ALG(t_1) = f(\alpha)$. By Lemma 5.4, we have

$$\frac{OPT(t_1)}{ALG(t_1)} \leq \frac{f(\alpha) + f(1-\alpha)}{f(\alpha)} \leq \frac{5}{3},$$

where the last inequality holds from $\frac{f(1-\alpha)}{f(\alpha)} \leq \frac{f(1-x_0)}{f(x_0)} = \frac{2}{3}$.

Case 2.2: $\alpha < x_0$. According to the step that item m passes through, we have the following four subcases. For $i \geq 1$, let γ_i be the ith largest type-2 item in the input. Assume that items $\gamma_1, \gamma_2, \ldots, \gamma_{k-1}, \gamma_k$ are selected in the knapsack just after time t_1, where $k \geq 0$ is the maximal index.

Case 2.2.1: the execution for item m passes through one of Steps 2.(a), 2.$(b)(i)$ or 3.(a). by Observations 5.11 and 5.12, $ALG(t_1) \geq f(x_0)$. By Lemma 5.4, $OPT(t_1) \leq f(x_0) + f(1-x_0)$. Hence $\frac{OPT(t_1)}{ALG(t_1)} \leq \frac{5}{3}$.

Case 2.2.2: the execution for item m passes through Step 2.$(b)(ii)$. There is only one type-1 item in the knapsack. We rename it as m. Let β_1 be the largest item of type-1. Remember that index k is the maximal index such that items $\gamma_1, \gamma_2, \ldots, \gamma_{k-1}, \gamma_k$ are selected in the knapsack just after time t_1. There are three cases on k.

Case 2.2.2.1: $k = 0$. By Observation 5.13 iv), there is no type-2 item given so far, otherwise item γ_1 has been selected in the knapsack. We have $ALG(t_1) = f(s(m))$ and $OPT(t_1) = f(s(\beta_1))$. Since both items m and β_1 are type-1, we have $OPT(t_1) \leq 1.5ALG(t_1)$.

Case 2.2.2.2: $k = 1$. By Observation 5.13 iv), item γ_1 must be accepted in the knapsack. If there is only one type-2 item in the input, then by Observation 5.13 iii), we have

$$ALG(t_1) = f(s(m)) + f(s(\gamma_1)), \text{ and } OPT(t_1) \leq f(s(\beta_1)) + f(s(\gamma_1)).$$

Hence we have $\frac{OPT(t_1)}{ALG(t_1)} < \frac{f(s(\beta_1))}{f(s(m))} \leq 1.5$. Else $s(\gamma_2) > 0$, due to the fact that there is no space in the knapsack for item γ_2, we have $x_0 + s(\gamma_1) + s(\gamma_2) > 1$. Due to $s(\gamma_1) \geq s(\gamma_2)$,

$$2s(\gamma_1) > 1 - x_0. \tag{5.3}$$

If $f(s(\gamma_1)) \geq \frac{f(1-x_0)}{2}$ then we have

$$\begin{aligned} ALG(t_1) &\geq f(s(m)) + f(s(\gamma_1)) \geq \frac{3}{2} \cdot f(1-x_0) = f(x_0) \qquad \text{(by definition of } x_0\text{)} \\ &= \frac{3}{5} \cdot (f(x_0) + f(1-x_0)) > \frac{OPT(t_1)}{5/3}, \end{aligned}$$

where the last inequality holds from that $OPT(t_1) \leq f(x_0) + f(1-x_0)$ by Lemma 5.4.

