A Survey of Yao's Minimax Principle

Brian Shimanuki bshimanuki@mit.edu
Mayuri Sridhar mayuri@mit.edu
Chelsea Voss csvoss@mit.edu

May 13, 2015

Abstract

We introduce Yao's minimax principle [9] and present a series of online and non-local games for which we illustrate the use of Yao's principle in bounding the e ectiveness of an optimal solution. In particular, we look at the removable online knapsack problem [5], the online coloring co-interval graph problem [10], and a multiplayer extension of the CHSH quantum game [1] to demonstrate how combining an algorithm with an input distribution can tighten the interval on which we know the optimal solution lies.

1 Introduction

Yao's minimax principle, proposed by Andrew Yao in 1977, was rst used to study lower bounds on the optimal running times of randomized algorithms for a particular problem. In this paper, we survey a series of games for which Yao's principle can be used to bound the e ectiveness of all randomized algorithms against an adversary. In this section, we present Yao's principle and give a proof of it.

But rst, we clarify the de nition of a randomized algorithm.

Randomized Algorithm An algorithm which has the capabilities of a deterministic algorithm, but with additional access to a stream of uniformly random bits.

Thus we can model a randomized algorithm as a deterministic algorithm with an additional secondary input of a stream of random bits. Conditioned on this stream, this is an entirely deterministic algorithm. Thus a randomized algorithm is modeled by a set of deterministic algorithms from which the randomized algorithm chooses at runtime with some probability distribution, before seeing the primary input.

Yao's Minimax Principle Any randomized algorithm when run on its own worst-case input can perform no better than the best deterministic algorithm is expected to perform on any speci c distribution of inputs.

More formally,

$$\max_{x \in X} \mathop{\mathbf{E}}_{r \in R} \texttt{cost}(r, x) \qquad \min_{r \in R} \mathop{\mathbf{E}}_{x \in X} \texttt{cost}(r, x)$$

where

X = the set of all possible inputs to the algorithm.

R = the set of all possible deterministic algorithms.

X = any probability distribution over inputs to the algorithm.

R = any randomized algorithm, ie., any probability distribution over deterministic algorithms.

$\texttt{cost}(r, x)$ = the cost of the algorithm r running on input x. Commonly this is running time, but in this paper we will consider other quantities to optimize.

In other words, if you can gure out how well the best deterministic algorithm runs against any distribution X of random inputs, you know that any randomized algorithm you formulate can do no better against its worst-case input. This gives you a lower bound on the randomized algorithm's running time. Although X need not be the worst possible distribution over random inputs for this principle to be useful, note that the worse X is as an input distribution, the tighter the resulting lower bound will be.

1.1 Proof of Yao's Principle

Consider an arbitrary input distribution X. Let d_X be the best deterministic algorithm for input distribution X, that is, the deterministic algorithm which minimizes expected cost $\mathbf{E}_{x2X}\,\texttt{cost}(d_X, x)$.

Now since any randomized algorithm R can be modeled as a distribution of deterministic algorithms, the expected cost of R on X is $\mathbf{E}_{x2X}\,\mathbf{E}_{r2R}\,\texttt{cost}(r, x)$, or $\mathbf{E}_{r2R}\,\mathbf{E}_{x2X}\,\texttt{cost}(r, x)$ by linearity of expectation. This is the weighted average of the expected costs of the deterministic algorithms in the distribution of R. Since the cost of each of these is at least $\mathbf{E}_{x2X}\,\texttt{cost}(d_X, x)$, we have

$$\mathop{\mathbf{E}}_{r2R}\mathop{\mathbf{E}}_{x2X}\texttt{cost}(r, x) \quad \mathop{\mathbf{E}}_{x2X}\texttt{cost}(d_X, x).$$

The cost of R on its worst-case input cannot be greater than the cost of R on this input distribution. Then

$$\max_{x2X}\mathop{\mathbf{E}}_{r2R}\texttt{cost}(r, x) \quad \mathop{\mathbf{E}}_{x2X}\mathop{\mathbf{E}}_{r2R}\texttt{cost}(r, x) = \mathop{\mathbf{E}}_{r2R}\mathop{\mathbf{E}}_{x2X}\texttt{cost}(r, x) \quad \mathop{\mathbf{E}}_{x2X}\texttt{cost}(d_X, x).$$

