Distributed Flow Scheduling in an Unknown Environment

Yaoqing Yang *†, Keqin Liu †, Pingyi Fan*
* Department of Electronic Engineering, Tsinghua University, Beijing, China
† Department of Electrical and Computer Engineering, UC Davis, California, USA
Email: yqyang1991@gmail.com, kqliu@ucdavis.edu

***Abstract*—Flow scheduling is crucial in the next-generation network but hard to address due to fast changing link states and tremendous cost to explore the global structure. In this paper, we first design a distributed virtual game to solve the optimization of flow scheduling problem assuming the priori knowledge of the distribution of edge cost as a random variable. In our virtual game, we use incentives to stimulate selfish users to reach a Nash Equilibrium Point which is suboptimum based on the analysis of the 'Price of Anarchy'. This algorithm is then generalized into the situation with unknown cost distribution, where the ultimate goal is to minimize the cost in the long run. In order to achieve a reasonable tradeoff between exploration of cost distribution and exploitation with limited information, we model this problem as a Multi-armed Bandit Game and combine the newly proposed DSEE with our virtual game design. Armed with these powerful tools, we find a totally distributed algorithm to ensure the logarithmic growing of regret with time, which is optimum in classic Multi-armed Bandit problem. Theoretical proof and simulation results both confirm the effectiveness of our algorithm. To the best of our knowledge, this is the first work to combine multi-armed bandit with distributed flow scheduling.**

Keywords–Distributed Flow Scheduling, Price of Anarchy, Multi-Armed Bandit, Logarithmic Regret

I. INTRODUCTION

Flow scheduling has wide applications in ATM and MPLS networks. But it is still an open problem. The flow scheduling can be formulated as an optimization problem, where all the routers in the network cooperate to find a network-wide sharing scheme. Since the network cost is usually a superlinear function of flow amount, the main concern is to handle possible congestion on the same edge to reduce the network cost. In order to be useful in practice, a distributed solution to such an optimization problem is also crucial. Moreover, control signals and information exchanges should be relatively small. There has been much related literature on these problems.

With development of the widely-used MPLS network, the *minimum interference routing*[1]−[4] has become a mainstream in flow scheduling and traffic engineering. However, minimum interference routing algorithms, like MIRA[2] and WSP[4], mainly consider load balancing to maintain the 'sustainability' to admit future flows. The objective of cost minimization cannot be guaranteed. Besides, the atomic routings in routing games[5][7]−[9] have attracted much attention. Although the atomic routings[5] can be used to handle the flow scheduling problem directly, it is implemented with players deciding the best paths, which is impractical in nature. A more reasonable solution should be to let distributed routers carry out the path selection. Moreover, [5] only considers the environment with known edge costs. Its generalization to an unknown environment is nontrivial.

To address the routing problem with unknown edge cost, the Multi-armed Bandit (MAB) algorithm recently has emerged as an effective way[13]−[15]. The classic MAB considers a single player and N independent arms, each arm, if played, incurs a random cost with an unknown distribution. The player should decide the sequence to play each arm to obtain the minimum cost in the long run. Its key point lies in maintaining a reasonable tradeoff between exploration and exploitation, which respectively means to play a new arm and learn its cost distribution or to play the arm with minimum cost. A frequently used criterion on judging an adopted sequence is the so called *regret* or *cost of learning*, defined as the difference in total cost between the chosen sequence and the optimum sequence. The best regret, logarithmically growing with time, is obtained in [10] by Lai and Robbins. In [11][12], the index-type policies are provided to achieve logarithmic regret. Routing problems with unknown edge cost distributions can be modeled as a MAB problem if each path is viewed as an arm. However, paths with shared edge cannot be viewed as independent. This issue has been partially addressed in [13]-[15] by generalizing the classic MAB algorithm into shortest path problem. But these methods do not apply in routing problems with congestion.

In this paper, we firstly focus on the known scenario to solve the optimization of flow scheduling. We will devise a novel virtual game, in which the incentives are used to stimulate selfish users to reach a Nash Equilibrium point. In contrast to the atomic routing game, the virtual game is implemented with distributed routers, instead of players, deciding the best paths. Moreover, our virtual game is completely online, with no direct information exchange between users. We prove that our virtual game reaches a Nash Equilibrium Point in limited circles. The performance of the Nash Equilibrium point is analyzed based on the Price of Anarchy, defined as the ratio of total cost in a Nash Equilibrium Point to an optimum total cost. In the second half, the proposed algorithm is generalized into an environment with unknown

edge costs, with the expectations of these costs estimated through exploration. Then the exploitation starts by applying the virtual game based on the exploration results. Like most MAB problems, explorations introduce extra cost, or reward loss. So we propose to use the DSEE Sequence[17] to optimize the exploration time. The ultimate object of our design is to minimize long-run total cost for the whole network, so the time consumptions for both explorations and exploitations are taken into account. Instead of making modifications to classic MAB algorithms, we solve the congestion problem beforehand in the virtual game designing. We prove that the regret also grows logarithmically with time, which is optimum in MAB problem.

