

Efficient Algorithm with No-Regret Bound for Sleeping Expert Problem

by

Junhao Lin

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2025

© Junhao Lin 2025

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

The sleeping experts problem is a variant of decision-theoretic online learning (DTOL) where the set of available experts may change over time. In this thesis, we study a special case of the sleeping experts problem with constraints on how the set of available experts can change. The benchmark we use is ranking regret, which is a common benchmark used in sleeping experts problem. Previous research shows that achieving sublinear ranking regret bound in the general sleeping experts problem is NP-hard, so we relax the sleeping experts problem by imposing constraints on how the set of available experts may change. Under those constraints, we present an efficient algorithm which achieves a sublinear ranking regret bound.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor, Professor Ian Munro, for his invaluable guidance, continuous support, and patience throughout my research. His insightful feedback and encouragement were instrumental in shaping this thesis.

Secondly, I am also thankful to my undergraduate supervisor, Professor Nishant Metha at the University of Victoria, who introduced me to the field of decision-theoretic online learning.

Next, I am very grateful to Hamid Shayestehmanesh, who is a former master student at the University of Victoria and the author of the paper “Dying experts: Efficient algorithms with optimal regret bounds” [9]. He gave me a lot of inspiration when I worked on the thesis.

Finally, I would thank the readers Prof. Grant Weddell and Prof. Gautam Kamath for their constructive feedback.

Table of Contents

Author's Declaration	ii
Abstract	iii
Acknowledgements	iv
List of Figures	vii
1 Introduction	1
1.1 Expert Advice Problem	1
1.2 Hedge Algorithm	2
1.3 Sleeping Expert Problem	2
1.4 Ranking Regret	2
1.5 Hardness of the Sleeping Expert Problem	3
2 Model	5
2.1 Inspiration from the dying expert setting	5
2.2 Definition of $\Pi_{i,j}$	8
2.2.1 Definition of $\Pi_{i,j}(j > i)$	8
2.2.2 Definition of $\Pi_{i,j}(j < i)$	9
2.2.3 Definition of $\Pi_{i,j}(j = i)$	9
2.3 Graph representation of the model	9

2.3.1	Columns (or Stages)	9
2.3.2	Blocks (or Sub-groups)	10
2.3.3	Edges	11
2.3.4	Edge Weight $W_{(a,b) \rightarrow (c,d)}^{s_1 \rightarrow s_2}$	12
2.4	An example with three experts	12
2.5	Extension of the model	14
3	Algorithm	15
3.1	Wake up in original order and go to sleep in reverse order	16
3.2	Go to sleep in original order and wake up in reverse order	22
3.3	Observation	28
3.4	Algorithm Pseudo Code	29
3.5	Space and Runtime	29
3.6	Trick to Extend the Algorithm	32
4	Conclusion and Open Problem	33
	References	35

List of Figures

2.1	K groups of permutations	6
2.2	Weight redistribution	7
2.3	Expert 2 goes to sleep again	8
2.4	Wake up in original order	10
2.5	Go to sleep in reverse order	11
2.6	Full Model	14
3.1	Paths of permutation in $\Pi_{r,r}^{s+1}, \Pi_{r,r+1}^{s+1}, \dots, \Pi_{r,K}^{s+1}$	19
3.2	Paths of permutation in $\Pi_{r,k}^s$	21
3.3	Paths of permutation in $\Pi_{r,1}^s, \Pi_{r,2}^s, \dots, \Pi_{r,r}^s$	25
3.4	Paths of permutation in $\Pi_{k,r}^s$	27

Chapter 1

Introduction

1.1 Expert Advice Problem

The Expert Advice Problem, also known as prediction with expert advice, is a variant of Decision-theoretic online learning (DTOL)[2][8][10][11]. It is a sequential game, and there are two types of players in the game: the learner and experts. In each round, the learner can play a distribution over the experts. At the end of each round, the learner receives feedback for each expert, which is called **loss**¹. **The loss is a number between 0 and 1**, and the smaller the loss an expert receives, the better the expert has performed in that round. Suppose that there are K experts and T rounds in total. If the probability the learner plays for expert i in round t is $p_{i,t}$, and the loss for expert i in round t is $l_{i,t}$, then the total loss the learner suffers in round t is $\sum_{j=1}^K p_{i,t} * l_{i,t}$. The best expert in this game refers to the expert with the lowest cumulative loss: $\operatorname{argmin}_{j \in \{1,2,\dots,K\}} \sum_{t=1}^T l_{j,t}$. **Regret refers to be the gap between the cumulative loss of the learner and the cumulative loss of the best expert:** $\sum_{t=1}^T \sum_{i=1}^K p_{i,t} * l_{i,t} - \min_{j \in \{1,2,\dots,K\}} \sum_{t=1}^T l_{j,t}$. In this game, our goal is to find a strategy to compute the probability or weight given the feedback (or loss) for each expert in each round such that the regret is minimized.

¹The term “loss” in decision-theoretical online learning comes from classical decision theory and statistical learning theory, where it denotes the cost or penalty for making an incorrect or suboptimal decision. The terminology has a long-standing foundation in probability, statistics, and game theory.

1.2 Hedge Algorithm

In the Expert Advice Problem, the set of experts is fixed throughout the game, and there is already an efficient algorithm called Hedge² which can achieve a no-regret bound for the normal setting[3]. The Hedge Algorithm achieves a regret bounded by $O(\sqrt{T \log K})$ which grows sub-linearly with T . Therefore, the average regret in each round is $\sqrt{\frac{\log K}{T}}$, and $\lim_{T \rightarrow \infty} \sqrt{\frac{\log K}{T}} = 0$, which we call a no-regret bound. The contribution of the Hedge Algorithm is to determine how to update the probability for each experts for the next round depending on the loss that each expert receives. The Hedge Algorithm maintains K weights for the K experts in each round: $w_{1,t}, w_{2,t}, \dots, w_{K,t}$ and the probability for each expert is obtained by calculating the normalized weight: $p_{i,t} = \frac{w_{i,t}}{\sum_{j=1}^K w_{j,t}}$. When the learner receives the loss for each expert, the Hedge Algorithm will update the weight for each expert depending on the loss: $w_{i,t+1} = w_{i,t} \cdot f(l_{i,t}) = w_{i,t} \cdot e^{-\eta l_{i,t}}$ where η is some constant.

1.3 Sleeping Expert Problem

However, in reality, the set of experts can change in the game. Sleeping Expert Problem is a special case of the Expert Advice Problem in which the set of available experts can change[4][5][7]. Shayestehmanesh et al.[9] study the dying expert setting in which the experts can be removed from the game. The motivation for studying sleeping experts is that it is a more general setting, and it has wider applications. For example, in stock trading, each stock can be viewed as an expert, and the money we spent on each stock can be viewed as the probability distribution. The loss that a stock receives depends on the rise and fall of stock prices. The set of available stocks might change everyday, which is close to the sleeping expert setting.

1.4 Ranking Regret

In the Expert Advice Problem, we try to minimize the gap between the learner and the best expert throughout the game (which we don't know before we finish the game)[3].

²The name ‘‘Hedge’’ might come from the financial term ‘‘hedging’’, which refers to diversifying investments to reduce risk. In the setting of decision-theoretic online learning, it means reducing the regret in the worst case scenario.

However, we cannot do that in dying experts setting or in sleeping experts setting because the loss for an expert can be undefined in some rounds (an expert can receive loss only when it is awake in that round). Therefore, the **ranking regret**[7] was used in the dying expert setting[9]. There are $K!$ permutations of the K experts, and each permutation can be viewed as a special expert which behaves as the first awake expert in the permutation. In this case, each permutation can receive a valid loss in each round. For example, suppose that there are 3 experts, the permutation 123 means that when all experts are awake, it behaves the same as expert 1 and receive loss $l_{1,t}$; and when only expert 2 and expert 3 are awake, it behaves the same as expert 2 and receive loss $l_{2,t}$. **The ranking regret refers to the gap between the cumulative loss of the learner and the cumulative loss of the best permutation throughout the game.**

In the thesis, we also use the ranking regret, which is good for the sleeping expert setting. Firstly, if we apply Hedge on those $K!$ permutations, we can still get a no-regret bound because the ranking regret can be bounded by $O(\sqrt{T \log K!}) = O(\sqrt{TK \log K})$ and the average regret in each round is $O(\sqrt{\frac{K \log K}{T}})$ which still approaches zero when T goes to infinity. Secondly, the number of experts (the permutations) is independent of the number of rounds in the game. That is, no matter how many rounds in the game, we only need to compete the learner with those $K!$ permutations. However, if we keep track of the weights of $K!$ permutations and update them at each round, the runtime will be too large when K is large.

1.5 Hardness of the Sleeping Expert Problem

To achieve sublinear ranking regret bound for the general sleeping expert setting is believed to at least as hard as PAC-learning DNFs, which is NP-hard[6]. Therefore, we have to relax the problem by either aiming for a sub-optimal regret bound or achieving a sublinear ranking regret bound in a restricted setting. Emamjomeh-Zadeh et al.[1] develop efficient algorithms with approximate regret bounds for the general setting. They also mentioned that the research of Shayestehmanesh et al.[9] was the only one that worked on a special case of the sleeping experts setting. In our thesis, we want to achieve a sublinear ranking regret bound in a more general setting than the dying expert setting[9]: At the beginning of the game, the experts wake up one by one, and we number the experts according to the order they wake up. We denote this order as original order $(1, 2, \dots, K)$. After they all wake up, they can go to sleep in original order $(1, 2, \dots, K)$ or in the reverse order $(K, K - 1, \dots, 1)$. After they all go to sleep, they can wake up again in either original order or reverse order. The experts can wake up and go to sleep for many times, but

they must follow either original order or reverse order. Although this setting has many constraints comparing with the general setting, it is much more general than the previous research on dying expert setting. In next chapter, we will introduce a model together with an efficient algorithm which is able to achieve a sublinear ranking regret bound in this setting.

Chapter 2

Model

In the previous section, we explain the setting we want to solve. In this setting, the set of experts can change, but only within specific constraints. Recall that the Hedge Algorithm tells us how to update the weight of a permutation using the feedback of the leading expert. However, we do not need to worry about how to update the weights after we receive feedback for the experts. That is, we can assume that for the current round, for every awake expert i , we are already given the factor $f(i)$, and we will update the weight of the permutation with i be the leading expert by multiplying the weight by the factor $f(i)$. Note that the runtime of the Hedge Algorithm is proportional to the number of experts in the game. Clearly, we do not want to keep track of the weights for all $K!$ permutations and update them one by one. Before we introduce our model, we first explain the inspiration we get from the dying expert setting, and how we use that to design our model and algorithm.

2.1 Inspiration from the dying expert setting

Although there are $K!$ permutations, in each round we play at most K experts because each permutation behaves as one of the awake expert and the weight for each expert is the total weights of permutations with that expert as the leading expert. So we just need to determine the total weights of permutations with experts $1, 2, \dots, K$ as the leading expert. There are at most K outcomes and thus at most K different ways we can update the weights of those permutations in each round because how we update the weight of a permutation only depends on which is the leading expert in the permutation. Therefore, we want to divide the permutations into K groups, group i contains the permutations with i be the leading expert at current round. In this way, the weights of the permutations in the

same group will be multiplied by the same factor at the end of each round. We maintain the total weights of permutations in each group: $w_1, w_2, \dots, w_K$. Since all permutation weights in one group are multiplied by the same factor, we can update those K weights for the next round easily: $w_{i,\text{next}} = w_{i,\text{prev}} \cdot f(i)$. At the same time, once we figure out the weights for each group, the probability for playing expert i is $\frac{w_i}{\sum_{j=1}^K w_j}$, which is equivalent to playing a probability distribution over $K!$ permutations according to their weights. Shayestehmanesh et al [9] use this idea to develop a linear-time algorithm. In the dying experts setting, instead of keeping track of the weights for $K!$ permutations, they only keep track of K groups of permutations (each group contains all permutations behaving as the same expert at the current round) with the K weights mentioned above: $w_1, w_2, \dots, w_K$. When an expert i goes to sleep, w_i will become 0 in the next round, and the weight will move to the non-empty groups. As shown in Figure 2.1, when $K = 4$, there are 24 permutations in total. We want to divide those permutations into 4 groups depending on which is the first available expert, and we use group 1, 2, 3, 4 to denote those groups. If all experts are available, each group contains exactly 6 permutations. When expert 2 goes to sleep, group 2 will become empty since expert 2 will no longer be the first available expert in those permutations. Those permutations will move to group 1, group 3, and group 4. Notice that same number of permutations are moving into those 3 groups because expert 1, 3 and 4 have equivalent chance to become the next leading available expert. We also use different colors for the permutations in the same group, indicating that the weights of the permutations with different colors might be different. Similarly, when expert 3 goes to sleep, the permutations in group 3 will move to group 1 and group 4. For both black permutations and blue permutations, same number of those permutations will move to group 1 and group 4.