Now d_X is simply $\arg\min_{r2R}\mathbf{E}_{x2X}\,\texttt{cost}(r, x)$ by de nition. Thus we have

$$\max_{x2X}\mathop{\mathbf{E}}_{r2R}\texttt{cost}(r, x) \quad \min_{r2R}\mathop{\mathbf{E}}_{x2X}\texttt{cost}(r, x).$$

2 Removable Online Knapsack Problem

In this section, we present the 2-competitive solution of Han et al. [5] to the removable online knapsack problem and use Yao's principle to show $1 + 1/e$ is a lower bound on the competitiveness of any algorithm against an oblivious adversary.

The **Knapsack Problem** asks: Given a set of items e_i with weights w_i and values v_i, select a subset with the maximum sum of values such that the sum of the weights is at most the capacity of the knapsack. We call this sum of values the *value* of the knapsack. The o ine version is a classic NP-hard optimization problem, though it admits a simple dynamic programming pseudo-polynomial solution as well as a FPTAS. Here, we consider the *online* version, meaning each item is given to the algorithm after the algorithm makes a decision on the previous item. *Removable* means the knapsack is a working set, from which we can remove items and add new items, but not add previously removed items. At all times, the knapsack is limited by its capacity. The objective is to maximize the value of the nal knapsack. We say an algorithm is α-competitive if the expected value of the nal knapsack is at least $1/\alpha$ times the expected value of the optimal knapsack.

We assume the knapsack has capacity 1. This is valid because only weights relative to the knapsack's capacity matters for the capacity constraint, so we can always scale down to a unit capacity.

2.1 Upper Bound

To prove an upper bound on the competitiveness of algorithms which yield a solution to the removable online knapsack problem, we present a 2-competitive randomized algorithm.

Since the online nature of the problem is what makes it impossible to always achieve the optimal solution, it is insightful to rst consider algorithms which behave similarly in both online and o ine situations. We look at *MAX* and *GREEDY*.

MAX simply selects the item with the largest value whose weight does not exceed 1. In the o ine case, we can process each item in an arbitrary order and keep track of the best item so far. Therefore, in the online case, we can do exactly the same with the order given to us, storing the best item so far in our knapsack.

GREEDY selects the k items with the highest ratios of value to weight v_i/w_i, maximizing k while keeping the sum of the weights of the selected items at most 1. In the o ine case, this can be done trivially by sorting the items by ratio and then selecting the highest k items. An online algorithm can perform at least as well by keeping a working set of the items with the highest ratio while processing each input.

Algorithm 1: online *GREEDY*

```
S ;
foreach item e_i with w_i 1, in order of arrival do
    S S [ fe_ig
    while sum_{e_j 2 S} w_j > 1 do
        remove the item from S with the smallest ratio
```

When the sum of the weights in S is greater than 1, an item with ratio less than that of the k items with the largest ratios must be in S; therefore the k items will never be removed. Thus S is a superset of the items the o ine *GREEDY* algorithm selects.

Neither *MAX* nor *GREEDY* alone is competitive. Consider an input sequence $f(w_i, v_i)g$ of $f(\epsilon, 2\epsilon),\ (\epsilon, \epsilon),\ (\epsilon, \epsilon),\ \ldots g$ on the *MAX* algorithm and $f(\epsilon, 2\epsilon),\ (1, 1)g$ on the *GREEDY*

algorithm. In fact, no deterministic online algorithm is competitive with a constant factor [6]. However, using both with randomness can yield a competitive algorithm.

Theorem 1. *Running* MAX *and* GREEDY *uniformly at random is a 2-competitive algorithm [5].*

Proof. For a given input sequence T, let $GREEDY(T)$, $MAX(T)$, and $OPT(T)$ be the values of the sets returned by *GREEDY*, *MAX*, and an optimal solution, respectively. Consider the *fractional* knapsack solution to the o ine problem with input sequence T: that is, when we are allowed to take fractional parts of items. Then, a greedy solution is optimal: Sort by ratio, and take the highest ratio items until the knapsack is full, potentially taking only part of the last item in the knapsack. The fractional knapsack problem is a relaxation of the integral knapsack problem, so the fractional solution is not worse than the integral solution.