Else we have $f(s(\gamma_1)) < \frac{f(1-x_0)}{2}$. By Observation 5.13 iii), there is at most one type-1 item in the optimal solution. By Lemma 5.14, $OPT(t_1) \leq f(x_0) + 2f(s(\gamma_1))$. Then

$$\frac{OPT(t_1)}{ALG(t_1)} \leq \frac{f(x_0) + 2f(s(\gamma_1))}{f(1-x_0) + f(s(\gamma_1))} \leq \frac{1.5 + 2 \cdot 0.5}{1 + 0.5} = \frac{5}{3}.$$

Case 2.2.2.3: $k \geq 2$. Then $ALG(t_1) = f(s(m)) + \sum_{i=1}^{k} f(s(\gamma_i))$, by Lemma 5.14 we have

$$OPT(t_1) \leq f(s(\beta_1)) + \sum_{i=1}^{k+1} f(s(\gamma_i)),$$

where $s(\gamma_{k+1}) = 0$ if item γ_{k+1} does not exist. We also know

$$f(s(\beta_1)) \geq f(s(m)) \geq f(s(\gamma_1)) \geq \cdots \geq f(s(\gamma_{k+1})).$$

For $k \geq 2$, it is not difficult to see that $OPT(t_1) \leq \max\{1.5, \frac{k+1}{k}\} ALG(t_1) = 1.5 \cdot ALG(t_1)$. Hence, after Step 2.$(b)(ii)$, we have $OPT(t_1) \leq 5/3 \cdot ALG(t_1)$.

Case 2.2.3: the execution for item m passes through Step 2.(c). Before item m is arrived, there is no type-1 or type-0 item, and the total value is less than $f(x_0)$. Item m is the unique type-1 item in the input and it is the largest item given so far. Thus all the discarded items must be type-2. Then by the similar arguments used in Case 2.2.2, we have

$$\frac{OPT(t_1)}{ALG(t_1)} \leq \frac{f(s(m)) + \sum_{i=1}^{k+1} f(s(\gamma_i))}{f(s(m)) + \sum_{i=1}^{k} f(s(\gamma_i))} \leq 1 + \frac{f(s(r_{k+1}))}{f(s(m)) + \sum_{i=1}^{k} f(s(\gamma_i))} \leq 1 + \frac{1}{k+1} \leq 1.5.$$

Case 2.2.4: the execution for item m passes through Step 3.(b). Then the total value is less than $f(x_0)$. If there is a type-1 item in the knapsack at time t_1, by the similar arguments used in Case 2.2.2, we have $\frac{OPT(t_1)}{ALG(t_1)} \leq \frac{5}{3}$. Else there are no type-1 or 0 items given before t_1, all the items given so far are type-2. If no type-2 item has been discarded, then $OPT(t_1) = ALG(t_1)$. Else $k \geq 2$ since $s(\gamma_1) \leq (1 - x_0) < 1/2$. By Lemma 5.7, we have $OPT(t_1) \leq \frac{k+1}{k} ALG(t_1) \leq 1.5 \cdot ALG(t_1)$. □

5.3.2 Applications of Algorithm ALG_2

Redefine x_0 in algorithm ALG_2 as below: $\frac{f(x_0)}{f(1-x_0)} = q$, where $q = \frac{1+\sqrt{5}}{2}$ is the golden ratio. In this subsection, we prove that if the convex function $f(\cdot)$ has the following property:

$$f\left(\frac{x_0}{2}\right) \geq \frac{f(x_0)}{q^2},$$

then the competitive ratio can be improved to q. For example, for any $1 \leq c \leq 1.3884$, function $f(x) = x^c$ satisfies the above property.

Theorem 5.16. Given a convex function $f(\cdot)$ with $f(0) = 0$, after redefining x_0 in ALG_2 such that $\frac{f(x_0)}{f(1-x_0)} = q$, if $f\left(\frac{x_0}{2}\right) \geq \frac{f(x_0)}{q^2}$, then algorithm ALG_2 is q-competitive.