II. SYSTEM MODEL

A. Cost Modeling

We consider a flow scheduling problem in a graph $G = (V, E)$. Assume that $|V| = N$ which represents the number of the distributed routers. There are K flows, each with unit amount $f_k = 1$ and source-destination pairs (s_k, t_k) where $s_k, t_k \in V$ and $s_k \neq t_k$. For simplicity, in the following we use 'kth flow' and 'user k' interchangeably. For each edge $e \in E$, define flow summation on the edge

$$f_e = \sum_{k:e\in p_k} f_k \tag{1}$$

in which the p_k represents the path chosen by the kth flow. Since all flows have unit amount, each flow summation f_e takes a discrete value from $\{1, 2, ..., K\}$. Define

$$C(F) = \sum_{e\in E} c_e(f_e) \tag{2}$$

as the total cost in one time slot, in which the $c_e(f_e)$ represents the cost for edge e. Note that with a different flow summation f_e, $c_e(f_e)$ is a different random variable whose expectation value increases when f_e grows. When f_e is fixed, for different time slots, $c_e(f_e)$ is an i.i.d. random process. F denotes the whole flow distribution on the network. $F = (f_{e_1}, f_{e_2} \cdots f_{e_{|E|}})$. We should state here that flow distribution is a vector representing the flow summation in each edge, while we may refer to another meaning which represents the distribution of a random variable. There is clear a distinction between these two. In order to minimize the time average of $C(F)$, we try to obtain the best flow distribution F in a distributed way to minimize the expectation of $C(F)$. Henceforth we use a bar to represent the expectation. For example, $\bar{C}(F)$ denotes the expectation of $C(F)$. The unit amount is the granularity of all flows. Obviously, generalization to multi-commodity scenario is trivial if we split flows into flow units and treat each unit as an independent sub-flow.

B. Incentive

Although the final goal of us is to minimize the expected total cost by finding a best flow distribution $C(F)$, this social warfare point might not meet the needs of all the users. Furthermore, in our paper, we assume that users do not exchange information directly. So users are blind and only trying to optimize its own usage, which may lead to congestions over common links. In order to stimulate users to cooperate, we set revenues as incentives for them. Assume at some time t, there are already K_t flows in the network and the whole flow distribution is currently F_t. Then the whole cost of the network equals $\bar{C}(F_t)$. For a certain kth flow, let $F_t(k)$ denote the flow distribution when f_k is withdrawn from F_t. We define

$$\bar{C}(F_t) - \bar{C}(F_t(k)) \tag{3}$$

as the revenue for the kth flow. Obviously, when user k has a chance to change its routing path, he must choose the path that introduces the minimum extra cost (3). Since $\bar{C}(F_t(k))$ is independent of user k, the total cost will decrease surely. Obviously, this is an game theoretical method to minimize the total cost (2). We are trying to get the social warfare point but the distributed and blind users are only trying to optimize its own revenue. Thus it is required to find a properly designed mechanism to set revenue for all the users and guide them toward the social warfare point. This game is only virtual since users are not real competitors. We are only trying to use this virtual game to achieve the optimum point when $E(C(F))$ is minimized.

C. System Description

In this paper, we focus on the distributed case when a central controller is implicit. That is to say, our algorithm is essentially different from some classic ones where a central controller collects all the link information and broadcasts the result of a centralized algorithm. In contrast, in our settings, all the routers form a distributed operating system and each router only communicates with its direct neighbors in the graph G. In this case, we propose a virtual game (detailed in algorithm 1) which only needs local information to calculate link prices. For the exploration period, we present another algorithm operated in a distributed way.

Nevertheless, we still need a central controller which helps start the algorithm and broadcast some initial settings. Also, a central controller is needed to broadcast the timing information periodically. However, the adoption of our distributed virtual game and the distributed exploration algorithm really reduce the refinements and control information of the central controller.

III. ALGORITHM IN KNOWN MODEL

A. Virtual Game Design

In this section, a virtual game design will be proposed to solve the optimization problem of flow scheduling. We first consider the known environment, i.e., assume that routers know all $\bar{c}_e(f_e)$ apriori, with the unknown environment discussed later. Each user takes turns to hire routers to do Bellman Ford Algorithm[16]. The price for each edge is set as the incentive described in II.B. The total cost decreases

each time when a user changes path as indicated in part II.B. Besides, there will be $N * K$ time slots reserved for one circle. So time reserved for each user is N slots, enough for Bellman Ford Algorithm to converge[16].

The details are described in algorithm 1. This represents one circle of the virtual game. The virtual game will go on until all the users reach a Nash Equilibrium Point. We discuss the equilibrium point in the next subsection.