Now the o ine *GREEDY* algorithm by de nition selects all the items in the fractional solution except the nal fractional item (if one exists). Thus the online *GREEDY* algorithm also selects all except the fractional item. Finally, the *MAX* algorithm selects an item with value at least that of the fractional item, so $GREEDY(T) + MAX(T) \quad OPT(T)$. Thus the competitive ratio is

$$\frac{2 \quad OPT(T)}{GREEDY(T) + MAX(T)} \quad 2. \qquad \square$$

2.2 Lower Bound

We use Yao's principle to prove a lower bound of $1 + 1/e$ for the competitive ratio of any removable online knapsack algorithm.

To do so, we must nd a distribution of inputs for which no deterministic algorithm performs \well", which means the expected value of the output from any deterministic algorithm is at most $(1 + 1/e)^{\ 1}$ times the expected optimum. As the power of an o ine algorithm is in the additional information it has over an online algorithm, it is reasonable that in constructing our \bad" input, we would want to limit the information the algorithm has about future items. Thus it is not surprising that we will construct a family of inputs where each input is a pre x of the same base sequence.

For simplicity in analysis, we will let the rst item $e_0 = (w_0, v_0) = (1, 1)$ with future items having much smaller weights and values but a higher ratio. Then the problem is essentially reduced to determining when to remove the rst item from the knapsack.

This is reminiscent of the *ski rental problem*, one of the most well-understood online problems [7], where a client must choose whether and when to buy skis for a xed cost versus paying a smaller cost each day. Our knapsack problem is the reverse of this: instead of deciding when to switch from renting skis to buying skis to minimize total cost, we are deciding when to switch from having a single large item in our knapsack to getting a stream of smaller items with a better ratio to maximize total value.

Finally, we want the expected value of removing e_0 and selecting all following items to be close to 1 at any given point. A simple way to achieve this is to let the remaining items have a constant value and have the p_k form an exponential distribution.

This leads us towards the input sequence given by Han et al. [5]:

$$(1,1),\ \underbrace{(1/n^2,1/n),\ (1/n^2,1/n),\ldots\ (1/n^2,1/n)}_{k \text{ items}} \tag{1}$$

for a given value of n and where there are k + 1 items with probability p_k = $\frac{1\ e^{-1=n}}{1\ e^{-n}} e^{\ (k\ 1)=n}$ for k = $1,2,\ldots,n^2$.

Theorem 2. *No online algorithm has competitive ratio less than* 1 + $1/e$ *for the removable online knapsack problem [5].*

Proof. We apply Yao's principle on the input distribution (1). The optimal strategy with full information is to keep e_0 for k n and remove e_0 immediately otherwise. This leads us to the expected optimal value

$$\sum_{i=1}^{n} 1\ \ p_i + \sum_{i=n+1}^{n^2} \frac{i}{n} p_i = \frac{1\ \ e^{\ 1=n}}{1\ \ e^{\ n}} \left(\sum_{i=1}^{n} e^{\ (i\ \ 1)=n} + \frac{1}{n} \sum_{i=n+1}^{n^2} i\ \ e^{\ (i\ \ 1)=n} \right),$$

which has value 1 + $1/e$ as n approaches in nity.

Since on this set of inputs, the initial item has weight 1 and the sum of the rest of the items is at most 1, any optimal algorithm having removed the initial item can perform a greedy algorithm and put all future items in the knapsack. Thus an optimal online deterministic algorithm can only decide how many items it must see before it removes e_0. For inputs $k < l$, the algorithm keeps e_0 and removes all incoming items, so it obtains value 1. For k l, the algorithm removes e_0 when the l^{th} item appears, and accumulates items e_l through e_k. Then the expected value of this algorithm is

$$\sum_{i=1}^{l\ \ 1} 1\ \ p_i + \sum_{i=l}^{n^2} \frac{i\ \ l}{n} p_i = \frac{1\ \ e^{\ 1=n}}{1\ \ e^{\ n}} \left(\sum_{i=1}^{l\ \ 1} e^{\ (i\ \ 1)=n} + \frac{1}{n} \sum_{i=l}^{n^2} (i\ \ \ 1) e^{\ (i\ \ 1)=n} \right),$$

which has value 1 as n approaches in nity.