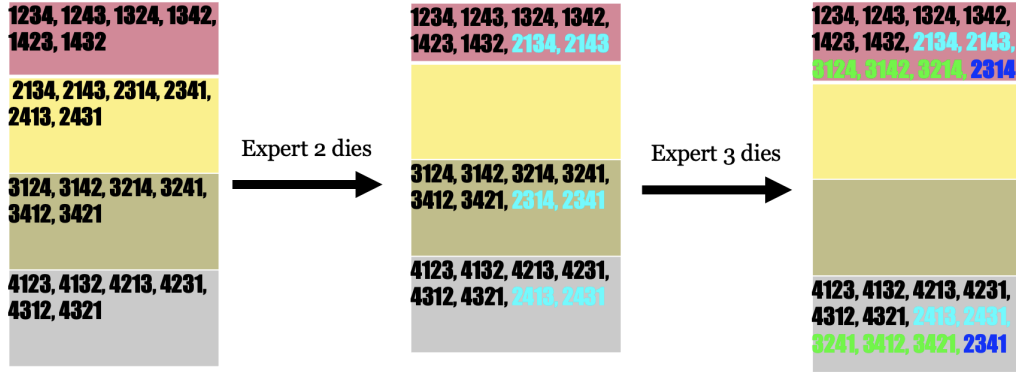


Figure 2.1: K groups of permutations

As a result, as shown in Figure 2.2, when expert 2 goes to sleep, the weights of the permutations in group 2 will spread evenly into group 1, 3 and 4. Also, when expert 3 goes to sleep, the weights of the permutations in group 3 will spread evenly into group 1 and 4. Therefore, it is feasible to only keep track of the total weights of the permutations in those K groups in the dying expert setting.

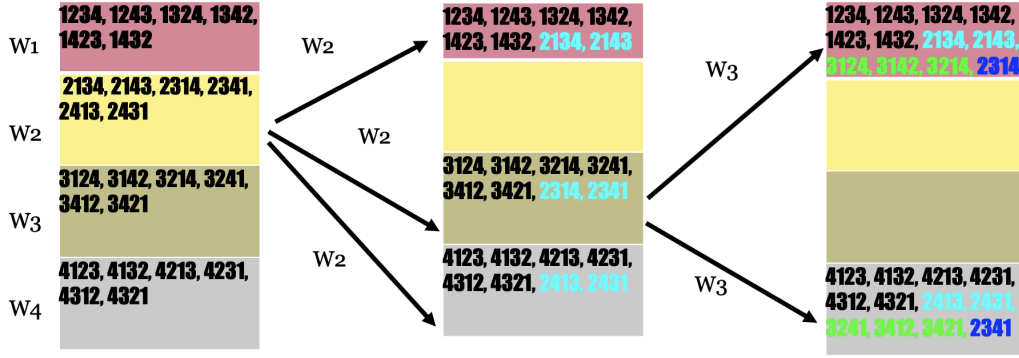


Figure 2.2: Weight redistribution

However, the general setting is much more complex than the dying expert setting, and we cannot only keep track of track of K groups of permutations. As shown in Figure 2.3, when expert 2 goes to sleep again, although the number of permutations moving into group 1, 3, and 4 is still the same, the weights of those permutations might be different (we use different colors to indicate that). Therefore, if we only keep track of the total weight of group 2, we cannot determine the weights of the permutations moving to group 1, 3, 4 respectively. However, if we further divide the permutations in group 2 into more subgroups (according to the color) and keep track of the weights of those subgroups, then we can determine the weights of permutations moving to group 1, 3, 4 respectively.

In our model, we further divide each of those K groups into K disjoint subgroups. Let $\Pi_1, \Pi_2, \dots, \Pi_K$ denote the K groups, recall that Π_i contains all the permutations with expert i be the leading expert among the awake experts at current round. If we use $W(\Pi)$ to denote the total weights of permutations in group Π , then $w_i = W(\Pi_i)$. Now, a group Π_i will be divided into K disjoint subgroups $\Pi_{i,1}, \Pi_{i,2}, \dots, \Pi_{i,K}$, we will give the definition for $\Pi_{i,j}$ below. In this case, $w_i = \sum_{j=1}^K W(\Pi_{i,j})$. As a result, we want to keep track of K^2 subgroups of permutations with K^2 weights $W_{i,j}$, and our goal is to use the weights in the previous round to update the weights for the next round.

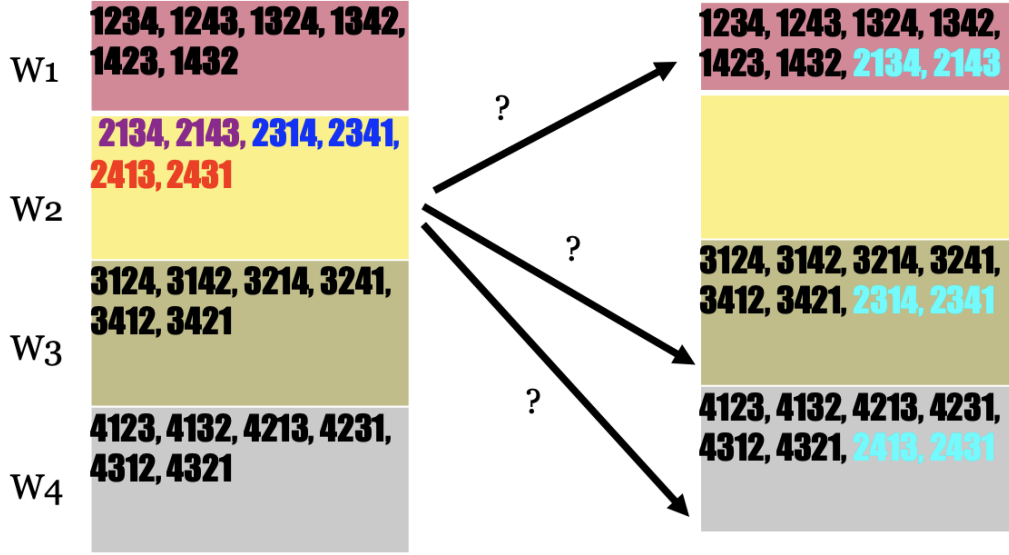


Figure 2.3: Expert 2 goes to sleep again

2.2 Definition of $\Pi_{i,j}$

in this section, we will give the definition of the K^2 sub-groups $\Pi_{i,j}$ mentioned above. The first subscript i indicates that it contains the permutations in which expert i is the leading expert among the awake experts, so $\Pi_{i,j}$ is empty when i is not awake. The second subscript j indicates the relationship between expert i and expert j in the permutations in the subgroup. Note that we have different definitions when $j > i$, $j < i$ and $j = i$. The subgroups $\Pi_{i,j}(j > i)$ are only used when the experts wake up in original order and go to sleep in reverse order while the subgroups $\Pi_{i,j}(j < i)$ are only used when the experts go to sleep in original order and wake up in reverse order.

2.2.1 Definition of $\Pi_{i,j}(j > i)$

$\Pi_{i,j}(j > i)$ is a set of permutations that satisfies:

1. i is the leading expert among the awake experts at current round
2. j is the smallest number expert preceding expert i .

2.2.2 Definition of $\Pi_{i,j}(j < i)$

$\Pi_{i,j}(j < i)$ is a set of permutations that satisfies:

1. i is the leading expert among the awake experts at current round
2. j is the largest number expert preceding expert i

2.2.3 Definition of $\Pi_{i,j}(j = i)$

$\Pi_{i,j}(j = i)$ is a set of permutations that satisfies:

1. i is the leading expert among the awake experts at current round
2. i is the leading expert among all experts

2.3 Graph representation of the model

Our model is three dimensional. The first dimension is the stage (or set of awake experts); the second dimension is the K groups of permutations which contains the permutations with expert 1 to expert K be the leading expert respectively in some stage. The last dimension is the K subgroups of each group of permutations. The model can be represented using a graph, and we expand the last two dimensions of the model so that the first dimension is stage and the second dimension is the K^2 subgroups. Figure 2.4 and Figure 2.5 show the model which can be used to compute the weights of each subgroup when the experts wake up in original order and go to sleep in reverse order and when the number of experts $K = 3$. The graph contains $K + 1 = 4$ columns, one column represents a stage. Each column consists of $K^2 = 9$ blocks standing for 9 sub-groups of permutations. There are edges connecting the blocks in two adjacent columns. We will explain those elements in details in the following sections.

2.3.1 Columns (or Stages)

Each column in the model represents a stage which is the set of awake experts. There are $(K + 1)$ different stages. The game starts at stage 1 where the set of awake expert is an

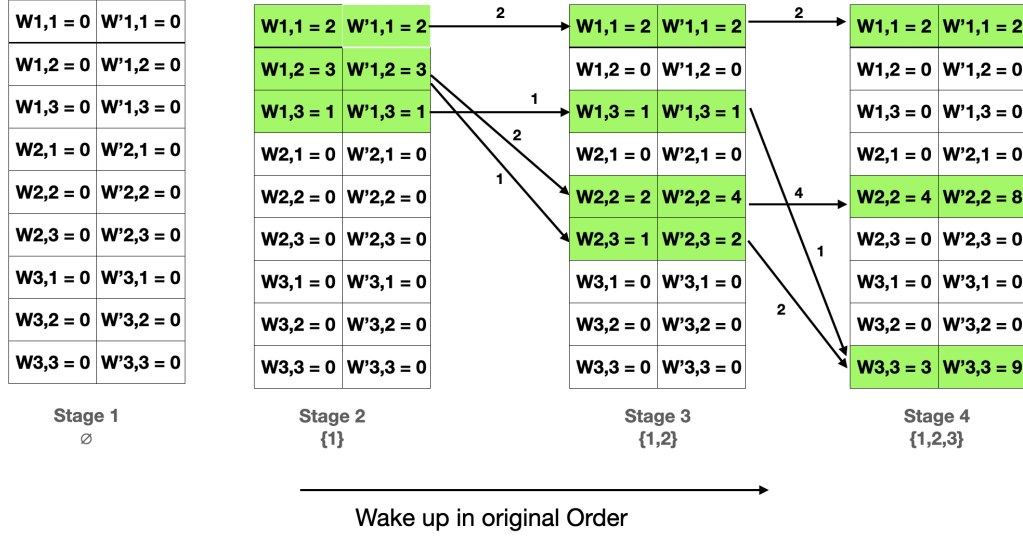


Figure 2.4: Wake up in original order

empty set $\emptyset$, and when the experts wake up one by one in original order, the set of awake experts (or stage) will change as follows: $\emptyset \rightarrow \{1\} \rightarrow \{1, 2\} \rightarrow \dots \rightarrow \{1, 2, \dots, K\}$. That is, we move from left to right in the model. If the experts go to sleep in reverse order after that, we will move from right to left in the first part. When an expert wakes up or goes to sleep, we move from one stage to another adjacent stage, that is, we move from one column to another adjacent column. When we move to a new stage, the permutations in one subgroup might move to other subgroups. For example, when expert i goes to sleep, subgroups $\Pi_{i,1}, \Pi_{i,2}, \dots, \Pi_{i,K}$ will be empty, and if a permutation belongs to one of those subgroups before, it will move to another subgroup. Therefore, we need to compute the weights of permutations moving from one subgroup to another subgroup, and then compute the weights of the permutations in each subgroup in the new stage. After that, each awake expert will receive a feedback and we need to update the weights of each subgroups depending on the feedback of each expert (we just need to multiply the weight of a subgroup by some factor which is given to us as mentioned before).