Algorithm 1 One cycle of the Virtual Game (NK time slots)

Initialize

The central controller collects and broadcasts information like numbers of nodes N, number of flows K.

For k from 1 to K

1) Take out the kth flow from current flow distribution. This could be carried out by each router to clear flow k in its memory. If it is the first time for this flow to do step (1), then no memory about flow k has been established, so we do not need to take it out.

2) Calculate price on each edge. The price is the extra cost if this edge is chosen:

$$P_e(F) = \bar{c}_e(f_e) - \bar{c}_e(f_e - f_k) \tag{4}$$

Here F and f_e are temporary flow distribution and flow summation (1) if flow k chooses e.

3) Start the Bellman Ford Algorithm from source s_k and wait for N slots to ensure its convergence. Find out the path p_k with the minimum price (7) to transmit flow to d_k.

4) Add up f_k on each selected edge to transmit the kth flow.

end

B. Nash Equilibrium Point

Theorem 1: The proposed algorithm 1 in III.A guarantees that after finite circles the whole network always reaches a Nash Equilibrium Point, and the Convergence time is bounded in $\lceil \frac{S_M}{S_m} \rceil$ times of Bellman Ford circles. Here S_M denotes the maximum difference between cost of two different flow distributions, and S_m denotes the minimum.

Proof: During one circle, one of two events below must occur:

a).At least one user changes his path.

b).No one changes his path.

If event 'b' happens, we know that no one could change his path unilaterally. Obviously the network has reached Nash Equilibrium Point.

However, if event 'a' happens, total cost decreases. This has been stated in II.B. Since there will be limited paths for one flow to take, the number of flow distributions will be limited, too. So 'a' won't happen all the time.

We can further figure out the upper bound of convergence time to reach a Nash Equilibrium Point. In fact, we need $\lceil \frac{S_M}{S_m} \rceil$ times of Bellman Ford circles. This is true because during each Bellman Ford circle, the cost of the whole network will at least decrease by S_m if 'b' does not happen. □

The basic idea of the proof is: In each circle if one user changes his path, total network cost decreases. But this can not happen forever since the optimum point exists. In Fig.3 we show the behavior of 'regret' growth which will be defined properly. In this figure one can directly see the fast convergence to an Equilibrium Point, even the environment has been changed to an unknown one.

IV. PRICE OF ANARCHY

In this part we will analyze Algorithm 1 in terms of 'Price of Anarchy'. This notion was originally defined in [8]. In [9], the authors analyzed the price of anarchy of an atomic routing game consisting of polynomial edge cost with nonnegative coefficient. The results of $d^{O(d)}$ was revealed, in which d represents the highest order of the polynomial edge cost function. This result is confirmed in [5]. In this paper, we still need analysis of the 'Price of Anarchy' since our ultimate goal is to solve an optimization problem. Though we have shown that our algorithm is able to make users reach a Nash Equilibrium Point, it is useful to figure out the difference between a Nash Equilibrium Point and the optimum point.

Definition 1: We define the **Price of Anarchy** as

$$\bar{C}(F_N)/\bar{C}(F^*) \tag{5}$$

The F_N represents flow distribution of one Nash equilibrium point. And F^* represents flow distribution of the optimum point in terms of minimizing the network cost.

Next we will first investigate the existence of constant price of anarchy for general polynomial edge cost. Then we will provide concrete value for polynomials with nonnegative coefficients. Here we assume the functions to be convex, a trivial condition when congestion is concerned[3][5][6]. The assumption of polynomial edge cost is common in previous work of Routing Games[5][8][9]. Moreover, polynomial functions are quite enough to model congestions in our problem, so we still use this assumption.

A. General Polynomial Function: Existence

Definition 2: We define **Total Price** for distribution F as

$$P(F) = \sum_{e \in E} [\bar{c}_e(f_e) - \bar{c}_e(f_e - 1)] \cdot f_e \tag{6}$$

We simply replace unit flow with 1 in the sequel, as claimed in section II.A. What's important is why we define (6) as the 'total price'. In fact, from (3)(4) we know that the incentive pricing scheme asks for the kth user a price of

$$P_k(F) = \sum_{e \in p_k} [\bar{c}_e(f_e) - \bar{c}_e(f_e - 1)] \tag{7}$$

We add up (7) for all K users and simply change the order of summation. It yields (6).

Theorem 2: If the expectation of edge cost function $\bar{c}_e(f_e)$ is convex and grows polynomially with f_e, as shown in (8), there exists a constant bound for the 'Price of Anarchy' independent of network size and flow distribution.

$$\bar{c}_e(f_e) = a_e f_e^d + \sum_{i=1}^{d} a_e^{(i)} f_e^{d-i} \tag{8}$$

To prove this theorem, we need the following two lemmas.
Lemma 1: For a given network G=(V,E), there exist some constant numbers $A_l, A_r, A_u$and a constant set $J = [J_L, J_R]$. For any flow distribution F, we have

$$A_l \le \frac{P(F)}{\bar{C}(F)} \le A_r \tag{9}$$

For any flow distribution and any edge e, we have

$$\frac{\bar{c}_e(f_e+1) - \bar{c}_e(f_e)}{\bar{c}_e(f_e) - \bar{c}_e(f_e-1)} \le A_u \tag{10}$$

$$J_L \le \frac{\bar{c}_e(f_e) - \bar{c}_e(f_e-1)}{f_e^{d-1}} \le J_R \tag{11}$$

These constant numbers are all independent of the network size and flow distribution.