Proof. Main ideas: we will use the same approach in Theorem 5.15 to prove this result. First assume that $\frac{OPT(t_0)}{ALG(t_0)} \leq q$ for a time step $t_0 \geq 1$, then we prove that $\frac{OPT(t_1)}{ALG(t_1)} \leq q$, where t_1 is the next time step of t_0. Just by redefining x_0 such that $\frac{f(x_0)}{f(1-x_0)} = q$, and following the proof in Theorem 5.15, observe that $\frac{OPT(t_1)}{ALG(t_1)} \leq q$ holds in Cases 2.1, 2.2.1, 2.2.2.1, 2.2.2.3, 2.2.3. We also find that if $\frac{OPT(t_1)}{ALG(t_1)} \leq q$ holds in Case 2.2.2.2, then it also holds in Case 2.2.4. To prove this theorem, we only need to prove that $\frac{OPT(t_1)}{ALG(t_1)} \leq q$ holds in Case 2.2.2.2, where the second largest type-2 item γ_2 is discarded or removed, and the largest item in the input is at most x_0.

Next we prove that after item γ_2 is discarded, then the total value by ALG_2 is at least $f(x_0)$.

In Case 2.2.2.2, there is a type-1 item in the knapsack, say m. And the largest type-2 item γ_1 is also selected in the knapsack. Since item γ_2 cannot fit together with items m and γ_1, we have

$$2s(\gamma_1) > 1 - s(m). \tag{5.4}$$

By Lemma 5.9, we have $x_0 \leq \frac{1}{q} \leq 0.619 \leq \frac{2}{3}$. Then

$$3x_0 < 2 \Rightarrow 1 - x_0 > 2x_0 - 1 \Rightarrow s(m) \geq 1 - x_0 > 2x_0 - 1. \tag{5.5}$$

Then

$$\begin{aligned}
f(s(m)) + f(s(\gamma_1)) &> f(s(m)) + f(\tfrac{1-s(m)}{2}) && \text{by (5.4)} \\
&\geq f(1-x_0) + f(s(m)/2 + x_0 - 1/2) && \text{by (5.5) and (5.2)} \\
&\geq f(1-x_0) + f(x_0/2) && \text{by } (x_0 + s(m) \geq 1) \\
&\geq \tfrac{f(x_0)}{q} + \tfrac{f(x_0)}{q^2} = f(x_0).
\end{aligned}$$

Since the size of the largest item is at most x_0, by Lemma 5.4, the optimal value is at most $f(x_0) + f(1-x_0)$, hence $\frac{OPT(t_1)}{ALG(t_1)} \leq q$ holds. □

Lemma 5.17. For any $1 \leq c \leq \log_2 q^2 \approx 1.3884$, if $f(x) = x^c$, then the competitive ratio of algorithm ALG_2 is $(1+\sqrt{5})/2$.

Proof. After defining x_0 as the root of equation $\frac{f(x)}{f(1-x)} = q$ for $x > \frac{1}{2}$, it is not difficult to see that $\frac{f(\frac{x_0}{2})}{f(x_0)} = 2^{-c} \geq \frac{1}{q^2}$. □

5.4 Lower Bound

In this section, we prove that the lower bound of the competitive ratio is $(1+\sqrt{5})/2$ for any convex function $f(\cdot)$. In [48] Iwama and Taketomi first proved that the lower bound of the competitive ratio is $(1+\sqrt{5})/2$ for function $f(x) = x$. Here we generalize their idea for any convex function $f(\cdot)$.

The main ideas are below: the adversary gives the first two items, one is large and one is small, but the two items cannot be accepted together, we have to make a decision which one we need to select; if the smaller one is selected then the adversary stops the input, else the adversary gives the third item which is a large item and can be accepted together with the smaller item in the knapsack, and stops the input. For each case, we can prove the competitive ratio cannot be smaller than $(1+\sqrt{5})/2$ for any online algorithm.

Theorem 5.18. Assume $f(0) = 0$. There is no online algorithm with competitive ratio strictly less than $(1+\sqrt{5})/2$.

Proof. Assume there exists an online algorithm A with competitive ratio $r < (1+\sqrt{5})/2$. Next we construct an input L and prove that $\frac{OPT(L)}{A(L)} > r$, i.e., the assume is wrong, there is no algorithm with a competitive ratio strictly less than $(1+\sqrt{5})/2$.