By Yao's principle, since every deterministic algorithm nds (1 + $1/e$) $^{\ 1}$ of the optimum set on this distribution of inputs, any online algorithm performs at least as poorly. Thus any online algorithm to this problem has competitive ratio at least 1 + $1/e$. □

3 Online Coloring Co-Interval Graphs

This section studies the online problem of coloring co-interval graphs as described by Zarrabi-Zadeh [10]. This problem is set up as follows:

Input A set of intervals on the real line in arbitrary order.

Output A color matched to each interval, such that `color`(I_k) 6= `color`(I_j) when intervals I_k and I_j are such that $j < k$ and intervals I_j and I_k don't intersect.

Since this is an online problem, the algorithm must assign colors as it goes, without knowing the complete set of intervals. That is, the algorithm will only receive information about I_{j+1} after it has assigned colors to all intervals I_k where $j \quad k$.

First, we present the *FIRSTFIT* approximation algorithm, which solves this problem with a competitive ratio of at most 2. Then, using Yao's principle, we can prove that there exist no randomized algorithm that achieves a competitive ratio that's better than $3/2$.

3.1 Upper Bound

To solve the problem of assigning intervals di erent colors, the set of intervals is rst constructed as a graph. That is, the input to our problem becomes:

V := Set of intervals where v_i represents the i^{th} interval

E := Set of edges where (u, v) *2* E i interval u and interval v are disjoint

Note that any two vertices that aren't connected by an edge can be colored in the same color, since those vertices represent overlapping intervals. Thus, in such a graph, our problem is reduced to nding the minimum number of colors such that no connected vertices are the same color.

The *FIRSTFIT* algorithm, as follows, is the simplest solution to this problem.

Algorithm 2: *FIRSTFIT* algorithm

L ordered list of colors
foreach *new vertex v* **do**
    assign v the rst color in L that doesn't cause a collision

Essentially, the algorithm maintains a list of colors that can be used. These colors are sorted in an arbitrary order. The rst vertex is simply assigned the smallest color. For any future vertex v, the algorithm scans its list of colors. A *collision* between vertices u and v at color k occurs if u is colored k and v is connected to u by an edge in E. If $k = L[i]$ and a collision occurs, then we increment i. We assign v the color at $L[j]$, where j doesn't cause any collisions, and we try to minimize j.

It has been shown by Gyarfas and Lehel [4] that *FIRSTFIT* uses at most $2\omega + 1$ colors on the best competitive ratio that can be obtained.

3.2 Lower B und

Yao's minimax principle proves that the expected competitive ratio of an optimal deterministic

deterministic algorithms are at least α-competitive, then α is the best ratio that can be obtained by using randomization.

To de ne such a set of inputs, essentially, we need to nd a set of intervals that satisfy the condition that any deterministic algorithm is at most 3/2-competitive. The idea behind constructing such a set of intervals is that any deterministic algorithm will make the same decision on any speci c interval given the same information, regardless of what the future intervals are.

3.2.1 Problem Construction

For 1 i k for a given value of k, the intervals de ned by Zarrabi-Zadeh [10] are:

$$a_i = [3i \quad 3, 3i \quad 2],$$
$$b_i = [3i + 1, 3i + 2], \text{ and}$$
$$c_i = [3i + 2, 1).$$

We can then de ne a block as

$$B_i = \begin{cases} b_1, c_1 & \text{if } i = 1 \\ a_i, b_i, c_i & \text{if } i \ 2 \ [2, k) \\ a_k, b_k & \text{if } i = k. \end{cases}$$

From this set of blocks, we can choose a set of k input sequences, simply by de ning our input sequence X_i as the concatenation of blocks B_1, B_2, ..., B_i, in order. Let I be our set of all possible inputs fX_i where i 2 $[1, k]g$. It can be shown that the minimum number of colors required to color X_i without any edges sharing a color is i.