2.3.2 Blocks (or Sub-groups)

Each column contains K^2 blocks which stand for the K^2 subgroups of permutations mentioned before. Each block represents a subgroup $\Pi_{i,j}$, and we have two numbers $W_{i,j}$ and

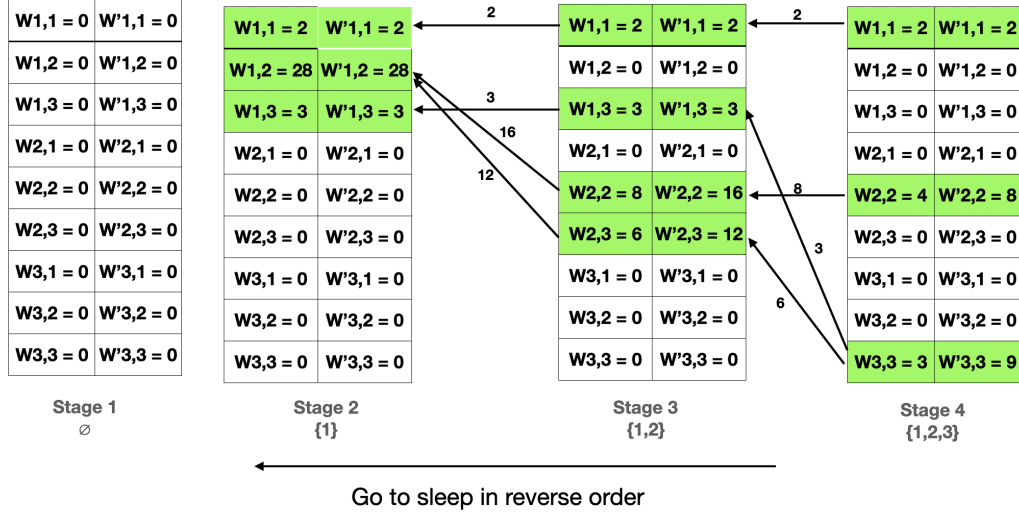


Figure 2.5: Go to sleep in reverse order

$W'_{i,j}$ for each subgroup. $W_{i,j}$ represents the total weights of permutations in subgroup $\Pi_{i,j}$ before we receive feedback for the awake experts, and $W'_{i,j}$ is the updated weight of the subgroup after we receive feedback. Therefore, $W'_{i,j} = W_{i,j} \cdot f(i)$ where $f(i)$ is given to us. Note that each block in a column always contains the same set of permutations (given the set of awake experts, we can compute which subgroup a permutation belongs to), but the weight of the permutation can change.

2.3.3 Edges

Each edge connects two blocks in two adjacent columns indicating the permutations moving from one block (subgroup) to another block when stage changes. Each edge has a weight representing the weights of the permutations moving from one subgroup to the other subgroup when stage changes. Therefore, the edges have directions. At the beginning, there is no edge in the graph, and we start at stage 1. When we are at some stage s , we can either move forward to stage $(s + 1)$ or move back to stage $(s - 1)$. When we move from one stage to its adjacent stage, we can use the weights $W'_{i,j}$ in the previous stage together with the weights of the edges (if there are any) between the two stages to compute the weights $W_{i,j}$ in the next stage. At the same time, we also build or update the edges between the two stages. Those edges will be used in future computation.

2.3.4 Edge Weight $W_{(a,b) \rightarrow (c,d)}^{s_1 \rightarrow s_2}$

We use $W_{(a,b) \rightarrow (c,d)}^{s_1 \rightarrow s_2}$ to denote the weight of the edge which starts at block $\Pi_{a,b}$ in stage s_1 and points to block $\Pi_{c,d}$ in stage s_2 . We use $W_{(c,d)}^{s_1 \rightarrow s_2}$ to represent the total weights of the edges reaching block $\Pi_{c,d}$ in stage s_2 .

In each round, one specific expert wakes up or goes to sleep, and we move from one stage to another adjacent stage, that is, we move from one column to another adjacent column. Depending on the order the experts wake up or go to sleep, we know the current stage and the previous stage. We want to use the weights $W'_{i,j}$ in the previous stage together with weights of edges between the two stages to update the weights $W_{i,j}$ in current stage. I will use an example to explain this process.

2.4 An example with three experts

When there are only three experts, there are 6 permutations: 123, 132, 213, 231, 312, 321. Assume that the initial weights for each permutation is 1. When we move to a new stage, suppose that the weights of the permutations with expert i being the current leading expert will always be multiplied by i . That is, $W'_{1,j} = W_{1,j} (1 \leq j \leq 3)$, $W'_{2,j} = 2 \cdot W_{2,j} (1 \leq j \leq 3)$, and $W'_{3,j} = 3 \cdot W_{3,j} (1 \leq j \leq 3)$. Figure 2.4 shows the weights of subgroups and edges in the model when the experts wake up in original order. The non-empty subgroups in each stage are highlighted. We start at stage 1 with no expert awake. There are four columns representing four stages. Each column contains 9 blocks representing the 9 sub-groups of permutations. In stage 1, since no expert is awake, all sub-groups are empty, and the weights are all 0.

When the experts wake up in original order, we move from left to right until we reach stage 4. In stage 1, expert 1 is the only awake expert. According to the definition in section 2.2.3, $\Pi_{1,1}$ contains the permutations with expert 1 as the leading expert among all experts, which are 123 and 132, and thus $W_{1,1} = 2$. Similarly, $\Pi_{1,2}$ contains the permutations with expert 2 as the lowest number expert preceding 1, which are 213, 231 and 321, and $W_{1,2} = 3$; $\Pi_{1,3}$ contains only the permutation 312, and $W_{1,3} = 1$. Then, the weights of all permutations will be multiplied by 1 (remain unchanged) since expert 1 is the current leading expert. When expert 2 wakes up, we move from stage 2 to stage 3. Permutations 213 and 231 will move from $\Pi_{1,2}$ to $\Pi_{2,2}$ because expert 2 will become the leading expert among the awake experts (1 and 2) in those permutations, and expert 2 is the leading expert among all experts. Similarly, permutation 321 will move from $\Pi_{1,2}$ to $\Pi_{2,3}$ because expert 2 will become the leading expert among the awake experts in the permutations, and expert 3

is the lowest number expert preceding expert 2. Therefore, in stage 3, $\Pi_{2,2} = 2$ and $\Pi_{2,3} = 1$. Also, we create two edges, one from $\Pi_{1,2}$ to $\Pi_{2,2}$ and the other from $\Pi_{1,2}$ to $\Pi_{2,3}$ with weights 2 and 1 respectively. For the permutations in $\Pi_{1,1}$ and $\Pi_{1,3}$, they will stay in the same subgroups. Therefore, in stage 3, $\Pi_{1,1} = 2$ and $\Pi_{1,3} = 1$ and we also create two edges, one from $\Pi_{1,1}$ to $\Pi_{1,1}$ and the other from $\Pi_{1,3}$ to $\Pi_{1,3}$, with weights 2 and 1 respectively. After that, the weights of permutations 123, 132 and 312 will remain unchanged while the weights of permutations 213, 231 and 321 will be multiplied by 2. Therefore, $W'_{1,1} = W_{1,1} = 2$, $W'_{1,3} = W_{1,3} = 1$, $W'_{2,2} = 2 \cdot W_{2,2} = 4$ and $W'_{2,3} = 2 \cdot W_{2,3} = 2$. Similarly, we update the weights in stage 4 and create edges between stage 3 and stage 4. After we reach stage 4, all experts are awake.

If the experts go to sleep in reverse order, we move from stage 4 back to stage 1 (as shown in Figure 2.5). When expert 3 goes to sleep, Permutations 312 will move from $\Pi_{3,3}$ to $\Pi_{1,3}$, note that these are the permutations moving from $\Pi_{1,3}$ to $\Pi_{3,3}$ when we move from stage 3 to stage 4. Permutation 321 will move from $\Pi_{3,3}$ to $\Pi_{2,3}$, it is the permutation moving from $\Pi_{2,3}$ to $\Pi_{3,3}$ when we move from stage 3 to stage 4. Therefore, if a permutation follows some edge when moving from stage s to stage $(s+1)$, it must follow the same edge back when moving from stage $(s+1)$ to stage s . This is not hard to understand because if a permutation follows an edge from subgroup g_1 to subgroup g_2 , it indicates that the permutation belongs to subgroup g_1 in stage s and belongs to subgroup g_2 in stage $(s+1)$, so when moving from stage $(s+1)$ to stage s , this permutation must move from subgroup g_2 to subgroup g_1 , which is the reverse direction. Note that the weights of the permutations in $\Pi_{3,3}$ in stage 4 have different weights. The weight of permutation 312 is 3 and the weight of permutation 321 is 6. Therefore, we update two edges, the one from $\Pi_{3,3}$ to $\Pi_{1,3}$ and the other from $\Pi_{3,3}$ to $\Pi_{2,3}$, with weights 3 and 6 respectively. Also, in stage 3, $W_{1,3}$ and $W_{2,3}$ are updated to 3 and 6 respectively. Next, the weight of permutation 312 will remain unchanged and the weight of permutation 321 will double, and $W'_{1,3} = W_{1,3} = 3$, $W'_{2,3} = 2 \cdot W_{2,3} = 12$. Similarly, we update the weights in stage 2 and update the edge weights. Finally when we go back to stage 1 again, all experts are not awake and all subgroups are empty. The next time when the experts wake up in original order, when expert 1 wakes up, we move to stage 2 and we do not need to update the weights $W_{i,j}$ in stage 2. $W'_{i,j}$ in stage 2 also remains unchanged since expert 1 is the current leading expert of all permutations. When we move from stage 2 to stage 4 again, the model already had edges, and we need to update the weights of the edges and the weights in the next stage.

2.5 Extension of the model

The model presented in the previous section can only compute the weights of the subgroups when the experts wake up in original order or go to sleep in reverse order. However, we can extend the model to compute the weights of the subgroups when the experts go to sleep in original order or wake up in reverse order. This can be done easily by simply adding more stages to the model. The new model now contains $(2K + 1)$ stages. As shown in Figure 2.6, when $K = 3$, we have $2K + 1 = 7$ stages (note that we have two different stages when the set of awake experts is empty). When moving from stage 1 to stage 4, the experts wake up in original order; when moving from stage 4 to stage 1, the experts go to sleep in reverse order; when moving from stage 4 to stage 7, the experts go to sleep in original order; when moving from stage 7 to stage 4, the experts wake up in reverse order. Although there can be many rounds in the game, we always move between stages within the model.

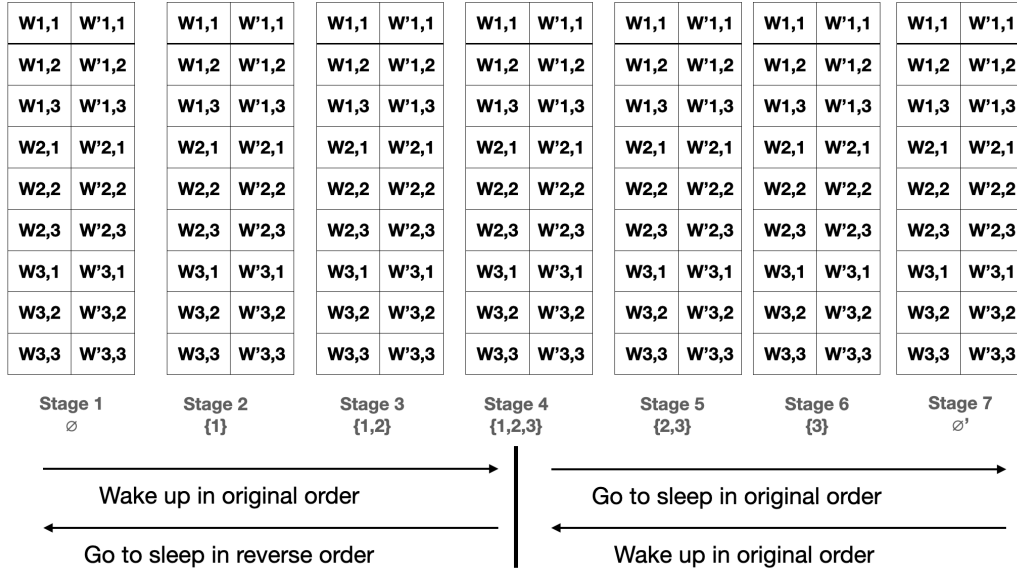


Figure 2.6: Full Model

Chapter 3

Algorithm

In this section, we will explain how to update the weights of permutations in each subgroup when the stage changes before we receive feedback for the experts and the weight of each permutation does not change. That is, the weight change of subgroups due to permutations moving from one subgroup to another when the stage changes. To be more specific, when we move from stage s_1 to stage s_2 , we want to determine how to use the weights $W'_{i,j}$ in stage s_1 together with the weights of the edges (if there are any) to update the weights $W_{i,j}$ in stage s_2 . We assume that the initial weight for each permutation is $\frac{1}{K!}$ so the sum of the weights of all permutations will be 1 at the beginning. In the first section, we will explain how to update the weights and edges when the experts wake up in original order and go to sleep in reverse order, which is moving from left to right and moving from right to left within the first part of the model. In the second section, we will explain how to update the weights and edges when the experts go to sleep in original order and wake up in reverse order, which is moving from left to right and moving from right to left within the second part of the model. In the last section, we will explain a trick to update the weights and edges when the experts wake up in reverse order after they go to sleep in reverse order or when the experts wake up in original order after they go to sleep in original order. We have the following notations for permutations and weights as we change stage. When moving from stage s_1 to stage s_2 , $\Pi_{i,j}^{s_1}$ denotes the subgroup of permutations $\Pi_{i,j}$ in stage s_1 and $\Pi_{i,j}^{s_2}$ denotes the subgroup of permutations $\Pi_{i,j}$ in stage s_2 . $W_{i,j}^{s_1'}$ denotes the weight $W'_{i,j}$ in stage s_1 and $W_{i,j}^{s_2}$ denotes the weight $W_{i,j}$ in stage s_2 .