Define x_0 as the root of equation $\frac{f(x)}{f(1-x)} = q$ for $x > \frac{1}{2}$. By Lemma 5.9, we know x_0 exists and is unique. The first two items have size $x_0 + \epsilon$ and $1 - x_0$, where $\epsilon > 0$ is sufficiently small and satisfies the condition

$$\frac{f(x_0) + f(1-x_0)}{f(x_0+\epsilon)} > r.$$

Observe that $1 \leq r < q$ and $\frac{f(x_0)+f(1-x_0)}{f(x_0)} = q$ and $\frac{f(x_0)+f(1-x_0)}{f(1)} \leq 1$. Then following the similar approach used in Lemma 5.9, we can prove that such $\epsilon > 0$ must exist. After the second time step, there is at most one item selected in the knapsack. If the item with size $1 - x_0$ is selected, then we stop the input and have

$$\frac{OPT(L)}{A(L)} \geq \frac{f(x_0+\epsilon)}{f(1-x_0)} \geq q > r.$$

Else the item with size $x_0 + \epsilon$ is selected, then the third item with size x_0 is given and we stop the input. In this case, we have $OPT(L) = f(x_0)+f(1-x_0)$ and $ALG(L) \leq f(x_0+\epsilon)$. Therefore we have

$$\frac{OPT(L)}{A(L)} \geq \frac{f(x_0) + f(1-x_0)}{f(x_0+\epsilon)} > r.$$

Hence, an online algorithm with a competitive ratio strictly less than $(1 + \sqrt{5})/2$ does not exist. □

Chapter 6: Proportional Cost Buyback Problem は論文投稿中のため公表を延期します．5 年以内に出版予定です．

Chapter 7: Unit Cost Buyback Problem は論文投稿中のため公表を延期します．5 年以内に出版予定です．

Chapter 8: Optimal Composition Ordering Problems は論文投稿中のため公表を延期します．5 年以内に出版予定です．

Chapter 9

Conclusion

9.1 Summary

In this thesis, we have studied the competitive ratios for several variants of the online knapsack problem and related problems, and the time complexities for the optimal composition ordering problems.

In Chapter 4, we have given good competitive randomized algorithms for removable and non-removable online knapsack problems.

In Chapter 5, we have given $\frac{1+\sqrt{5}}{2}$-competitive algorithm for online knapsack problem under convex functions with specific properties. This competitive ratio coincides with the competitive ratio of the unweighted removable online knapsack problem.

In Chapter 6, we have presented optimal competitive algorithms for the proportional cost buyback problem. We have extended results by Babaioff *et al.* [5] and Constantin *et al.* [23] for the single element and the matroid cases to the case when each element has upper and lower bounds of weights. We have also presented an optimal competitive algorithm when the unweighted knapsack constraint with lower bound of weights.

In Chapter 7, we have proposed optimal competitive algorithms for the unit cost buyback problem when the constraint is a matroid constraint or the unweighted knapsack constraint.

In Chapter 8, we have introduced the optimal composition ordering problem and provided time complexities for the problem. We have showed that the maximum total composition ordering problem and the minimum total composition ordering problem are mutually reducible to one another, and the maximum partial composition ordering problem and the minimum partial composition ordering problem are also mutually reducible. We have presented a polynomial time algorithm for the maximum total composition ordering problem and the maximum partial composition ordering problem when the functions are monotone increasing and linear. We have also proposed polynomial time algorithm for the maximum partial composition ordering problem when the functions are piecewise increasing, i.e., $f_i(x) = \max\{a_i x + b_i, c_i\}$ $(a_i \geq 0)$. Moreover, we have proved that the optimal composition ordering problem is NP-hard even if the functions are monotone increasing, convex (concave), and at most 2-piece piecewise linear.