The nature of Lemma 1 is very simple. With polynomial $\bar{c}_e$, numerator and denominator of the fractions in (9)(10)(11) are of the same order of flow summation f_e. When f_e grows larger, these fractions all converge to constants. So these fractions are certainly bounded. Rigorous proof of this lemma is put in Appendix A [18].

Lemma 2: For a given network G=(V,E) and a Nash Equilibrium point F of K users, for any flow distribution F', we have

$$P(F) \le A_u \sum_{e\in E} [\bar{c}_e(f_e) - \bar{c}_e(f_e-1)] \cdot f_e{}' \tag{12}$$

This is the 'Variational Inequality Characterization'[5], which describes the basic feature of a Nash Equilibrium Point. The rigorous proof of Lemma 2 is in Appendix B [18]. The key is the combining of (10) with the following feature (equation (30) of [18]) for a Nash Equilibrium Point F

$$P(F) \le \sum_{e\in E} [\bar{c}_e(f_e+1) - \bar{c}_e(f_e)] \cdot f_e' \tag{13}$$

Proof of Theorem 2: Let F represent a random Nash Equilibrium point and F^* denote the optimum point. For a certain edge e, from (11), we have

$$\frac{[\bar{c}_e(f_e) - \bar{c}_e(f_e-1)]/f_e^{d-1}}{[\bar{c}_e(f_e^*) - \bar{c}_e(f_e^*-1)]/(f_e^*)^{d-1}} \le \frac{J_L}{J_R} \tag{14}$$

Here * represents the optimum point. From this inequality we can derive directly and get

$$\begin{aligned} &[\bar{c}_e(f_e) - \bar{c}_e(f_e-1)] \cdot f_e^* \\ \le &(\frac{J_L}{J_R})^{\frac{1}{d}} \{[\bar{c}_e(f_e) - \bar{c}_e(f_e-1)] \cdot f_e\}^{\frac{d-1}{d}} \\ &\cdot \{[\bar{c}_e(f_e^*) - \bar{c}_e(f_e^*-1)] \cdot f_e^*\}^{\frac{1}{d}} \end{aligned} \tag{15}$$

Combining (15) with the $H\ddot{o}\ lder$ inequality, we get

$$\begin{aligned} &\sum_{e\in E} [\bar{c}_e(f_e) - \bar{c}_e(f_e-1)] \cdot f_e^* \\ \le &(\frac{J_L}{J_R})^{\frac{1}{d}} \{\sum_{e\in E} [\bar{c}_e(f_e) - \bar{c}_e(f_e-1)] \cdot f_e\}^{\frac{d-1}{d}} \\ &\cdot \{\sum_{e\in E} [\bar{c}_e(f_e^*) - \bar{c}_e(f_e^*-1)] \cdot f_e^*\}^{\frac{1}{d}} \\ = &(\frac{J_L}{J_R})^{\frac{1}{d}} \cdot [P(F)]^{\frac{d-1}{d}} \cdot [P(F^*)]^{\frac{1}{d}} \end{aligned} \tag{16}$$

From (12) we arrive at

$$P(F) \le A_u \cdot (\frac{J_L}{J_R})^{\frac{1}{d}} \cdot [P(F)]^{\frac{d-1}{d}} \cdot [P(F^*)]^{\frac{1}{d}} \tag{17}$$

It means

$$P(F)/P(F^*) \le (A_u)^d \cdot \frac{J_L}{J_R} \tag{18}$$

And we have Lemma 1, so we finally get

$$\begin{aligned} \bar{C}(F)/\bar{C}(F^*) = &\frac{\bar{C}(F)}{P(F)} \cdot \frac{P(F)}{P(F^*)} \cdot \frac{P(F^*)}{\bar{C}(F^*)} \\ \le &(A_u)^d \cdot \frac{A_r}{A_l} \frac{J_L}{J_R} \end{aligned} \tag{19}$$

From previous Lemmas, we know absolutely that constants on the right side of this inequality are independent from network topology and flow distribution. Since F_N is a random flow distribution, **Theorem 2** has been proved. □