3.1 Wake up in original order and go to sleep in reverse order

First we must determine which subgroups will be non-empty during this period.

Lemma 3.1.1. *When the experts wake up in original order or when they go to sleep in reverse order, for any awake expert r , $\Pi_{r,k}(k < r)$ is empty and $\Pi_{r,r}$ is not empty. For any $k > r$, if k is awake, then $\Pi_{r,k}$ is empty. Otherwise, $\Pi_{r,k}$ is not empty.*

Proof. When the experts wake up in original order or when they go to sleep in reverse order, if an expert r is awake, we know that for any expert $k < r$, it is also awake. According to definition of $\Pi_{r,k}(k < r)$, expert k is preceding expert r . Therefore, r cannot be the leading expert among the awake experts, and $\Pi_{r,k}(k < r) = \emptyset$. $\Pi_{r,r}$ is not empty because r is the leading expert among all experts, and r must be the leading expert among the awake experts. As for $\Pi_{r,k}(k > r)$, if k is awake, according to the definition of $\Pi_{r,k}(k > r)$, k is preceding r , so r cannot be the leading expert among the awake experts, and $\Pi_{r,k} = \emptyset$. On the other hand, if k is not awake, then any expert $j \geq k$ is not awake. According to the definition, we know that j is the lowest number expert preceding expert r , so r must be the leading expert among all awake experts because any expert larger than k is also not awake, and $\Pi_{r,k}$ is not empty. □

Next we explain how to update the weights of all subgroups in the next stage and the weights of edges when the stage changes. Recall that when the experts wake up in original order, we are moving from left to right in the first part of the model. When the stage changes from s to $(s + 1)$, we use the weights in the stage s to update the weights in stage $(s + 1)$. However, if we move from stage s to stage $(s + 1)$ for the first time, there are no edges between stage s and stage $(s + 1)$. Otherwise, we also have to use the edge weights in order to update the weights of all subgroups in the next stage. Theorem 3.1.3 and Theorem 3.1.5 explain how we deal with these two cases separately. On the other hand, when the experts go to sleep in reverse order, we are moving from right to left in the first part of the model. When stage changes from $(s + 1)$ to s , since we have reached stage $(s + 1)$ before, there must be edges between stage s and stage $(s + 1)$, and we also need to use the weights of those edges to update the weights of all subgroups in stage s . Theorem 3.1.4 explains how to deal with this case. Also, when expert 1 wakes up for the first time, Lemma 3.1.2 explains how to compute the initial weights at stage 1.

Lemma 3.1.2. *At stage 1, when expert 1 wakes up for the first time, $W_{1,1} = \frac{1}{K}$ and $W_{1,k} = \frac{1}{(k-1)k}$ ($1 < k \leq K$)*

Proof. $\Pi_{1,1}$ is the set of permutations with expert 1 as the leading expert among all experts, and $(K-1)!$ permutations satisfy this condition, and therefore $W_{1,1} = \frac{(K-1)!}{K!} = \frac{1}{K}$. On the other hand, $\Pi_{1,k}$ contains the permutations with 1 being the leading expert among expert 1 to expert $(k-1)$, there are $(k-2)!$ possibilities. Also, expert k must be inserted in front of expert 1, The ordering of rest of the experts (there are $(K-k)$ of them) does not matter, so the number of possible permutations is $(k-2)! \cdot (k+1) \cdot (k+2) \cdot \dots \cdot K$. Therefore, $W_{1,k} = \frac{(k-2)! \cdot (k+1) \cdot (k+2) \cdot \dots \cdot K}{K!} = \frac{1}{(k-1)k}$

□

Theorem 3.1.3. *When experts wake up in original order, suppose that when expert r wakes up, the stage changes from s to $(s+1)$, and there are no edges between stage s and stage $(s+1)$. For any $(1 \leq i \leq r-1)$, the permutations in the subgroups $\Pi_{i,r}^s$ at stage s will move to subgroups $\Pi_{r,r}^{s+1}, \Pi_{r,r+1}^{s+1}, \dots, \Pi_{r,K}^{s+1}$ at stage $(s+1)$. Also, $W_{(i,r) \rightarrow (r,r)}^{s \rightarrow (s+1)} = \frac{r}{K} \cdot W_{i,r}^s$, and $W_{i,j}^{s+1} = \sum_{i=1}^{r-1} W_{(i,r) \rightarrow (r,r)}^{s \rightarrow (s+1)} = \sum_{i=1}^{r-1} \frac{r}{K} \cdot W_{i,r}^s$. For any $r+1 \leq k \leq K$, $W_{(i,r) \rightarrow (r,k)}^{s \rightarrow (s+1)} = \frac{r}{(k-1)k} \cdot W_{i,r}^s$, and $W_{r,k}^{s+1} = \sum_{i=1}^{r-1} W_{(i,r) \rightarrow (r,k)}^{s \rightarrow (s+1)} = \sum_{i=1}^{r-1} \frac{r}{(k-1)k} \cdot W_{i,r}^s$. The rest of the subgroups will remain unchanged. Let $\Pi_{x,y}$ be one of them, then $W_{x,y}^{s+1} = W_{x,y}^s$ and $W_{(x,y) \rightarrow (x,y)}^{s \rightarrow (s+1)} = W_{x,y}^s$*

Proof. When the expert r wakes up, some permutations with $i < r$ being the previous leading expert will change to having r being the leading expert at stage $(s+1)$. if we consider subgroups of permutations $\Pi_{i,j}^s$ in the previous stage which are not empty, only the permutations in $\Pi_{i,r}^s$ will change to having r being the leading expert in stage $(s+1)$ because:

- (1) for the permutations in $\Pi_{i,j}^s$ with $j > r$, j is the lowest number expert preceding i , so i is still the leading expert at stage $(s+1)$, and those permutations will move to $\Pi_{i,j}^{s+1}$.
- (2) for the permutations in $\Pi_{i,i}^s$, expert i is still the leading expert among awake experts in stage $s+1$, and $\Pi_{i,i}^{s+1} = \Pi_{i,i}^s$.
- (3) for $j < r$, $\Pi_{i,j}^{s+1} = \Pi_{i,j}^s = \emptyset$ according to Lemma 3.1.1.

Figure 3.1 shows all the possible paths throughout the game for the permutations whose leading expert is r when r wakes up (permutation in $\Pi_{r,r}^{s+1}, \Pi_{r,r+1}^{s+1}, \dots, \Pi_{r,K}^{s+1}$). The path of a permutation refers to the leading expert at each time of a permutation. All paths start at a leaf node on the left. Note that all leaves on the left are labeled 1 because all permutations

have expert 1 be the first leading expert when the experts wake up in original order. The middle node r means that the leading expert becomes expert r when r wakes up. Therefore, a $1 \rightarrow r$ path (a black path) indicates the leading experts of that permutation at each time until r wakes up. Also, if a permutation follows one specific black path when the experts wake up in original order, it will follow the same path but in reverse direction when the experts go to sleep in reverse order. After that, experts $r + 1$ to experts K wake up one by one, and those permutations will follow different red paths. When all experts wake up, the red paths will reach $r, r + 1, \dots, K$. Similarly, if a permutation follows one specific red path when the experts wake up in original order, it will follow the same path but in reverse direction when the experts go to sleep in reverse order. Before the experts go to sleep in reverse order, the experts might go to sleep in original order and then wake up in reverse order multiple times, which is represented by the green paths. For any $r \leq i \leq K$, a $i \rightarrow K$ green path denotes a path of permutation when the experts go to sleep in original order. Similarly, if a permutation follows one specific green path when the experts go to sleep in original order, it will follow the same path but in reverse direction when the experts wake up in reverse order. Also, even if the experts go to sleep in reverse order right after they wake up in original order, we can still assume that the permutations follow the green path back and forth for multiple times but the weights of the permutations remain unchanged.

The subgroup $\Pi_{i,r}^s$ contains permutations following a subset of the black paths (the black path with expert i be the last leading expert before expert r). Similarly, the subgroup $\Pi_{r,k}^{s+1}$ ($r \leq k \leq K$) contains permutations following a subset of the red paths. If there are no edges between stage s and stage $(s + 1)$, then expert r wakes up for the first time, and all permutations following one black path $1 \rightarrow r$ must have the same weight. Also, for the set of permutations following any black path, the fraction of permutations following each red path is a fixed number because a permutation following which black path only depends on the ordering of $1, 2, \dots, r - 1$ in the permutation and a permutation following which red path only depends on the ordering of $r, r + 1, \dots, K$ in the permutation. Since the subgroup $\Pi_{r,k}^{s+1}$ ($r \leq k \leq K$) contains permutations following a subset of the red paths, for the set of permutations following any black path, the fraction of permutations moving into each subgroup $\Pi_{r,k}^{s+1}$ ($r \leq k \leq K$) is also a fixed number. Next, we will calculate this fraction.

for permutations in subgroup $\Pi_{i,r}^s$, i is the leading expert among experts 1 to experts $(r - 1)$, and r is ahead of i . Now, if we consider any expert $k > r$, if r is the leading expert among experts 1 to experts $(k - 1)$ and expert k is ahead of expert r . To insert expert $(r + 1)$ into the permutation, there are $(r + 1)$ positions, and r of them are behind r . Similar for experts $(r + 2)$ to $(k - 1)$. Finally, to insert expert k , there k positions, and only one of them is ahead of r . Therefore, $\frac{r}{r+1} \cdot \frac{r+1}{r+2} \cdot \dots \cdot \frac{r+(k-r)-2}{r+(k-r)-1} \cdot \frac{1}{k} = \frac{r}{(k-1)k}$ of those permutations will

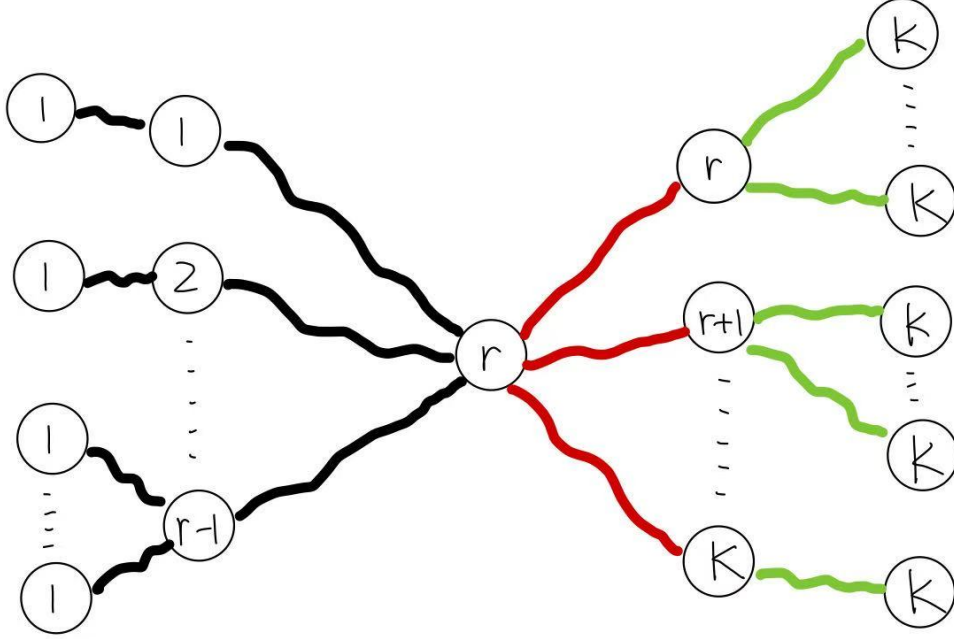


Figure 3.1: Paths of permutation in $\Pi_{r,r}^{s+1}, \Pi_{r,r+1}^{s+1}, \dots, \Pi_{r,K}^{s+1}$

move to subgroup $\Pi_{r,k}^{s+1}$. Therefore, for any $r+1 \leq k \leq K$, $W_{(i,r) \rightarrow (r,k)}^{s \rightarrow s+1} = \frac{r}{(k-1)k} \cdot W_{i,r}^{s'}$, and $W_{r,k}^{s+1} = \sum_{i=1}^{r-1} W_{(i,r) \rightarrow (r,k)}^{s \rightarrow s+1} = \sum_{i=1}^{r-1} \frac{r}{(k-1)k} \cdot W_{i,r}^{s'}$. If r is the leading expert among all experts, for the rest $(K-r)$ experts, we need to insert them all after r . Similar to the previous argument, $\frac{r}{r+1} \cdot \frac{r+1}{r+2} \cdot \dots \cdot \frac{K-1}{K} = \frac{r}{K}$ of those permutations will move to subgroup $\Pi_{r,r}^{s+1}$. Therefore, $W_{(i,r) \rightarrow (r,r)}^{s \rightarrow s+1} = \frac{r}{K} \cdot W_{i,r}^{s'}$, and $W_{i,j}^{s+1} = \sum_{i=1}^{r-1} W_{(i,r) \rightarrow (r,r)}^{s \rightarrow s+1} = \sum_{i=1}^{r-1} \frac{r}{K} \cdot W_{i,r}^{s'}$.