9.2 Open Problems

One important question is whether our algorithms and analysis for buyback problem can be extended to the more general case of packing problem, e.g., the *online packing problem.*

The online packing problem introduced by Buchbinder and Naor [16] is described as follows. Let us consider the following an integer programming formulation of a packing problem:

$$\begin{aligned} \text{maximize} \quad & \textstyle\sum_{i=1}^{n} b_i x_i \\ \text{s.t.} \quad & \textstyle\sum_{i=1}^{n} a_{i,j} x_i \leq c_i, \quad (\forall j \in [m]). \\ & x_i \geq 0, \quad (\forall i \in [n]). \end{aligned}$$

The values c_i ($i \in [n]$) are known in advance, but the profit function and the exact packing constraints are not known in advance. In the ith round, a new variable x_i is introduced to the algorithm, along with its set of coefficients $a_{i,j}$ ($j \in [m]$) and b_i. The algorithm can only increase the value of a variable x_i in the round in which it is given and cannot change the values of any previously given variables. The goal is to find a feasible solution that maximizes the objective function. Buchbinder and Naor [16] solved this problem with online primal dual method.

In a removable setting, the algorithm can also reduce the value of variables x_k ($k < i$) in the ith round. In a buyback setting, the algorithm can reduce the value of variables x_k ($k < i$) with some cost in the ith round. These problems are generalizations of the removable online knapsack problem or the buyback problem. Thus, another question is whether the primal dual method can be extended for this problem.

There are many open problems related to the optimal composition ordering problems. For example, it is unknown whether or not the minimum total composition ordering problem for monotone decreasing linear functions can be solved in polynomial time. To give pseudo-polynomial time algorithm or approximation algorithm for monotone increasing piecewise linear functions is also open.

References

[1] E. Anshelevich, A. Dasgupta, J. M. Kleinberg, É. Tardos, T. Wexler, and T. Roughgarden. The price of stability for network design with fair cost allocation. *SIAM Journal on Computing*, 38(4):1602–1623, 2008.

[2] B. V. Ashwinkumar. Buyback problem - approximate matroid intersection with cancellation costs. In *Automata, Language and Programming*, volume 6755, pages 379–390. Springer, 2011.

[3] B. V. Ashwinkumar and R. Kleinberg. Randomized online algorithms for the buyback problem. In *Internet and Network Economics*, volume 5929, pages 529–536. Springer, 2009.

[4] Y. Azar. On-line load balancing. In *Online Algorithms*, volume 1442 of *Lecture Notes in Computer Science*, pages 178–195. Springer, 1998.

[5] M. Babaioff, J. D. Hartline, and R. D. Kleinberg. Selling banner ads: Online algorithms with buyback. In *Proceedings of the 4th Workshop on Ad Auctions*, 2008.

[6] M. Babaioff, J. D. Hartline, and R. D. Kleinberg. Selling ad campaigns: Online algorithms with cancellations. In *Proceedings of the 10th ACM Conference on Electronic Commerce*, pages 61–70, 2009.

[7] M. Babaioff, N. Immorlica, D. Kempe, R. Kleinberg, M. Babaioff, N. Immorlica, D. Kempe, and R. Kleinberg. A knapsack secretary problem with applications. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 16–28, 2007.

[8] M. Babaioff, N. Immorlica, and R. Kleinberg. Matroids, secretary problems, and online mechanisms. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 434–443, 2007.

[9] N. Bansal, N. Buchbinder, A. Madry, and J. Naor. A polylogarithmic-competitive algorithm for the k-server problem. In *Proceedings of IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 267–276, 2011.

[10] R. E. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[11] S. Ben-David, A. Borodin, R. M. Karp, G. Tardos, and A. Wigderson. On the power of randomization in on-line algorithms. *Algorithmica*, 11(1):2–14, 1994.

[12] E. Biyalogorsky, Z. Carmon, G. E. Fruchter, and E. Gerstner. Research note: Overselling with opportunistic cancellations. *Marketing Science*, 18(4):605–610, 1999.