B. Polynomial Function with Nonnegative Coefficients: Concrete Value

For polynomial edge cost with nonnegative coefficients, here we will give out concrete value of the bound. We take advantage of the nonnegative coefficients to simplify proof in IV.A. First we give out some definitions. If (8) holds and coefficients are all nonnegative, we have the following equation for each edge e

$$\begin{aligned} \bar{c}_e(f_e+1) - \bar{c}_e(f_e) = &a_e[(f_e+1)^d - f_e^d] \\ &+ \sum_{i=1}^{d} a_e^{(i)}[(f_e+1)^{d-i} - f_e^{d-i}] \end{aligned} \tag{20}$$

Obviously, all terms in (20) have nonnegative coefficients. We assume

$$\bar{c}_e(f_e+1) - \bar{c}_e(f_e) = \sum_{i=0}^{d} \tilde{a}_e^{(i)} f_e^{d-i-1} \tag{21}$$

in which $\tilde{a}_e^{(i)} > 0$ and $\tilde{a}_e^{(0)} = a_e$. Moreover,

$$\sum_{i=0}^{d} \tilde{a}_e^{(i)} = \bar{c}_e(1) - \bar{c}_e(0) \tag{22}$$

We assume

$$s_e = \min_{a_e^{(i)}>0} (a_e^{(i)}) \tag{23}$$

$$L = \max_{e\in E}[\bar{c}_e(1) - \bar{c}_e(0)] \tag{24}$$

Theorem 3: For a given network G=(V,E), if all edge cost functions satisfy (8) and coefficients are nonnegative, constant upper bound of the Price of Anarchy exists. The constant is $[(d+1)L \max_{e\in E} \frac{1}{s_e}]^d = d^{O(d)}$.

Proof: Conditions in this theorem also ensure the functions to be convex. This results in, for any flow distribution

$$\bar{C}(F) \le P(F) \tag{25}$$

From (22)(24) we have, for any i and any $e \in E$

$$\tilde{a}_e^{(i)} < L \tag{26}$$

Based on the hölder inequality, we have

$$\begin{aligned}
&\sum_{e\in E}[\bar{c}_e(f_e+1) - \bar{c}_e(f_e)] \cdot f_e^* \\
=&\sum_{i=0}^{d}\sum_{e\in E} \tilde{a}_e^{(i)} f_e^{d-i-1} f_e^* \\
\le&\sum_{i=0}^{d}\{\sum_{e\in E} \tilde{a}_e^{(i)} (f_e^{d-i-1})^{\frac{d-i}{d-i-1}}\}^{\frac{d-i-1}{d-i}} \{\sum_{e\in E} \tilde{a}_e^{(i)} (f_e^*)^{d-i}\}^{\frac{1}{d-i}} \\
\le&L\sum_{i=0}^{d}\{\sum_{e\in E} \frac{1}{s_e}\bar{c}_e(f_e)\}^{\frac{d-i-1}{d-i}} \{\sum_{e\in E} \frac{1}{s_e}\bar{c}_e(f_e^*)\}^{\frac{1}{d-i}} \\
\le&L \max_{e\in E}\frac{1}{s_e} \cdot \sum_{i=0}^{d}\{\bar{C}(F)\}^{\frac{d-i-1}{d-i}} \{\bar{C}(F^*)\}^{\frac{1}{d-i}}
\end{aligned}$$

Since $\bar{C}(F^*) \le \bar{C}(F)$, we have

$$\begin{aligned}
&\sum_{e\in E}[\bar{c}_e(f_e+1) - \bar{c}_e(f_e)] \cdot f_e^* \\
\le&(d+1)L \max_{e\in E}\frac{1}{s_e} \cdot \{\bar{C}(F)\}^{\frac{d-1}{d}} \{\bar{C}(F^*)\}^{\frac{1}{d}}
\end{aligned} \tag{27}$$

For one random Nash equilibrium F and the optimum point F^*, it follows from (13) and (25) that

$$\bar{C}(F) \le P(F) \le \sum_{e\in E}[\bar{c}_e(f_e+1) - \bar{c}_e(f_e)] \cdot f_e^* \tag{28}$$

Combining (27)(28), we have

$$\bar{C}(F)/\bar{C}(F^*) \le [(d+1)L \max_{e\in E}\frac{1}{s_e}]^d = d^{O(d)} \tag{29}$$

And this constant is independent of network topology and flow distribution.□

V. ALGORITHM IN UNKNOWN MODEL

In this section, we will generalize the proposed algorithm above to address the unknown model where the cost distribution of each edge is unknown at the beginning. We adopt the newly proposed DSEE Sequence in [17] which cuts time into interleaving exploration and exploitation periods. We first give out description of DSEE (Figure 1) and then describe exploration and exploitation period respectively.

A. *Definition of the DSEE*

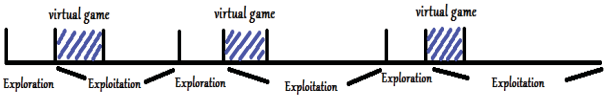

Fig. 1: DSEE sequence

A Deterministic Sequencing of Exploration and Exploitation (DSEE) is a time division scheme which cuts time into interleaving exploration and exploitation periods, as shown in Figure 1. An exploration period begins this sequence, then an exploitation sequence follows. Then the second exploration period starts. Slightly different from the original sequence defined in [17], we put one Virtual Game cycle at the beginning of each exploitation period. One cycle of the virtual game is defined in Algorithm 1.