□

Theorem 3.1.4. *When experts go to sleep in reverse order, suppose that when expert r goes to sleep, the stage changes from s to $(s-1)$. For any $r \leq k \leq K$, the permutations in the subgroups $\Pi_{r,k}^s$ at stage s will move to subgroups $\Pi_{1,r}^{s-1}, \Pi_{2,r}^{s-1}, \dots, \Pi_{r-1,r}^{s-1}$ at stage $(s-1)$.*

For any $1 \leq i \leq r-1$ and for any $r \leq k \leq K$, $W_{(r,k) \rightarrow (i,r)}^{s \rightarrow (s-1)} = \frac{W_{(i,r) \rightarrow (r,k)}^{(s-1) \rightarrow s}}{W_{(r,k)}^{(s-1) \rightarrow s}} \cdot W_{r,k}^s$, and $W_{i,r}^{s-1} = \sum_{k=r}^K W_{(r,k) \rightarrow (i,r)}^{s \rightarrow (s-1)} = \sum_{k=r}^K \frac{W_{(i,r) \rightarrow (r,k)}^{(s-1) \rightarrow s}}{W_{(r,k)}^{(s-1) \rightarrow s}} \cdot W_{r,k}^s$. The rest of the subgroups will remain unchanged. Let $\Pi_{x,y}$ be one of them, then $W_{x,y}^{s-1} = W_{x,y}^s$ and $W_{(x,y) \rightarrow (x,y)}^{s \rightarrow (s-1)} = W_{x,y}^s$.

Proof. If the experts go to sleep in reverse order, when expert r goes to sleep, we know that expert $(r+1)$ to expert K are not awake. Therefore, we know that for any $r \leq k \leq K$, the subgroup of permutations $\Pi_{r,k}^s$ is not empty. Also, there are edges between stage $(s-1)$ and stage s , and the permutations will move from $\Pi_{r,k}^s$ to subgroups $\Pi_{1,r}^{s-1}, \Pi_{2,r}^{s-1}, \dots, \Pi_{r-1,r}^{s-1}$, which is the reverse process of Theorem 3.1.3. The permutations in set $\Pi_{r,k}^s$ will have expert 1 to expert $(r-1)$ be the leading expert after r goes to sleep, the question is how to figure out the weights of the permutations with $i(1 \leq i \leq r-1)$ as the leading expert. Figure 3.2 is a sub-graph of Figure 3.1, which shows the paths of the permutations in $\Pi_{r,k}^s (r \leq k \leq K)$. Suppose that the number of different red-green paths is N . Any black path from 1 to r contains a subgroup of permutations having the same leading experts at each time, but after r wakes up, those permutations will follow different red-green paths, and finally go back to the middle node r again before r goes to sleep. We use p_1 to p_N to denote those paths. When expert r wakes up for the first time, for any black path, same fraction of the permutations will follow path p_1 to p_N because following which black path only depends on the ordering of the expert 1 to expert $(r-1)$, and following which red-green path only depends on the ordering of expert r to expert K . Let $\alpha_1, \alpha_2, \dots, \alpha_N$ to denote those fractions. When expert r wakes up for the first time, all the permutations in a black path will have the same weights. Therefore, for any black path from 1 to r , suppose that the total weights of the permutations in that path is W , we know that the weights of the permutations following $p_1, p_2, \dots, p_N$ are $W \cdot \alpha_1, W \cdot \alpha_2, \dots, W \cdot \alpha_N$. When those permutations follow the red-green path and go back to the middle node r again, the weights of those permutations will become $W \cdot \alpha_1 \cdot f_1, W \cdot \alpha_2 \cdot f_2, \dots, W \cdot \alpha_N \cdot f_N$. As a result, the ratio of the weights of the permutations follow each red-green path is the same ($\alpha_1 \cdot f_1 : \alpha_2 \cdot f_2 : \dots : \alpha_N \cdot f_N$) for each black path. Therefore, consider one of the red-green path p_u , if there are M unique black paths $1 \rightarrow r$ and let $w_1, w_2, \dots, w_M$ be the total weights of the permutations following each of those paths, then the total weights of the permutations in p_u is $\sum_{i=1}^M W_i \cdot \alpha_u \cdot f_u$, with $W_1 \cdot \alpha_u \cdot f_u, W_2 \cdot \alpha_u \cdot f_u, \dots, W_M \cdot \alpha_u \cdot f_u$ coming from each of those black paths respectively. We know that $W_{(1,r) \rightarrow (r,k)}^{(s-1) \rightarrow s}, W_{(2,r) \rightarrow (r,k)}^{(s-1) \rightarrow s}, \dots, W_{(r-1,r) \rightarrow (r,k)}^{(s-1) \rightarrow s}$ represent the total weights of the permutations at the end of the black path which follow a subset of the black paths with $1, 2, \dots, r-1$ be the last leading expert before expert r respectively. Therefore, for the permutations following any red-green path and go back to

r again, the ratio of the weights of the permutations coming from those $(r - 1)$ subsets is $\frac{W_{(1,r) \rightarrow (r,k)}^{(s-1) \rightarrow s}}{W_{(r,k)}^{(s-1) \rightarrow s}}, \frac{W_{(2,r) \rightarrow (r,k)}^{(s-1) \rightarrow s}}{W_{(r,k)}^{(s-1) \rightarrow s}}, \dots, \frac{W_{(r-1,r) \rightarrow (r,k)}^{(s-1) \rightarrow s}}{W_{(r,k)}^{(s-1) \rightarrow s}}$. The set $\Pi_{r,k}^s$ contains the permutations following one or more red-green paths, but each red-green path contains the same ratio of the weights coming from those $(r - 1)$ subsets. Therefore, for any $1 \leq i \leq r - 1$ and for any $r \leq k \leq K$, $W_{(r,k) \rightarrow (i,r)}^{s \rightarrow (s-1)} = \frac{W_{(i,r) \rightarrow (r,k)}^{(s-1) \rightarrow s}}{W_{(r,k)}^{(s-1) \rightarrow s}} \cdot W_{r,k}^s$, and $W_{i,r}^{s-1} = \sum_{k=r}^K W_{(r,k) \rightarrow (i,r)}^{s \rightarrow (s-1)} = \sum_{k=r}^K \frac{W_{(i,r) \rightarrow (r,k)}^{(s-1) \rightarrow s}}{W_{(r,k)}^{(s-1) \rightarrow s}} \cdot W_{r,k}^s$. The rest of the subgroups will remain unchanged and let $\Pi_{x,y}$ be one of them, we have $W_{x,y}^{s-1} = W_{x,y}^s$ and $W_{(x,y) \rightarrow (x,y)}^{s \rightarrow (s-1)} = W_{x,y}^s$. The rest of the subgroups will remain unchanged. $\square$

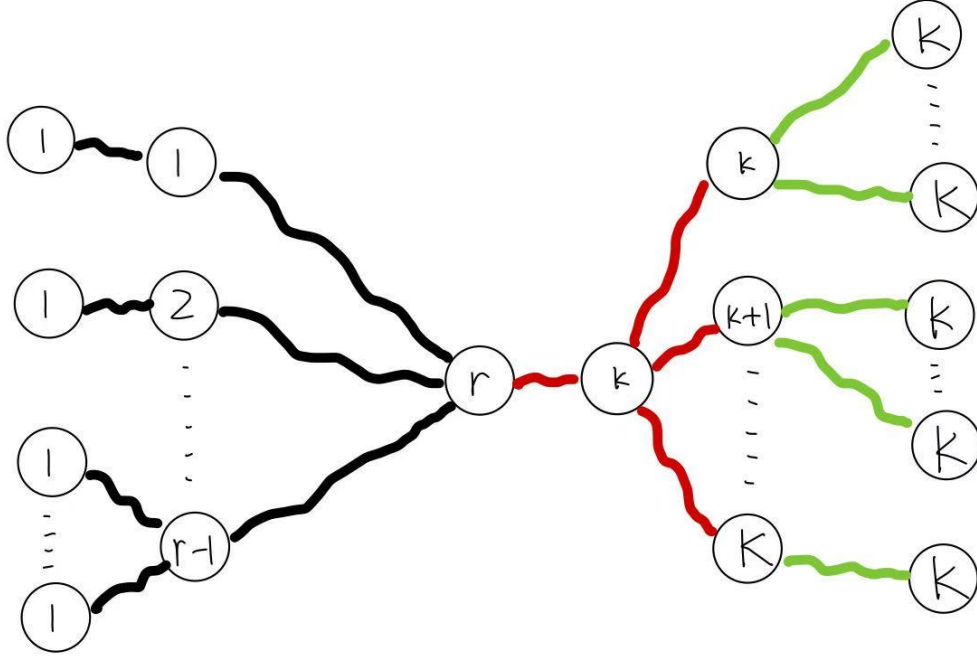


Figure 3.2: Paths of permutation in $\Pi_{r,k}^s$

Theorem 3.1.5. *When experts wake up in original order, suppose that when expert r wakes up, the stage changes from s to $(s + 1)$ and there are edges between stage s and stage $(s + 1)$. For any $1 \leq i \leq r - 1$, the permutations in the subgroups $\Pi_{i,r}^s$ at stage s will move to subgroups $\Pi_{r,r}^{s+1}, \Pi_{r,r+1}^{s+1}, \dots, \Pi_{r,K}^{s+1}$ at stage $(s + 1)$. For any $r \leq k \leq K$, $W_{(i,r) \rightarrow (r,k)}^{s \rightarrow (s+1)} = \frac{W_{(r,k) \rightarrow (i,r)}^{(s+1) \rightarrow s}}{W_{(i,r)}^{(s+1) \rightarrow s}} \cdot W_{i,r}^s$, and $W_{r,k}^{s+1} = \sum_{i=1}^{r-1} W_{(i,r) \rightarrow (r,k)}^{s \rightarrow (s+1)} = \sum_{i=1}^{r-1} \frac{W_{(r,k) \rightarrow (i,r)}^{(s+1) \rightarrow s}}{W_{(i,r)}^{(s+1) \rightarrow s}} \cdot W_{i,r}^s$. The rest of the subgroups will remain unchanged. Let $\Pi_{x,y}$ be one of them, then $W_{x,y}^{s+1} = W_{x,y}^s$ and $W_{(x,y) \rightarrow (x,y)}^{s \rightarrow (s+1)} = W_{x,y}^s$.*

Proof. Theorem 3.1.5 is a reverse version of Theorem 3.1.4. Recall that in Theorem 3.1.4, we prove that for any red-green path p_u , a fraction $\frac{W_{(i,r) \rightarrow (r,k)}^{(s+1) \rightarrow s}}{W_{(r,k)}^{(s+1) \rightarrow s}}$ of the weights of the permutations coming from the subset of black paths with expert i be the previous leading expert before expert r . Now we want to determine when expert r wakes up again after the experts go to sleep in reverse order, what are the weights of the permutations moving from $\Pi_{i,r}^s$ to $\Pi_{r,k}^{s+1}$ ($r \leq k \leq K$). From the previous proof, we know that in any of the black path in Figure 3.1, the ratio of weights of the permutations following $p_1, p_2, \dots, p_N$ and going back to r is $\alpha_1 \cdot f_1 : \alpha_2 \cdot f_2 : \dots : \alpha_N \cdot f_N$. After that, they will follow the same black path but in reverse direction and reach 1, and then follow the black path again and reach r . After they reach r again, the ratio of weights of the permutations following $p_1, p_2, \dots, p_N$ previously is still $\alpha_1 \cdot f_1 : \alpha_2 \cdot f_2 : \dots : \alpha_N \cdot f_N$. We know that the set $\Pi_{r,k}^{s+1}$ contains the permutations following a subset of paths in $p_1, p_2, \dots, p_N$, and the ratio of the weights of permutations following those paths is $\frac{W_{(r,k) \rightarrow (i,r)}^{(s+1) \rightarrow s}}{W_{(i,r)}^{(s+1) \rightarrow s}}$ according to Theorem 3.1.4. Therefore, for any $r \leq k \leq K$, $W_{(i,r) \rightarrow (r,k)}^{s \rightarrow (s+1)} = \frac{W_{(r,k) \rightarrow (i,r)}^{(s+1) \rightarrow s}}{W_{(i,r)}^{(s+1) \rightarrow s}} \cdot W_{i,r}^s$, and $W_{r,k}^{s+1} = \sum_{i=1}^{r-1} W_{(i,r) \rightarrow (r,k)}^{s \rightarrow (s+1)} = \sum_{i=1}^{r-1} \frac{W_{(r,k) \rightarrow (i,r)}^{(s+1) \rightarrow s}}{W_{(i,r)}^{(s+1) \rightarrow s}} \cdot W_{i,r}^s$. The rest of the subgroups will remain unchanged

□

3.2 Go to sleep in original order and wake up in reverse order

Similar to the previous section, we first determine which subgroups are non-empty during this period.