To use Yao's principle on this input distribution, we need to show that the expected competitive ratio will be at least 3/2 if we set up a probability distribution over all possible X_i. The optimal deterministic algorithm that we will examine per each X_i input is *FIRSTFIT*; as previously shown, this algorithm uses 2ω 1 colors on each X_i. In this case, the chromatic number $\omega = i$, so, the expected number of colors for a given X_i is exactly $2i$ 1.

For a given X_i, a new color is chosen (assuming *FIRSTFIT*) on each a_m and each b_m, since the blocks are given in order. A probability distribution P can be de ned over all X_i, where p_i is the probability that X_i is chosen as input, as follows:

$$p_i = \begin{cases} \frac{2^k}{2^k \ 1} \quad \frac{i}{2^i} & \text{if } i < k \\ \frac{k}{2^k \ 1} & \text{if } i = k. \end{cases}$$

The expected competitive ratio of *FIRSTFIT* on this distribution then becomes $\sum_{i=1}^{k} p_i \frac{2i \ 1}{i}$, since *FIRSTFIT* uses $2i$

If no deterministic algorithm can obtain a better competitive ratio on this set of inputs, then this is the best that a randomized algorithm could achieve.

3.2.2 Deterministic Algorithm Bounds

Theorem 3. *No deterministic algorithm can obtain a better competitive ratio on the distribution of inputs I.*

Proof. To show this, we choose an arbitrary deterministic algorithm A. We can then de ne d_i as the decision that A makes on receiving interval c_i, where d_i = 1 i a new color is chosen for c_i. In an optimal solution, all c_i can be colored the same as b_i, since all b_i don't intersect any preceding a_j or b_j, but intersect all previous c_j.

Given a sequence of decisions $D = hd_1, d_2, \ldots, d_{k\ 1}i$, then we can de ne a deterministic algorithm *FF* as follows.

> If we receive an interval c_j, we check what d_j is and assign a new color i d_j = 1. Else, we assign it the same color as b_j.
>
> If we receive an interval of type a or b, we behave as the online *FIRSTFIT* algorithm normally does.

Lemma 4. *Among all deterministic online algorithms,* FF *behaves optimally and chooses the minimum number of colors for any input I.*

Proof. This can be shown by induction over the minimum number of colors on an input of size m. □

Lemma 5. *The competitive ratio of* FF *is ρ.*

Proof. This proof follows by induction:

De ne $P(m)$ to be true if ρ is an upper bound for the competitive ratio of *FF* in the case where there are m ones in our sequence of decisions D. $P(0)$ holds trivially { in a sequence of all zeros, *FF* behaves exactly the same as *FIRSTFIT* and our competitive ratio is still approximately 2.

Using strong induction, we assume that $P(k)$ is true for all k 2 $[0, m\ \ 1]$.

By looking at the bits of D where FF with m ones would di er from FF with $m\ \ 1$ ones, we can derive the formula

$$\rho_m = \rho_{m\ 1} + \rho_i \left(\frac{1}{i}\right) \sum_{j=i+1}^{k} \rho_i \left(\frac{1}{i}\right).$$

Substituting $\rho_i = \frac{2^k}{2^k\ 1}\ \frac{i}{2^i}$ gives us that the last term sums to exactly $\rho_i\ \ 1/i$. This gives us that $\rho_m = \rho_{m\ 1}$, and the proof follows by induction. □

In turn, this implies that the expected competitive ratio of any deterministic algorithm is at most ρ, which shows that no deterministic algorithm has a better competitive ratio on I. □

By choosing k arbitrarily large, ρ tends to $3/2$. As we have now bounded the behavior of any deterministic algorithm on the distribution of inputs I, Yao's principle thus gives us a lower bound of $3/2$ as the competitive ratio on any randomized algorithm.

4 Yao's Principle in Quantum Information

Yao's principle isn't just used for proving lower bounds on the competitiveness of algorithms. Another application is found in the study of certain games in quantum information, where Yao's principle can show worst-case bounds of game strategies. In this section, we will examine this use, as demonstrated in *Worst Case Analysis of Non-local Games* by Ambainis et al. [1].