[13] L. A. Blady. A study of replacement algorithms for virtual storage computers. *IBM Systems Journal*, 5:78–101, 1966.

[14] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cam-

bridge University Press, 1998.

[15] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[16] N. Buchbinder and J. Naor. Online primal-dual algorithms for covering and packing problems. In *Proceedings of the 13th Annual European Symposium*, pages 689–701, 2005.

[17] J.-Y. Cai, P. Cai, and Y. Zhu. On a scheduling problem of time deteriorating jobs. *Journal of Complexity*, 14(2):190–209, 1998.

[18] T. C. E. Cheng and Q. Ding. The complexity of scheduling starting time dependent tasks with release times. *Information Processing Letters*, 65(2):75–79, 1998.

[19] T. C. E. Cheng and Q. Ding. Single machine scheduling with deadlines and increasing rates of processing times. *Acta Informatica*, 36(9-10):673–692, 2000.

[20] T. C. E. Cheng and Q. Ding. Scheduling start time dependent tasks with deadlines and identical initial processing times on a single machine. *Computers & Operations Research*, 30(1):51–62, 2003.

[21] T. C. E. Cheng, Q. Ding, M. Y. Kovalyov, A. Bachman, and A. Janiak. Scheduling jobs with piecewise linear decreasing processing times. *Naval Research Logistics*, 50(6):531–554, 2003.

[22] T. C. E. Cheng, Q. Ding, and B. Lin. A concise survey of scheduling with time-dependent processing times. *European Journal of Operational Research*, 152(1):1–13, 2004.

[23] F. Constantin, J. Feldman, S. Muthukrishnan, and M. Pál. An online mechanism for ad slot reservations with cancellations. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1265–1274, 2009.

[24] B. Dean, M. Goemans, and J. Vondrák. Adaptivity and approximation for stochastic packing problems. In *Proceedings of the sixteenth annual ACM-SIAM Symposium on Discrete Algorithms*, pages 395–404. Society for Industrial and Applied Mathematics, 2005.

[25] B. Dean, M. Goemans, and J. Vondrák. Approximating the stochastic knapsack problem: the benefit of adaptivity. *Mathematics of Operations Research*, 33(4):945–964, 2008.

[26] T. S. Ferguson. Who solved the secretary problem? *Statical Science*, 4(3):282–289, 1989.

[27] A. Fiat, Y. Rabani, and Y. Ravid. Competitive k-server algorithms. In *Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science*, pages 454–463, 1990.

[28] M. Fredman and M. Saks. The cell probe complexity of dynamic data structures. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 345–354, 1989.

[29] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34(3):596–615, 1987.

[30] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman New York, 1979.

[31] S. Gawiejnowicz. Scheduling deteriorating jobs subject to job or machine availability constraints. *European Journal of Operational Research*, 180(1):472–478, 2007.

[32] S. Gawiejnowicz. *Time-Dependent Scheduling*. Springer, 2008.

[33] S. Gawiejnowicz and L. Pankowska. Scheduling jobs with varying processing times. *Information Processing Letters*, 54(3):175–178, 1995.

[34] R. L. Graham. Bounds for certain multiprocessor anomalies. *Bell System Technical Journal*, 45:1563–1581, 1966.

[35] J. N. Gupta and S. K. Gupta. Single facility scheduling with nonlinear processing times. *Computers & Industrial Engineering*, 14(4):387–393, 1988.

[36] X. Han, K. Iwama, and G. Zhang. Online removable square packing. *Theory of Computing Systems*, 43:38–55, 2008.

[37] X. Han, Y. Kawase, and K. Makino. Online unweighted knapsack problem with removal cost. *Algorithmica*, pages 1–16, 2013.

[38] X. Han, Y. Kawase, and K. Makino. Randomized algorithms for removable online knapsack problems. In *Frontiers in Algorithmics and Algorithmic Aspects in Information and Management*, volume 7924, pages 60–71. Springer, 2013.