Lemma 3.2.1. *When the experts go to sleep in original order or when they wake up in reverse order, for any awake expert r , $\Pi_{r,k}(k > r)$ is empty and $\Pi_{r,r}$ is not empty. For any $k < r$, if k is awake, then $\Pi_{r,k}$ is empty. Otherwise, $\Pi_{r,k}$ is not empty*

Proof. When the experts go to sleep in original order or when they wake up in reverse order, if an expert r is awake, we know that for any expert $k > r$, it is also awake. According to definition of $\Pi_{r,k}(k > r)$, expert k is preceding expert r . Therefore, r cannot be the leading expert among the awake experts, and $\Pi_{r,k}(k > r) = \emptyset$. $\Pi_{r,r}$ is not empty because r is the leading expert among all experts, and r must be the leading expert among the awake experts. As for $\Pi_{r,k}(k < r)$, if k is awake, according to the definition of $\Pi_{r,k}(k < r)$, k is preceding r , so r cannot be the leading expert among the awake experts, and $\Pi_{r,k} = \emptyset$. On the other hand, if k is not awake, then any expert $j \leq k$ is not awake. According to the definition, we know that j is the largest number expert preceding expert r , so r must be the leading expert among all awake experts because any expert less than k is also not awake, and $\Pi_{r,k}$ is not empty. □

Next we will explain how to update the weights and the edges when stage changes. Recall that when the experts go to sleep in original order, we are moving from left to right in the second part of the model. When stage changes from s to $(s + 1)$, we need to use the weights in the stage s to update the weights in stage $(s + 1)$. However, if we move from stage s to stage $(s + 1)$ for the first time, there are edges between stage s and stage $(s + 1)$. Otherwise, we also have to use the edge weights in order to update weights of subgroups in stage $(s + 1)$. Theorem 3.2.2 and Theorem 3.2.4 explain how we deal with these two cases separately. On the other hand, when the experts wake up in reverse order, we are moving from right to left in the second part of the model. When stage changes from $(s + 1)$ to s , we need to use the weights in the stage $(s + 1)$ to update the weights in stage s . Since we have reached stage $(s + 1)$ before, there must be edges between stage s and stage $(s + 1)$, and we also need to use the weights of those edges to update the weights of subgroups in stage s . Theorem 3.2.3 explains how to deal with this case.

Theorem 3.2.2. *When experts go to sleep in original order, suppose that when expert r goes to sleep, the stage changes from s to $s + 1$ and there are no edge between stage s and stage $(s + 1)$. For any $(1 \leq k \leq r)$, the permutations in the subgroups $\Pi_{r,k}^s$ at stage s will move to subgroups $\Pi_{r+1,r}^{s+1}, \Pi_{r+2,r}^{s+1}, \dots, \Pi_{K,r}^{s+1}$ at stage $(s + 1)$. For any $r + 1 \leq i \leq K$, $W_{(r,k) \rightarrow (i,r)}^{s \rightarrow (s+1)} = \frac{1}{K-r} \cdot W_{r,k}^{s \prime}$, and $W_{i,r}^{s+1} = \sum_{k=1}^r W_{(r,k) \rightarrow (i,r)}^{s \rightarrow (s+1)} = \sum_{k=1}^r \frac{1}{K-r} \cdot W_{r,k}^{s \prime}$. The rest of the subgroups will remain unchanged. Let $\Pi_{x,y}$ be one of them, then $W_{x,y}^{s+1} = W_{x,y}^{s \prime}$ and $W_{(x,y) \rightarrow (x,y)}^{s \rightarrow (s+1)} = W_{x,y}^{s \prime}$*

Proof. According to Lemma 3.2.1, subgroups $\Pi_{r,1}^s, \Pi_{r,2}^s, \dots, \Pi_{r,r}^s$ are not empty. For any $(1 \leq k \leq r)$, the subgroups $\Pi_{r,k}^s$ in stage s will be empty in stage $(s+1)$ ($\Pi_{r,k}^{s+1} = \emptyset$) because after expert r goes to sleep, r will no longer be the leading experts of those permutations. Those permutations will move to subgroups $\Pi_{r+1,r}^{s+1}, \Pi_{r+2,r}^{s+1}, \dots, \Pi_{K,r}^{s+1}$ depending on which is the next leading expert.

Figure 3.3 shows all the possible paths for the permutations whose leading expert is r before r goes to sleep (permutations in $\Pi_{r,1}^s, \Pi_{r,2}^s, \dots, \Pi_{r,r}^s$). All paths start at a leaf node on the left. Note that all leaves on the left are labeled 1 because all permutations have expert 1 be the first leading expert when the experts wake up in original order. The black path is the path for those permutations when the experts wake up in original order. If a permutation follows one specific black path when the experts wake up in original order, it will follow the same path but in reverse order when the experts go to sleep in reverse order. The experts might go to sleep in reverse order and then wake up in original order for multiple times before they go to sleep in original order. Therefore, those permutations might follow the black paths back and forth for multiple times before they follow the red paths when the experts go to sleep in original order. The middle node r means that all those paths reach r before r goes to sleep. After that, when expert $r, r+1, \dots, K-1$ go to sleep one by one, those permutations will follow different green paths and finally reach K . Then, the experts will wake up in reverse order, and if a permutation follows one specific green path when the experts go to sleep in original order, it will follow the same path but in reverse direction when the experts wake up in reverse order.

For any $1 \leq k \leq r$, $\Pi_{r,k}^s$ contains the permutations following a subset of the black-red paths. If there are no edges between stage s and stage $(s+1)$, then expert r goes to sleep for the first time (when the experts go to sleep in original order). Therefore, all the permutations following a black-red path must have the same weight. Note that a permutation following which black-red path only depends on the ordering of expert 1 to expert r . Therefore, experts $r+1, r+2, \dots, K$ have equivalent chance to be the leading expert of those permutations after r goes to sleep. Therefore, for any $r+1 \leq i \leq K$, $\frac{1}{K-r}$ of the permutations following each black-red path will have expert i be the leading expert after r goes to sleep. Also, we know that for any of those permutations, r is the leading expert among expert r to expert K since r is the leading expert of those permutations before it goes to sleep. Expert i is the leading expert among expert $r+1$ to expert K since i is the next leading expert after r . Therefore, r is the largest number expert preceding expert i , and those permutations will move to subgroup $\Pi_{i,j}^{s+1}$. As a result, for any $r+1 \leq i \leq K$, $W_{(r,k) \rightarrow (i,r)}^{s \rightarrow (s+1)} = \frac{1}{K-r} \cdot W_{r,k}^s$, and $W_{i,r}^{s+1} = \sum_{k=1}^r W_{(r,k) \rightarrow (i,r)}^{s \rightarrow (s+1)} = \sum_{k=1}^r \frac{1}{K-r} \cdot W_{r,k}^s$. The rest of the subgroups will remain unchanged.

□

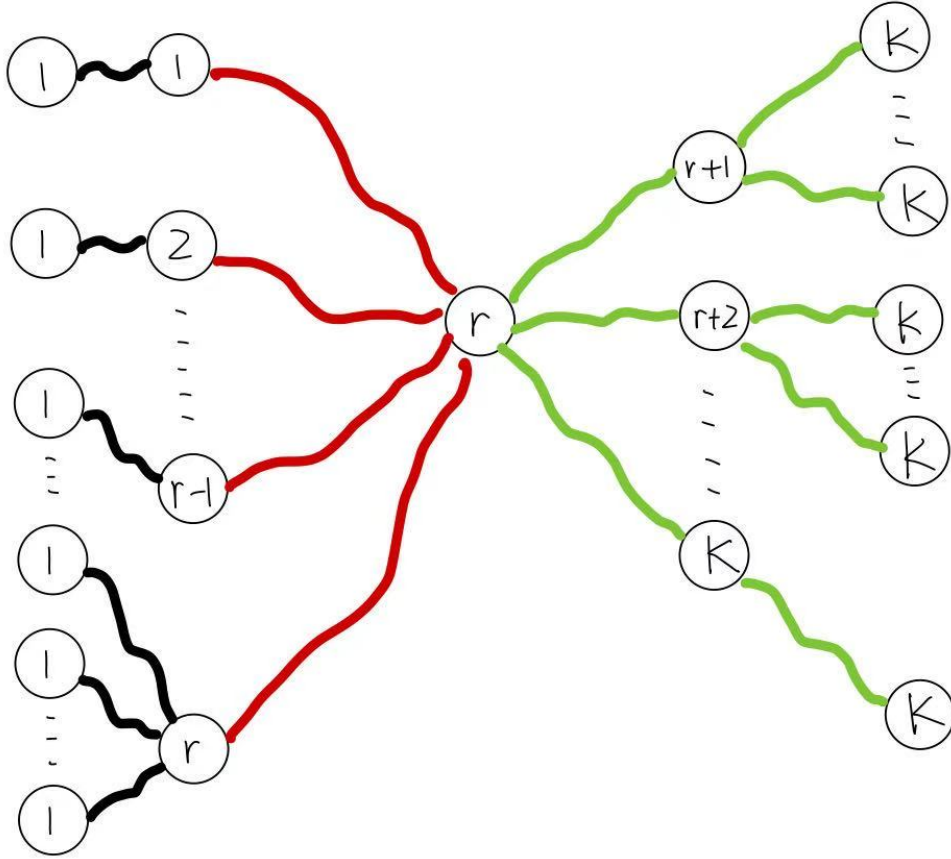


Figure 3.3: Paths of permutation in $\Pi_{r,1}^s, \Pi_{r,2}^s, \dots, \Pi_{r,r}^s$

Theorem 3.2.3. *When experts wake up in reverse order, suppose that when expert r wakes up and stage changes from s to $(s - 1)$. For any $r + 1 \leq k \leq K$, the permutations in the subgroups $\Pi_{k,r}^s$ at stage s will move to subgroups $\Pi_{r,1}^{s-1}, \Pi_{r,2}^{s-1}, \dots, \Pi_{r,r}^{s-1}$ at stage $(s - 1)$.*

For any $1 \leq i \leq r$ and for any $r+1 \leq k \leq K$, $W_{(k,r) \rightarrow (r,i)}^{s \rightarrow (s-1)} = \frac{W_{(r,i) \rightarrow (k,r)}^{(s-1) \rightarrow s}}{W_{(k,r)}^{(s-1) \rightarrow s}} \cdot W_{k,r}^s$, and $W_{r,i}^{s-1} = \sum_{k=r+1}^K W_{(k,r) \rightarrow (r,i)}^{s \rightarrow (s-1)} = \sum_{k=r+1}^K \frac{W_{(r,i) \rightarrow (k,r)}^{(s-1) \rightarrow s}}{W_{(k,r)}^{(s-1) \rightarrow s}} \cdot W_{k,r}^s$. The rest of the subgroups will remain unchanged. Let $\Pi_{x,y}$ be one of them, then $W_{x,y}^{s-1} = W_{x,y}^s$ and $W_{(x,y) \rightarrow (x,y)}^{s \rightarrow (s-1)} = W_{x,y}^s$.