4.1 Quantum Preliminaries

Consider a game between two players and an adversary, where the two players receive random input from the adversary and must each return outputs jointly satisfying a condition without communicating with each other during the game. Analysis under classical physics conditions and quantum physics conditions has demonstrated quantum information allows for better strategies.

An example of such a game is the Clauser{Horne{Shimonyi{Holt (CHSH) game [3]:

> The two players may collude to set up a strategy beforehand but may not communicate afterwards.
>
> The adversary chooses a random $\vec{x} = (x_1, x_2)$ from $X = f(0,0),(0,1),(1,0),(1,1)g$. The random choice is uniform, with $\frac{1}{4}$ probability each.
>
> The players 1 and 2 receive the values x_1 and x_2, respectively.
>
> In the quantum setting, the players 1 and 2 may also possess an entangled 2-part quantum state $j\psi i$, which they may measure.
>
> Using their strategy, the players 1 and 2 output the values $a_1(x_1)$ and $a_2(x_2)$, respectively.
>
> The players win if $a_1 \quad a_2 = x_1 \wedge x_2$.

Both under classical physics and under any \local hidden variable theory" { an explanation of quantum phenomena as being due to extra local hidden variables { the best that the players can achieve is a win probability of 0.75. If the players are allowed to share entangled quantum bits, however, then they can win more often, with a win probability of $\frac{1}{2} + \frac{1}{2\sqrt{2}} = 0.8535\ldots$ [1, 3].

The CHSH game was designed as a testable experiment, and experiments have since con rmed that the universe is consistent with quantum entanglement, and not consistent with local hidden variable theories [2].

In *Worst Case Analysis of Non-local Games* by Ambainis et al., the authors examine generalized games beyond the simple CHSH game, and analyze the worst-case performance of strategies in both classical and quantum environments [1].

4.2 Generalized CHSH

The CHSH game is played between two players and an adversary. We generalize the game from two players to n players. Ambainis et al. call this the *n-party AND game* [1].

The authors note that this game has not been studied before because of the simplicity of its strategies in the classical setting: by outputting $a_i = 0$, the players are guaranteed to win unless the adversary happened to pick $x_i = 1$. Therefore, against an adversary that always chooses $\vec{x}$ uniformly, the players can win with probability $1 - 2^n$.

However, this game becomes more interesting when we consider worst-case bounds: if the adversary is allowed to choose π, the optimal strategy for the players becomes less trivial. The authors show that $\lim_{n \to \infty} \Pr[win] = 2/3$ in both the classical case and the quantum case [1].

The n-party AND game and the CHSH game share the same win condition, $\bigoplus \vec{a} = \bigwedge \vec{x}$.

ratio $\omega_q(G)/\omega_c(G)$ [1]. Interestingly, for every possible proposition win, the separation ratio $\omega_q(G)/\omega_c(G)$ when the adversary chooses the worst distribution π *separately* for the classical and quantum case is at most the maximum value of $\omega_q(G)/\omega_c(G)$ over a xed distribution π. This proof uses Yao's minimax principle.

Theorem 6.

$$\frac{\omega_q(G)}{\omega_c(G)} \quad \max \frac{\omega_q(G)}{\omega_c(G)}.$$

Proof. As with the earlier applications of Yao's minimax principle, the adversary chooses a strategy over a distribution of inputs $\vec{x}$ to pass to the players, and the players may still use randomized algorithms as strategies. However, instead of letting the cost be *running time*, the cost is the *probability of winning*. The adversary attempts to minimize cost and the two players' joint algorithms attempt to maximize cost.

Recall Yao's minimax principle, but instead of stating that any (including the best) randomized algorithm when run on its own worst-case input can perform no better than the best deterministic algorithm when run on any (including the worst) speci c distribution of inputs, we will now state that the players' best randomized algorithms when run on their worst-case input can perform no better than the players' best deterministic algorithms when run on the worst distribution π of inputs from the adversary.

The players' best deterministic algorithm against the worst distribution π over random inputs from the adversary is $\max_a \min \mathbf{E}_{x2pi}\, \mathtt{win}(\vec{a}\ j\ \vec{x})$, which is de ned to be $\omega_c(G)$. The players' best randomized algorithm against the worst input is $\min \max_a \mathbf{E}_{x2pi}\, \mathtt{win}(\vec{a}\ j\ \vec{x})$, which is $\min \omega_c(G)$.