[39] X. Han, Y. Kawase, K. Makino, and H. Guo. Online knapsack problem under convex function. *Theoretical Computer Science*, In Press.

[40] X. Han and K. Makino. Online minimization knapsack problem. In *Approximation and Online Algorithms*, volume 5893, pages 182–193, 2010.

[41] X. Han and K. Makino. Online removable knapsack with limited cuts. *Theoretical Computer Science*, 411:3956–3964, 2010.

[42] R. Hassin and S. Rubinstein. Robust matchings. *SIAM Journal on Discrete Mathematics*, 15:530–537, 2002.

[43] K. I.-J. Ho, J. Y.-T. Leung, and W.-D. Wei. Complexity of scheduling tasks with time-dependent execution times. *Information Processing Letters*, 48(6):315–320, 1993.

[44] E. Horowitz and S. Sahni. Computing partitions with applications to the knapsack problem. *Journal of the ACM*, 21:277–292, 1974.

[45] O. H. Ibarra and C. E. Kim. Fast approximation algorithms for the knapsack and sum of subset problem. *Journal of the ACM*, 22:463–468, 1975.

[46] S. Irani and A. Karlin. *Online Computation*, volume Approximation Algorithms for NP-Hard Problems, chapter 13. PWS Publishing, 1997.

[47] H. Ito, S. Kiyoshima, and Y. Yoshida. Constant-time approximation algorithms for the knapsack problem. In *Theory and Applications of Models of Computation*, pages 131–142, 2012.

[48] K. Iwama and S. Taketomi. Removable online knapsack problems. In *Proceeding of the 29th International Colloquium on Automata, Languages and Programming*, pages 293–305, 2002.

[49] K. Iwama and G. Zhang. Optimal resource augmentations for online knapsack. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques.*, pages 180–188. Springer, 2007.

[50] N. Kakimura and K. Makino. Robust independence systems. *SIAM Journal on Discrete Mathematics*, 27(3):1257–1273, 2013.

[51] A. R. Karlin, C. Kenyon, and D. Randall. Dynamic tcp acknowledgement and other stories about e/(e-1). In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 502–509, 2001.

[52] A. R. Karlin, M. S. Manasse, L. A. McGeoch, and S. Owicki. Competitive randomized algorithms for nonuniform problems. *Algorithmica*, 11(6):542–571, 1994.

[53] A. R. Karlin, M. S. Manasse, L. Rudolph, and D. D. Sleator. Competitive snoopy caching. *Algorithmica*, 3(1):70–119, 1988.

[54] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, b, pages 85–103. Springer, 1972.

[55] Y. Kawase, X. Han, and K. Makino. Unit cost buyback problem. In *In proceedings of the 24th International Symposium on Algorithms and Computation*, pages 435–445, 2013.

[56] Y. Kawase, K. Makino, and K. Seimi. Optimal composition ordering problems. In preparation.

[57] H. Kellerer, R. Mansini, and M. G. Speranza. Two linear approximation algorithms for the subset-sum problem. *European Journal of Operational Research*, 120:289–296, 2000.

[58] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, 2004.

[59] D. E. Knuth. *The Art of Computer Programming, Volume 3: (2nd ed.) Sorting and Searching*. Addison-Wesley, 1998.

[60] B. Korte and J. Vygen. *Combinatorial Optimization: Theory and Algorithms.*

Springer, 2002.

[61] E. Koutsoupias and C. H. Papadimitriou. On the k-server conjecture. *Journal of the ACM*, 42(5):971–983, 1995.

[62] E. Koutsoupias and C. H. Papadimitriou. Worst-case equilibria. In *Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science*, pages 404–413, 1999.

[63] L. H. Loomis. On a theorem of von neumann. In *Proceedings of the National Academy of Sciences of the U.S.A.*, volume 32, pages 213–215, 1946.