Proof. Figure 3.4 is a sub-graph of Figure 3.3, which shows the paths of the permutations in $\Pi_{k,r}^s (r+1 \leq k \leq K)$. As mentioned in Theorem 3.2.4, a permutation following which black-red path only depends on the ordering of expert 1 to expert r . We also know that a permutation following which green path only depends on the ordering of expert $r+1$ to expert K . Therefore, similar to Theorem 3.1.4, we use p_1 to p_N to denote those green paths. when the experts go to sleep in original order for the first time, all the permutations in a black-red path will have the same weights. For the permutations following any of the black-red path, same fraction of the permutations will follow path p_1 to p_N . Let $\alpha_1, \alpha_2, \dots, \alpha_N$ to denote those fractions. For any black-red path, suppose that the total weights of the permutations in that path is W , we know that the weights of the permutations following $p_1, p_2, \dots, p_N$ are $W \cdot \alpha_1, W \cdot \alpha_2, \dots, W \cdot \alpha_N$. When those permutations follow the green path and go back to the middle node r again, the weights of those permutations will become $W \cdot \alpha_1 \cdot f_1, W \cdot \alpha_2 \cdot f_2, \dots, W \cdot \alpha_N \cdot f_N$. As a result, the ratio of the weights of the permutations following each green path is the same ($\alpha_1 \cdot f_1 : \alpha_2 \cdot f_2 : \dots : \alpha_N \cdot f_N$) for each black-red path. Therefore, consider one of the green path p_u , if there are M unique black-red paths and let $w_1, w_2, \dots, w_M$ be the total weights of the permutations following each of those paths, then the total weights of the permutations in p_u is $\sum_{i=1}^M W_i \cdot \alpha_u \cdot f_u$, with $W_1 \cdot \alpha_u \cdot f_u, W_2 \cdot \alpha_u \cdot f_u, \dots, W_M \cdot \alpha_u \cdot f_u$ coming from each of those black-red paths respectively. We know that $W_{(r,1) \rightarrow (k,r)}^{(s-1) \rightarrow s}, W_{(r,2) \rightarrow (k,r)}^{(s-1) \rightarrow s}, \dots, W_{(r,r) \rightarrow (k,r)}^{(s-1) \rightarrow s}$ represent the total weights of the permutations at the end of the black-red path which come from $\Pi_{r,1}^{s-1}, \Pi_{r,2}^{s-1}, \dots, \Pi_{r,r}^{s-1}$ respectively last time the stage changed from $(s-1)$ to s and those permutations moved to $\Pi_{k,r}^s$ and followed a subset of green paths. Therefore, for the permutations following any green path and go back to r again, the ratio of the weights of the permutations coming from those $(r-1)$ subgroups is $\frac{W_{(r,1) \rightarrow (k,r)}^{(s-1) \rightarrow s}}{W_{(k,r)}^{(s-1) \rightarrow s}} : \frac{W_{(r,2) \rightarrow (k,r)}^{(s-1) \rightarrow s}}{W_{(k,r)}^{(s-1) \rightarrow s}} : \dots : \frac{W_{(r,r) \rightarrow (k,r)}^{(s-1) \rightarrow s}}{W_{(k,r)}^{(s-1) \rightarrow s}}$.

The subgroup $\Pi_{k,r}^s$ contains the permutations following one or more green paths, but each green path contains the same ratio of weights coming from those $(r-1)$ subgroups.

Therefore, for any $1 \leq i \leq r$ and for any $r+1 \leq k \leq K$, $W_{(k,r) \rightarrow (r,i)}^{s \rightarrow (s-1)} = \frac{W_{(r,i) \rightarrow (k,r)}^{(s-1) \rightarrow s}}{W_{(k,r)}^{(s-1) \rightarrow s}} \cdot W_{k,r}^s$,

and $W_{r,i}^{s-1} = \sum_{k=r+1}^K W_{(k,r) \rightarrow (r,i)}^{s \rightarrow (s-1)} = \sum_{k=r+1}^K \frac{W_{(r,i) \rightarrow (k,r)}^{(s-1) \rightarrow s}}{W_{(k,r)}^{(s-1) \rightarrow s}} \cdot W_{k,r}^s$. The rest of the subgroups will remain unchanged.

□

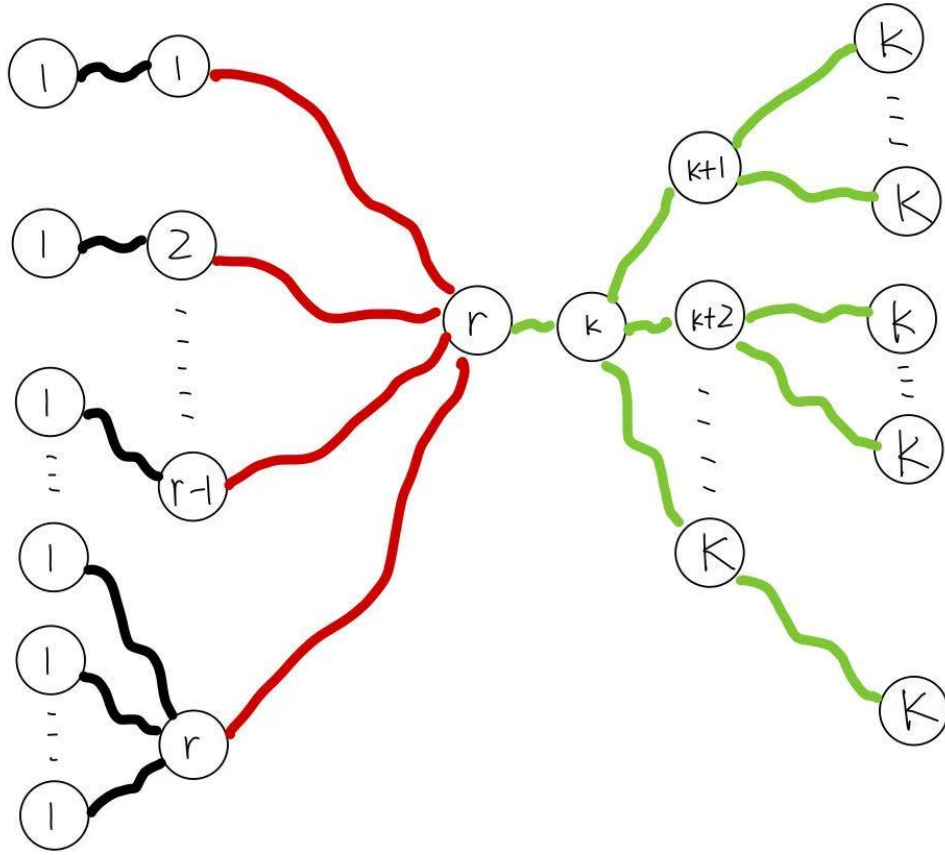


Figure 3.4: Paths of permutation in $\Pi_{k,r}^s$

Theorem 3.2.4. *When experts go to sleep in original order, suppose that when expert r goes to sleep, the stage changes from s to $(s+1)$ and there are edges between stage s and stage $(s+1)$. For any $(1 \leq k \leq r)$, the permutations in the subgroups $\Pi_{r,k}^s$ at stage s will move to subgroups $\Pi_{r+1,r}^{s+1}, \Pi_{r+2,r}^{s+1}, \dots, \Pi_{K,r}^{s+1}$ at stage $(s+1)$. For any $r+1 \leq i \leq K$,*

$W_{(r,k) \rightarrow (i,r)}^{s \rightarrow (s+1)} = \frac{W_{(i,r) \rightarrow (r,k)}^{(s+1) \rightarrow s}}{W_{(r,k)}^{(s+1) \rightarrow s}} \cdot W_{r,k}^s$, and $W_{i,r}^{(s+1)} = \sum_{k=1}^r W_{(r,k) \rightarrow (i,r)}^{s \rightarrow (s+1)} = \sum_{k=1}^r \frac{W_{(i,r) \rightarrow (r,k)}^{(s+1) \rightarrow s}}{W_{(r,k)}^{(s+1) \rightarrow s}} \cdot W_{r,k}^s$. The rest of the subgroups will remain unchanged. Let $\Pi_{x,y}$ be one of them, then $W_{x,y}^{s+1} = W_{x,y}^s$ and $W_{(x,y) \rightarrow (x,y)}^{s \rightarrow (s+1)} = W_{x,y}^s$.

Proof. Similar to Theorem 3.1.4, we use p_1 to p_N to denote those green paths in Figure 3.3. When the experts go to sleep in original order and expert r goes to sleep for the first time, for the permutations following any of the black-red path, same fraction of the permutations will follow path p_1 to p_N . Let $\alpha_1, \alpha_2, \dots, \alpha_N$ to denote those fractions. When expert r goes to sleep for the first time, all the permutations in a black-red path will have the same weights. Therefore, for any black-red path, suppose that the total weights of the permutations in that path is W , we know that the weights of the permutations following $p_1, p_2, \dots, p_N$ are $W \cdot \alpha_1, W \cdot \alpha_2, \dots, W \cdot \alpha_N$. When those permutations follow the green path and go back to the middle node r again, the weights of those permutations will become $W \cdot \alpha_1 \cdot f_1, W \cdot \alpha_2 \cdot f_2, \dots, W \cdot \alpha_N \cdot f_N$. As a result, the ratio of the weights of the permutations follow each green path is the same $(\alpha_1 \cdot f_1 : \alpha_2 \cdot f_2 : \dots : \alpha_N \cdot f_N)$ for each black-red path. After that, those permutations will follow the same black-red path and reach node 1 and then go back to node r again. Therefore, for any black-red path, the ratio of the weights of the permutations following $p_1, p_2, \dots, p_N$ is still $\alpha_1 \cdot f_1 : \alpha_2 \cdot f_2 : \dots : \alpha_N \cdot f_N$. We know that the set $\Pi_{i,r}^{s_2}$ contains the permutations following a subset of paths in $p_1, p_2, \dots, p_N$,

and the ratio of the weights of permutations following those paths is $\frac{W_{(i,r) \rightarrow (r,k)}^{(s+1) \rightarrow s}}{W_{(r,k)}^{(s+1) \rightarrow s}}$ according to Theorem 3.2.3. Therefore, for any $r+1 \leq i \leq K$, $W_{(r,k) \rightarrow (i,r)}^{s \rightarrow (s+1)} = \frac{W_{(i,r) \rightarrow (r,k)}^{(s+1) \rightarrow s}}{W_{(r,k)}^{(s+1) \rightarrow s}} \cdot W_{r,k}^s$, and $W_{i,r}^{(s+1)} = \sum_{k=1}^r W_{(r,k) \rightarrow (i,r)}^{s \rightarrow (s+1)} = \sum_{k=1}^r \frac{W_{(i,r) \rightarrow (r,k)}^{(s+1) \rightarrow s}}{W_{(r,k)}^{(s+1) \rightarrow s}} \cdot W_{r,k}^s$. The rest of the subgroups will remain unchanged

□

3.3 Observation

When the experts wake up in original order, we move from stage 1 to stage $(K+1)$ in the model. However, at any stage, we can always go back to the previous stage. That is, the last expert waking up can go to sleep and we will go back to the previous stage. This is the same when the experts wake up in reverse order, and when the experts go to sleep in

original order or reverse order. Any movement between two adjacent columns in the model is valid and does not affect the correctness of the algorithm.

3.4 Algorithm Pseudo Code

In this section, I will present the pseudo code for the algorithm. I develop two algorithms. The first algorithm update the weights when experts wake up in original order or go to sleep in reverse order. The second algorithm update the weights when experts go to sleep in original order or wake up in reverse order. There are two steps: (1) Compute the weights of the permutations moving from subgroups in the previous stage to subgroups in current stage (2) Update the weights of each subgroup in current stage given the feedback for each awake experts. We already solved step (1) in the previous sections. As for step (2), recall that the permutations which have the same leading expert receive the same loss, and the weights of those permutations will be multiplied by the same factor. We are given those factors, one for each expert: $f(1), f(2), \dots, f(K)$, and the weights of permutations in subgroup $\Pi_{i,j}$ will be multiplied by $f(i)$ since i is the current leading expert. We use M to represent our model, which consists of W - weights of subgroups, and E - edges between adjacent stages. $W_{i,j}^s \in W$ represents the weights of the subgroup $\Pi_{i,j}$ in stage s . For an edge $e \in E$ that connects $\Pi_{a,b}^{s_1}$ and $\Pi_{c,d}^{s_2}$, if the direction of the edge is from stage s_1 to stage s_2 , then $W_{(a,b) \rightarrow (c,d)}^{s_1 \rightarrow s_2}$ is the weight of the edge. Algorithm 1 below is the implementation of Theorem 3.1.3, 3.1.4 and 3.1.5. Similarly, Algorithm 2 is the implementation of Theorem 3.2.2, 3.2.3 and 3.2.4. Both algorithms return the updated weights of the current stage, and also update the weights and edges in the model.