Without loss of generality, we set that one exploration period ends after N time slots. In our scheme, at the beginning of each exploitation period, the first $N*K$ time slots are arranged to conduct one cycle of the virtual game, with the rest time of the exploitation used for routing or exploitation of the network. Exploitation terminates at time slot t when t satisfies the following inequality for the first time in this exploitation period.

$$card(t) < Glog(t) \tag{30}$$

In this inequality the $card(t)$ represents sum number of time slots used to do exploration up to time t. The parameter G is called the DSEE parameter and determined by the central controller beforehand. Certainly, the whole DSEE Sequence is determined beforehand once the parameter G has been chosen. Exploration periods and Bellman Ford periods cannot be too long since they introduce extra cost to the network.

B. *Exploration*

One exploitation period lasts for $N = |V|$ time slots. In one exploration period, only one source node s_k starts exploration. And K source nodes do exploration in turn, in different exploration periods. At the beginning of the first exploration period, s_1 sends out a short flow of a random amount f_1 to a random edge e_r related to it to explore a sample of the random variable $c_{e_r}(f_1)$. Then the other node of edge e_r receives this flow and forward it in the next time

slot. After one exploration, one sample of each $c_e(f)$ is stored in the router related to e and the router uses accumulated sample mean $\hat{c}_e(f)$ to approximate each expectation $\bar{c}_e(f)$. This whole exploration period terminates in $N = |V|$ time slots. In the next exploration period, as mentioned above, it is the turn of s_2 to start exploration instead of s_1. And so forth. This can be carried out with a token. If node s_n is a source node of a flow, it gets a token number α_n and sets the initial value to 1, which means that the next exploration period will be started by s_1. After each exploration period, α_n adds up with 1, which means that next flow will start exploring at the beginning of the next exploration period. When α_n reaches $K + 1$, it returns to 1. Noting that all α_n has the same value since the DSEE sequence is determined by parameter G and thus the same to all the nodes, we define α which is equal to the common value of α_n. Besides, the constant number $N = |V|$ is large enough to ensure a minimum probability $r = min_{e\in E, 1\leq f\leq K}(r_e(f)) > 0$, in which the r_e is the probability of the edge e being estimated under edge flow summation f.

C. *Exploitation*

At the beginning of each exploitation period, we implement one circle of the virtual game described in III.A.

However, we should replace (4) with

$$P_e(F) = \hat{c}_e(f_e) - \hat{c}_e(f_e - f_k) \tag{31}$$

where $\hat{c}_e(f_e)$ denotes the sample means.

We end this section with Algorithm 2 which describes the whole algorithm in the unknown model.

Algorithm 2 DSEE with a virtual game

Initialize

The central controller collects and broadcasts information like numbers of nodes N, number of flows K and the DSEE parameter G.

For each node n (from 1 to N)

Node s_n calculates the DSEE sequence with inequality (30) and memorize the beginning of each period. If node s_n is a source node of a flow, it defines a token number α and sets the initial value to 1.

end

For T from 1 to infinity

For each node n (from 1 to N)

1) Start counting an exploration period. End it after N time slots. At the beginning of this period, s_α starts the exploration by sending out the short exploration flow to a random neighbor. Then this node sends it out to another one randomly, until the end of this exploration period. Increase α with 1.
2) Start a cycle of the virtual game defined in algorithm 1. Wait for NK slots until the end of this cycle.
3) Start exploiting the network, end this exploitation period until time index t satisfies inequality (30).

end

end

VI. ANALYSIS IN REGRET

The conventional *regret* in a MAB problem is defined as the difference in total cost between the chosen strategy sequence and the optimum strategy sequence when cost distribution is known. In section III.B, we have proved that K users guarantee to reach the Nash Equilibrium Point in limited circles of virtual game. Based on this fact, we would provide a modified definition of *regret* different from the conventional one.

Definition 3: We define *regret* as the number of time slots when the network is not in a Nash Equilibrium Point.

Next we will reveal the equivalence of the new regret definition with the conventional one. Then we will prove that the regret grows logarithmically with time.

A. *Equivalence between new regret definition and conventional one*

In our algorithm, there exist two conditions that regret increases. The first one is exploration or Bellman Ford. During these periods, no flows are transmitted. However, if we define an extra constant cost for each of such slot to get a classic definition, we can see that this two regrets grow with time in the same order. The second one is when flows are not routed in a Nash Equilibrium Point in an exploitation period. But in one such slot, extra cost cannot be larger than S_M, the largest and bounded possible total cost for the whole network[18]. Therefore, even if we define a classic regret, it still grows with the same order of time.