This gives us a bound on the worst-case win probability under classical conditions: $\omega_c(G) \quad \min \omega_c(G)$. Suppose π is the probability distribution that achieves this minimum; then $\omega_q(G) \quad \omega_q(G)$, because xing π can only allow the players to achieve a better win probability than their worst-case $\omega_q(G)$. □

5 Conclusion

We have demonstrated Yao's principle to give lower bounds on the competitiveness of algorithms for a couple of online games, which is an upper bound on their e ectiveness. When these bounds match the competitiveness of a particular algorithm, the principle can be used to show the tightness of the algorithm.

There is a dual between using an algorithm to demonstrate a lower bound on the solvability of a problem (alternatively, an upper bound on the hardness of a problem), and using a series of inputs to demonstrate an upper bound on the solvability of a problem (alternatively, an lower bound on the hardness). As more work is done on a particular problem, the gap between the two shrink until a provably optimal algorithm is found.

Online problems, such as the removable online knapsack problem and the and the online coloring co-interval graphs problem, all deal with situations of *partial information*: Quantum non-local games are also situations of partial information: the players must form optimal strategies despite receiving input at random from an adversary. While Yao's principle is applicable to many types of costs when analyzing problems, including running time, as we

saw here it can also be applied to analyzing the competitiveness of algorithms in partial information scenarios.

References

[1] Andris Ambainis, Arturs Backurs, Kaspars Balodis, Agnis Skuskovniks, Juris Smotrovs, and Madars Virza. Worst case analysis of non-local games. In Peter van Emde Boas, Frans C.A. Groen, Giuseppe F. Italiano, Jerzy Nawrocki, and Harald Sack, editors, *SOFSEM 2013: Theory and Practice of Computer Science*, volume 7741 of *Lecture Notes in Computer Science*, pages 121{132. Springer Berlin Heidelberg, 2013.

[2] Alain Aspect, Philippe Grangier, and Gerard Roger. Experimental tests of realistic local theories via bell's theorem. *Phys. Rev. Lett.*, 47:460{463, Aug 1981.

[3] John F. Clauser, Michael A. Horne, Abner Shimony, and Richard A. Holt. Proposed experiment to test local hidden-variable theories. *Phys. Rev. Lett.*, 23:880{884, Oct 1969.

[4] A. Gyrfs and J. Lehel. On-line and rst t colorings of graphs. *Journal of Graph Theory*, 12(2):217{227, 1988.

[5] Xin Han, Yasushi Kawase, and Kazuhisa Makino. Randomized algorithms for removable online knapsack problems. In *Frontiers in Algorithmics* and *Algorithmic Aspects in Information and Management, Third Joint International Conference, FAW-AAIM 2013, Dalian, China, June 26-28, 2013. Proceedings*, pages 60{71, 2013.

[6] Kazuo Iwama and Guochuan Zhang. Optimal resource augmentations for online knapsack. In *Proceedings of the 10th International Workshop on Approximation and the 11th International Workshop on Randomization, and Combinatorial Optimization. Algorithms and Techniques*, APPROX '07/RANDOM '07, pages 180{188, Berlin, Heidelberg, 2007. Springer-Verlag.

[7] Anna R. Karlin, Claire Kenyon, and Dana Randall. Dynamic tcp acknowledgement and other stories about e/(e-1). In *Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing*, STOC '01, pages 502{509, New York, NY, USA, 2001. ACM.

[8] H.A. Kierstead and Jun Qin. Coloring interval graphs with rst- t. *Discrete Mathematics*, 144(1{3):47 { 57, 1995.

[9] Andrew Chi-Chin Yao. Probabilistic computations: Toward a uni ed measure of complexity. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, SFCS '77, pages 222{227, Washington, DC, USA, 1977. IEEE Computer Society.

[10] Hamid Zarrabi-Zadeh. Online coloring co-interval graphs. *Scientia Iranica*, 16(1):1{7, 2009.