3.5 Space and Runtime

The space needed for the algorithm is the space needed to store the model. The weights in the model requires $O(K^3)$ space since we have $2K + 1$ different stages and each stage contains K^2 weights for K^2 subgroups. As for edges, according to Algorithm 1 and Algorithm 2, we only need to maintain $O(K^2)$ edges weights between two adjacent stage, so the space we need for edges is also $O(K^3)$. Therefore, the space we need for the algorithm is $O(K^3)$. As for the runtime, it is easy to infer from Algorithm 1 and Algorithm 2 that the runtime for computing the weights in the new stage when the stage changes is $O(K^2)$.

Algorithm 1: Wake up in original order or go to sleep in reverse order

Input: The Model $M = (W, E)$;

Previous Stage and Current Stage s_1, s_2 ;

The expert that wakes up or goes to sleep r ;

Factors to update the weights $f(1), f(2), \dots, f(K)$

Output: The updated weights $W_{i,j} (1 \leq i, j \leq K)$ in current stage

```
1 for  $i = 1, 2, \dots, K$  do
2   for  $j = 1, 2, \dots, K$  do
3      $W_{i,j}^{s_2} := W_{i,j}^{s_1}$ 
4 if  $s_2 > s_1$  (experts wake up in original order) then
5   for  $i = 1, 2, \dots, r - 1$  do
6      $W_{i,r}^{s_2} := 0$ 
7   if Back edges from  $s_2$  to  $s_1$  are undefined then
8     for  $i = 1, 2, \dots, r - 1$  do
9        $W_{(i,r) \rightarrow (r,r)}^{s_1 \rightarrow s_2} := \frac{r}{K} \cdot W_{i,r}^{s_1}$ 
10       $W_{r,r}^{s_2} := W_{r,r}^{s_2} + W_{(i,r) \rightarrow (r,r)}^{s_1 \rightarrow s_2}$ 
11      for  $k = r + 1, r + 2, \dots, K$  do
12         $W_{(i,r) \rightarrow (r,k)}^{s_1 \rightarrow s_2} := \frac{r}{(k-1)k} \cdot W_{i,r}^{s_1}$ 
13         $W_{r,k}^{s_2} := W_{r,k}^{s_2} + W_{(i,r) \rightarrow (r,k)}^{s_1 \rightarrow s_2}$ 
14    else
15      for  $i = 1, 2, \dots, r - 1$  do
16        for  $k = r + 1, r + 2, \dots, K$  do
17           $W_{(i,r) \rightarrow (r,k)}^{s_1 \rightarrow s_2} := \frac{W_{(r,k) \rightarrow (i,r)}^{s_2 \rightarrow s_1}}{W_{(i,r)}^{s_2 \rightarrow s_1}} \cdot W_{i,r}^{s_1}$ 
18           $W_{r,k}^{s_2} := W_{r,k}^{s_2} + W_{(i,r) \rightarrow (r,k)}^{s_1 \rightarrow s_2}$ 
19 if  $s_2 < s_1$  (experts go to sleep in reverse order) then
20   for  $k = r, r + 1, \dots, K$  do
21      $W_{r,k}^{s_2} := 0$ 
22   for  $i = 1, 2, \dots, r - 1$  do
23     for  $k = r, r + 1, \dots, K$  do
24        $W_{(r,k) \rightarrow (i,r)}^{s_1 \rightarrow s_2} := \frac{W_{(i,r) \rightarrow (r,k)}^{s_2 \rightarrow s_1}}{W_{(r,k)}^{s_2 \rightarrow s_1}} \cdot W_{r,k}^{s_1}$ 
25        $W_{i,r}^{s_2} := W_{i,r}^{s_2} + W_{(r,k) \rightarrow (i,r)}^{s_1 \rightarrow s_2}$ 
26 for available number  $i$  do
27   for  $j = 1, 2, \dots, K$  do
28      $W_{i,j}^{s_2} := W_{i,j}^{s_2} \cdot f(i)$ 
29 Return  $W^{s_2}$ 
```

Algorithm 2: Wake up in reverse order or go to sleep in original order

Input: The Model $M = (W, E)$;

Previous Stage and Current Stage s_1, s_2 ;

The expert that wakes up or goes to sleep r ;

Factors to update the weights $f(1), f(2), \dots, f(K)$

Output: The updated weights $W_{i,j} (1 \leq i, j \leq K)$ in current stage

```

1 for  $i = 1, 2, \dots, K$  do
2   for  $j = 1, 2, \dots, K$  do
3      $W_{i,j}^{s_2} := W_{i,j}^{s_1}$ 
4 if  $s_2 > s_1$  (experts go to sleep in original order) then
5   for  $k = 1, 2, \dots, r$  do
6      $W_{r,k}^{s_2} := 0$ 
7   if Back edges from  $s_2$  to  $s_1$  are undefined then
8     for  $i = r + 1, r + 2, \dots, K$  do
9       for  $k = 1, 2, \dots, r$  do
10         $W_{(r,k) \rightarrow (i,r)}^{s_1 \rightarrow s_2} = \frac{1}{K-r} \cdot W_{r,k}^{s_1}$ 
11         $W_{i,r}^{s_2} := W_{i,r}^{s_2} + W_{(r,k) \rightarrow (i,r)}^{s_1 \rightarrow s_2}$ 
12   else
13     for  $i = r + 1, r + 2, \dots, K$  do
14       for  $k = 1, 2, \dots, r$  do
15         $W_{(r,k) \rightarrow (i,r)}^{s_1 \rightarrow s_2} = \frac{W_{(i,r) \rightarrow (r,k)}^{s_2 \rightarrow s_1}}{W_{(r,k)}^{s_2 \rightarrow s_1}} \cdot W_{r,k}^{s_1}$ 
16         $W_{i,r}^{s_2} := W_{i,r}^{s_2} + W_{(r,k) \rightarrow (i,r)}^{s_1 \rightarrow s_2}$ 
17 if  $s_2 < s_1$  (experts wake up in reverse order) then
18   for  $k = r + 1, r + 2, \dots, K$  do
19      $W_{k,r}^{s_2} := 0$ 
20   for  $i = 1, 2, \dots, r - 1$  do
21     for  $k = r + 1, r + 2, \dots, K$  do
22        $W_{(k,r) \rightarrow (r,i)}^{s_1 \rightarrow s_2} := \frac{W_{(r,i) \rightarrow (k,r)}^{s_2 \rightarrow s_1}}{W_{(k,r)}^{s_2 \rightarrow s_1}} \cdot W_{k,r}^{s_1}$ 
23        $W_{r,i}^{s_2} := W_{r,i}^{s_2} + W_{(k,r) \rightarrow (r,i)}^{s_1 \rightarrow s_2}$ 
24 for available number  $i$  do
25   for  $j = 1, 2, \dots, K$  do
26      $W_{i,j}^{s_2} := W_{i,j}^{s_2} \cdot f(i)$ 
27 Return  $W^{s_2}$ 

```

3.6 Trick to Extend the Algorithm

According to the model, after the experts go to sleep in reverse order (moving from stage $(K + 1)$ to stage 1), they can only wake up in original order (moving from stage 1 to stage $(K + 1)$). Similarly, after the experts go to sleep in original order, they can only wake up in reverse order. However, we can apply a trick to solve this problem.

Theorem 3.6.1. *When we are in stage s in the model, we can move forward to stage $(s + 1)$ or move back to stage $(s - 1)$. When we are at stage 1 or stage $(2K + 1)$, we can jump between those two stages so that when the set of awake experts is empty, the experts can wake up in original order or reverse order. We can update the weights for all subgroups in the next stage with an amortized runtime of $O(K^2)$.*

Proof. In the previous sections, we have proved that we can move from one stage to its adjacent stage and update the weights with a runtime of $O(K^2)$. If we want the experts wake up in reverse order after they go to sleep in reverse order, we need to jump from stage 1 to stage $(2K + 1)$. After the experts go to sleep in reverse order and we are at stage 1, we can pretend that we first move from stage 1 to stage $(2K + 1)$, which is also a stage with no expert awake. When we move from stage 1 to stage $(2K + 1)$, we apply the stage change algorithm mentioned above for $2K$ times, but the weight of each permutation does not change. When we are at stage $(2K + 1)$, the experts can wake up in reverse order. We can apply the same trick to jump from stage $(2K + 1)$ to stage 1 if we want the experts wake up in original order after they go to sleep in original order. Using this trick, we can extend the algorithm so that the experts can wake up in original or reverse order after they go to sleep in original or reverse order. The amortized runtime is still $O(K^2)$. $\square$

Chapter 4

Conclusion and Open Problem

In the thesis, we have demonstrated an efficient algorithm which is able to solve a much more general setting than the previous study on the dying expert setting[9]. The experts first wake up one by one in arbitrary order, which we call original order. The experts then go to sleep one by one in original order or reverse order. After that, they can wake up again in original order or reverse order. The experts can wake up and go to sleep for many times, but they must follow original order or reverse order. Also, we can go back to the previous stage at any time. That is, if an expert woke up in the previous round, it can go to sleep in the next round so we “undo” the previous round.

Although it is hard to develop a polynomial runtime algorithm to solve the whole sleeping expert setting, we can develop an algorithm to solve in using only polynomial space. Suppose that there are T rounds and K experts, there will be $K!$ permutations. However, we do not need to keep track of the weights of all permutations. Instead, we just need to store K number $w_1, w_2, \dots, w_K$ which represent the weights of the permutations with expert $1, 2, \dots, K$ as the leading expert. We also store the sets of permutations at each rounds and the loss for each awake expert at each round. In this way, for any permutation, we can calculate the weight of the permutation using the history because we know the leading expert of the permutation at each round, and we know how to update the weight of it given the loss of the leading expert. Therefore, we can iterate the permutation one by one (we can use the lexicographic next permutation algorithm, which runs in $O(K)$ time to get the next permutation), and calculate $w_1, w_2, \dots, w_K$.

The next question one might ask is that, if we want to have more freedom on the order that the experts wake up and go to sleep, can we solve it in polynomial time? The algorithm we have now only runs in $O(K^2)$ each round, and can solve original order and

reverse order. What is the relationship between the runtime and the freedom of the order? We believe this problem can still be solved in polynomial time in a more general setting. Also, we show a trick to extend the algorithm while having an amortized $O(K^2)$ bound on the runtime per stage change, but we do not know if there is a better way to extend the algorithm so that the runtime per stage change is still $O(K^2)$.

References

- [1] Ehsan Emamjomeh-Zadeh, Chen-Yu Wei, Haipeng Luo, and David Kempe. Adversarial online learning with changing action sets: Efficient algorithms with approximate regret bounds, 2021.
- [2] Yoav Freund and Manfred Opper. Drifting games and brownian motion. *J. Comput. Syst. Sci.*, 64(1):113–132, February 2002.
- [3] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.
- [4] Yoav Freund, Robert E. Schapire, Yoram Singer, and Manfred K. Warmuth. Using and combining predictors that specialize. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '97, page 334–343, New York, NY, USA, 1997. Association for Computing Machinery.
- [5] Satyen Kale, Chansoo Lee, and Dávid Pál. Hardness of online sleeping combinatorial optimization problems, 2016.
- [6] Varun Kanade and Thomas Steinke. Learning hurdles for sleeping experts. *ACM Trans. Comput. Theory*, 6(3), July 2014.
- [7] Robert Kleinberg, Alexandru Niculescu-Mizil, and Yogeshwer Sharma. Regret bounds for sleeping experts and bandits. *Mach. Learn.*, 80(2–3):245–272, September 2010.
- [8] Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Inf. Comput.*, 108(2):212–261, February 1994.
- [9] Hamid Shayestehmanesh, Sajjad Azami, and Nishant A Mehta. Dying experts: Efficient algorithms with optimal regret bounds. *Advances in Neural Information Processing Systems*, 32, 2019.

- [10] Vladimir Vovk. Aggregating strategies. In M. Fulk and John Case, editors, *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 371–383. Morgan Kaufmann, 1990.
- [11] Vladimir Vovk. A game of prediction with expert advice. *J. Comput. Syst. Sci.*, 56:153–173, 1995.