The only difference is the distance from one Nash Equilibrium Point to the Optimum Point. However, finding the Optimum Point for different flows tends to be NP hard and it cannot be implemented in a distributed way. So we choose to define regret based on a sub-optimal Nash Equilibrium Point which cannot be further improved in a distributed manner. Previous parts have shown the constant 'Price of Anarchy' bound, which convince the feasibility of our definition.

B. *Regret Order*

Theorem 4: If the chosen G in (14) satisfies

$$G \geq max(3/r, \frac{8m^2|E|\sigma^2}{rc^2}) \tag{32}$$

then regret(T) increases with $O(log(T))$ when doing the algorithm in section V. Some definitions in (32) are needed:

Definition 4: Let S be a m-dimensional vector space. A set

$B = \{x_1, x_2, ..., x_m\} \subset S$ is called a barycentric spanner for S if every x in S can be written as linear combination of elements of B with coefficients in $[-1, 1]$.

It is shown in [15] that if S is a compact set, then it has a barycentric spanner. We know that the set of different paths for a certain source-destination pair (s_k, d_k) is a compact vector space, thus it has a barycentric spanner with dimension m_k. We assume $m = \max_{k=1\sim K} m_k$. σ^2 is the largest variance of all the edge cost under different flow distributions. r is the minimum of the probability that a certain edge is chosen during explorations. c_k is the minimum price difference between two paths for the kth user under all different flow distributions. Since number of flow distributions is limited, c_k surely exists. Then we can define $c = \min_{k=1\sim K} c_k$. These parameters are all related to the network topology and can be obtained beforehand. However, while choosing a G based on (32) is doable, usually we can choose a smaller G. Here we only concern about the existence of a sufficient condition.

Proof of Theorem 4 is put in Appendix E in [18]. Here we provide the basic idea. If G is chosen big enough, sufficient times will be used for exploration so that we have relatively accurate sample means for the cost of each edge under different flow summation. Based on Bernstein's inequality, we can bound the variance of sample means of path cost. When this variance is small enough, we can bound the probability that we make mistakes in the virtual game circle. Mistake-free virtual game circles result in Nash Equilibrium (Theorem 1).

VII. SIMULATIONS

A. *Numerical Results of the Price of Anarchy*

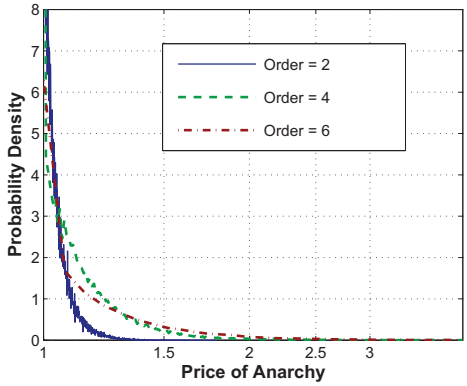

Fig. 2: 'Price of Anarchy' distribution

In this part we illustrate simulation results of the proposed algorithm in terms of the 'Price of Anarchy'. We run the simulations for many times in a graph of 10 nodes and 6 flows. Each time we run several circles of the proposed algorithm 1. When the total cost does not change again, we view the system as having entered a Nash Equilibrium Point and calculate the 'Price of Anarchy'. Figure 2 shows the probability density function of the 'Price of Anarchy' for different cost function orders. The results show that density near price 1 is quite large, which confirms the efficiency of our algorithm. Also, the relationship between the 'Price of Anarchy' and cost function order can be observed: distribution with a higher order has a longer tail.

B. *Numerical Results of the Regret*

In this part we demonstrate the simulation results for regret order. We use the same simulation settings in part A. However, this time the link cost function is defined to be unknown and we use our algorithm 2 to estimate the link cost function while utilizing the network, which is called 'Exploration and Exploitation'. We run the algorithm 2 for many times and calculate the expected regret by averaging. That is why we are getting a smooth curve. Figure 3 shows the growing behavior of regret with time under different G selections. We divide regret by log(T) to view its convergence. We choose the G_b as the basic G based on the condition shown in **Theorem 4**. Actually, this condition is just an sufficient condition that leads to logarithmic growing of regret. In real simulation, we have chosen a basic G smaller than in **Theorem 4** but can still help the logarithmic growth hold.