[64] G. S. Lueker. Average-case analysis of off-line and on-line knapsack problems. *Journal of Algorithms*, 29(2):277–305, 1998.

[65] M. S. Manasse, L. A. McGeoch, and D. D. Sleator. Competitive algorithms for server problems. *Journal of Algorithms*, 11:208–230, 1990.

[66] A. Marchetti-Spaccamela and C. Vercellis. Stochastic on-line knapsack problems. *Mathematical Programming*, 68:73–104, 1995.

[67] S. Martello and P. Toth. A new algorithm for the 0-1 knapsack problem. *Management Science*, 34:633–644, 1988.

[68] R. L. Mattison, J. Gecsei, D. R. Slutz, and I. L. Traiger. Evaluation techniques for storage hierarchies. *IBM System Journal*, 9(2), 1971.

[69] N. Megow and J. Mestre. Instance-sensitive robustness guarantees for sequencing with unknown packing and covering constraints. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 495–504, 2013.

[70] O. I. Melnikov and Y. M. Shafransky. Parametric problem of scheduling theory. *Cybernetics*, 15:352–357, 1980.

[71] G. Mosheiov. Scheduling jobs under simple linear deterioration. *Computers & Operations Research*, 21(6):653–659, 1994.

[72] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[73] C. T. Ng, M. Barketau, T. C. E. Cheng, and M. Y. Kovalyov. "Product partition" and related problems of scheduling and systems reliability: Computational complexity and approximation. *European Journal of Operational Research*, 207:601–604, 2010.

[74] N. Nisan, T. Roughgarden, É. Tardos, and V. V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, 2007.

[75] S. Oveis Gharan and J. Vondrák. On variants of the matroid secretary problem. In *Proceedings of the 19th Annual European Symposium on Algorithms*, pages 335–346, 2011.

[76] J. G. Oxley. *Matroid Theory*. Oxford University Press, New York, 1992.

[77] D. Pisinger. Linear time algorithm for knapsack problem with bounded weights. *Journal of Algorithms*, 33:1–14, 1999.

[78] S. S. Seiden. A guessing game and randomized online algorithms. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of computing*, 2000.

[79] D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the Association for Computing Machinery*, 28(2):202–208, 1985.

[80] D. D. Sleator and R. E. Tarjan. Self-adjusting binary search trees. *Journal of the ACM*, 32(3):652–686, 1985.

[81] V. S. Tanaev, V. S. Gordon, and Y. M. Shafransky. *Scheduling Theory: Single-Stage Systems*. Kluwer Academic Publishers, 1994.

[82] R. E. Tarjan. Efficiency of a good but not linear set union algorithm. *Journal of the ACM*, 22(2):215–225, 1975.

[83] R. E. Tarjan. Amortized computational complexity. *SIAM Journal on Algebraic and Discrete Methods*, 6(2):306–318, 1985.

[84] R. E. Tarjan and J. van Leeuwen. Worst-case analysis of set union algorithms. *Journal of the ACM*, 31(2):245–281, 1984.

[85] V. V. Vazirani. *Approximation algorithms*. Springer, 2004.

[86] J. von Neumann. Zur Theorie der Gesellschaftsspiele. *Mathematische Annalen*, 100:295–320, 1928.

[87] W. Wajs. Polynomial algorithm for dynamic sequencing problem. *Archiwum Automatyki i Telemechaniki*, 31(3):209–213, 1986.

[88] H. Whitney. On the abstract properties of linear dependence. *American Journal of Mathematics*, 57:509–533, 1935.

[89] D. P. Williamson and D. B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, New York, NY, USA, 1st edition, 2011.

[90] A. Yao. Probabilistic computations: Toward a unified measure of complexity. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, pages 222–227. IEEE, 1977.

[91] Y. Zhou, D. Chakrabarty, and R. Lukose. Budget constrained bidding in keyword auctions and online knapsack problems. *Internet and Network Economics*, pages 566–576, 2008.