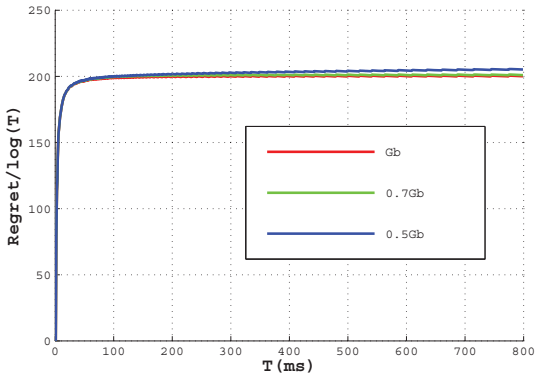

Fig. 3: regret(T) divided by log(T)

VIII. CONCLUSIONS

In this paper, we considered the flow scheduling problem under both known and unknown model. For the known model, we proposed a virtual non-cooperative game with incentive pricing to solve the cost optimization problem. To analyze this virtual game, we proved that our game always converges to a Nash Equilibrium Point which has a bounded price of anarchy. The constant bound was proved to be independent of network size and flow distribution. Then we extended this algorithm to situations when cost distributions were unknown apriori. We modeled this problem under multi-armed bandit model and combined the virtual game with the newly proposed DSEE Sequence which could achieve best regret for all light-tail cost distributions. It was proved that the regret of our algorithm was growing logarithmically with time if the DSEE

parameters were chosen properly, which is best in the convetinal online learning scenario. Finally, simulation results in terms of the Price of Anarchy and the regret growing behavior were provided to confirm the effectiveness of our algorithms in both two models.

ACKNOWLEDGEMENT

Prof. P. Fan's work was supported by the China Major State Basic Research Development Program (973 Program) No.2012CB316100(2), National Natural Science Foundation of China (NSFC) No. 61171064, the China National Science and Technology Major Project No.2010ZX03003-003 and NSFC No. 61021001.

REFERENCES

[1] K. Kar, M. Kodialam, T. V. Lakshman, "Minimum Interference Routing of Bandwidth Guaranteed Tunnels with MPLS Traffic Engineering Applications", *IEEE JSAC*, vol. 18, no. 12, pp. 2566-2579, December 2000.

[2] Mudi Kodialam T. V. Laksban, "Minimum Interference Routing with Applications to MPLS TraMic Engineering", *Proc.IEEE INFOCOM*, vol. 2, pp. 884-893, 2000.

[3] B. Fortz, M. Thorup, "Internet Traffic Engineering by Optimizing OSPF Weights", *Proc.IEEE INFOCOM*, vol. 2, pp. 519-528, 2000.

[4] R. Guerin, A. Orda, D. Williams, "QoS routing mechanisms and OSPF extensions", *Proc.IEEE Global Telecommunications Conference (GLOBECOM)*, vol. 3, pp. 1903-1908, 1997.

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

[6] J. B. Rosen, "Existence and Uniqueness of Equilibrium Points for Concave N-Person Games", *Econometrica*, vol. 33, no. 3, pp. 520-534, Jul. 1965.

[7] Baruch Awerbuch, Robert Kleinberg, "Online Linear Optimization and Adaptive Routing", *Journal of Computer and System Sciences*, vol. 74, no. 1, pp. 97-114, Feb. 2008.

[8] E. Koutsoupias, C. H. Papadimitriou, "Worstcase equilibria", *Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science*, pp. 404-413, 1999.

[9] B. Awerbuch, Y. Azar, and L. Epstein, "The price of routing Unsplittable flow", *Proc.37th Symp. Theory of Computing*, pp. 57-66, 2005.

[10] T. L. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules", *Advances in Applied Mathematics*, vol. 6, no. 1, pp. 422, 1985.

[11] R. Agrawal, "Sample mean based index policies with O(log(n)) regret for the multi-armed bandit problem", *Adv. Appl.Probab.*, vol. 27, no. 4, pp. 1054-1078, Dec. 1995.

[12] P. Auer, N. Cesa-Bianchi, and P. Fisher, "Finite time Analysis of the Multiarmed Bandit Problem", *Machine Learning*, vol. 47, no.2-3, pp.235-256, May, 2002.

[13] K. Liu and Q. Zhao, "Adaptive Shortest-Path Routing under Unknown and Stochastically Varying Link States", *Proc. of the 10th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, pp. 232-237, May, 2012.

[14] Y. Gai, B. Krishnamachari, and R. Jain, "Combinatorial Network Optimization with Unknown Variables: Multi-Armed Bandits with Linear Rewards and Individual Observations", *IEEE/ACM Transactions on Networking*, vol. 20, no. 5, 2012.

[15] K. Liu and Q. Zhao, "Distributed Learning in Multi-Armed Bandit With Multiple Players", *IEEE TRANSACTIONS ON SIGNAL PROCESSING*, vol. 58, no. 11, pp. 5667-5681, Nov. 2010.

[16] Dimitri P. Bertsekas and Robert G. Gallager, "Data Networks(2nd edition)", *Prentice Hall*, 1992.

[17] K. Liu and Q. Zhao, "Multi-Armed Bandit Problems with Heavy-Tailed Reward Distributions", *Proc. of Allerton Conference on Communications, Control, and Computing*, pp. 485-492, Sep. 2011.

[18] Y. Yang, K. Liu, Q.Zhao "Distributed Flow Scheduling in Unknown Environment", *Online Available:* http://arxiv.org/pdf/1210.1708.pdf