Lu PY, Yu CY. Worst-case Nash equilibria in restricted routing. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 27(4): 710–717 July 2012. DOI 10.1007/s11390-012-1257-5

Worst-Case Nash Equilibria in Restricted Routing

Pin-Yan Lu[1] (陆品燕), *Member*, *ACM*, and Chang-Yuan Yu[2] (余昌远)

[1]*Microsoft Research Asia, Beijing 100080, China*

[2]*Baidu. Inc, Beijing 100085, China*

E-mail: pinyanl@microsoft.com; yuchangyuan@baidu.com

Received March 7, 2011; revised January 15, 2012.

Abstract We study the network routing problem with restricted and related links. There are parallel links with possibly different speeds, between a source and a sink. Also there are users, and each user has a traffic of some weight to assign to one of the links from a subset of all the links, named his/her allowable set. The users choosing the same link suffer the same delay, which is equal to the total weight assigned to that link over its speed. A state of the system is called a Nash equilibrium if no user can decrease his/her delay by unilaterally changing his/her link. To measure the performance degradation of the system due to the selfish behavior of all the users, Koutsoupias and Papadimitriou proposed the notion Price of Anarchy (denoted by PoA), which is the ratio of the maximum delay in the worst-case Nash equilibrium and in an optimal solution. The PoA for this restricted related model has been studied, and a linear lower bound was obtained. However in their bad instance, some users can only use extremely slow links. This is a little artificial and unlikely to appear in a real world. So in order to better understand this model, we introduce a parameter for the system, and prove a better Price of Anarchy in terms of the parameter. We also show an important application of our result in coordination mechanism design for task scheduling game. We propose a new coordination mechanism, *Group-Makespan*, for unrelated selfish task scheduling game with improved price of anarchy.

Keywords routing, Nash equilibria, price of anarchy

1 Introduction

Network routing is one of the most important problems in the network management. In most networks, especially in a large-scale network like Internet, it is unlikely that there is a centralized controller who can coordinate the behavior of all the users in the network. In such situations, every user in the network decides how to route his/her traffic, aware of the congestion caused by other users. Users only care about the delay they suffer, and their selfish behavior often leads the whole network to a suboptimal state. Recently, researchers start to investigate the performance degradation due to the lack of the coordination for the users.

In the model first studied by Kautsoupias and Papadimitriou (KP model)[1], there are m identical parallel links from the same origin to the same destination. There are n users, and each with a traffic of weight w_i. We assume that each user's traffic cannot be split and as a result each user chooses exactly one link. After all the users choose their links, the delay of a link is equal to the total weight of the traffics on it, and the delay a user suffers is equal to the delay of the link he/she chooses. The performance of the system we consider here is the maximum delay of all the links. We are mainly interested in stable states, where no user can decrease his/her delay by unilaterally changing his/her choice. In game theory, such a state is also called a Nash equilibrium. In order to measure the performance degradation, they compared the performance of Nash equilibrium with the optimal solution when there is centralized coordination. In particular, we analyze the Price of Anarchy (PoA for short) of the system, which is defined to be the performance ratio between the worst-case Nash equilibrium and an optimal solution. In [1], Kautsoupias and Papadimitriou showed that the PoA of that system is at most $2 - 1/m$.

Since then, a lot of research works have been done along this line. There are mainly two generalized models of this problem which are well studied. One model is routing with related links, where different links may have different speeds and the delay of a link is equal to the total weight on this link over its speed. In this uniform related model, Czumaj and Vöcking proved that

Regular Paper

A preliminary version of this paper appeared in WINE 2008. Work was done when the authors were graduate students at Tsinghua University.

the PoA is $\Theta(\frac{\log m}{\log\log m})$[2]. The other model is routing with restricted links, where each user i is only allowed to choose links from a subset S_i of all the links. However the links are still identical in the sense that the speed of each link is the same. In this restricted model, Awerbuch *et al.* proved that the PoA is also $\Theta(\frac{\log m}{\log\log m})$[3].

In light of these results, one may conjecture that the common extension of these two models, where the links are both related and restricted, also has a PoA of $\Theta(\frac{\log m}{\log\log m})$. In fact, this model was studied by Gairing *et al.* in [4], and they showed that the PoA of this problem can be as large as $m-1$. However, in their bad instance demonstrating the lower bound of $m-1$, some users can only use extremely slow links (with speed less than $\frac{s_{\max}}{(m-1)!}$, where $s_{\max}$ is the largest speed). This is a little artificial and unlikely to appear in a real world. So in order to better understand this model, we introduce a property called λ-goodness for the structure of the users' link sets. An instance is called λ-good if and only if every user can at least use a link with speed no less than $\frac{s_{\max}}{\lambda}$. Now in our notation, the result in [4] says that the PoA can be as large as $m-1$ when the system is only $(m-1)!$-good. So what is the exact relation between the PoA and the λ-goodness of a system?

In this paper, we answer this question completely by giving a tight bound for the PoA of a λ-good system in term of λ. This reflects the exact relation between the PoA and the λ-goodness of the system. We prove that for λ-good instances, the PoA is $\Theta\big(\frac{\log \lambda m}{\log\log \lambda m}\big)$, as long as $\lambda \leqslant (m-1)!$. Formally, we have the following theorem.

Theorem 1. *For λ-good instances, the price of anarchy is $\Theta\big(\min\{\frac{\log \lambda m}{\log\log \lambda m}, m\}\big)$.*

Another brightness in this work lies on our technique used in the proof. In the proof of Czumaj and Vöcking for related links, they essentially used the property that the links are uniformly related, which means that each link has a fixed speed and all the users can choose it. And in the proof of Awerbuch *et al.* for restricted links, they essentially used the property that the links are identical, which means that all the links have the same speed. In our extended model, namely restricted related links, none of the two properties holds and as a result none of their techniques can be adopted to analyze the PoA of the new model directly. In this paper, we use a new proof approach. We calculate the delay of links interval by interval, obtain some recursive relations between them based on the property of Nash equilibrium, and finally we are able to derive a bound of the maximum delay in the system.

Our result also has an important application in task scheduling game with coordination mechanism. Task scheduling can be viewed as another model for routing problem by treating the links as machines, the traffics as tasks, the delay of a user as the completion time of his/her task, and the delay of the system as the makespan of the system. Then we have scheduling with identical machines, related machines, and restricted machines corresponding to the above three models of routing problems. Furthermore, we also have a general model, called scheduling with unrelated machines, in which each machine may have different speeds for different tasks. An instance of scheduling unrelated machines is denoted by a matrix $\boldsymbol{t} = (t_{ij})$, where t_{ij} denotes the processing time that machine j needs for task i. In this language, when each machine uses the Makespan policy, i.e., to process its tasks in such a parallel way that all of them are completed at the same time, the task scheduling game is essentially the same as the routing problem. However, as observed by Christodoulou, Koutsoupias, and Nanavati in [5], the scheduling policies of the machines may affect the choices of the users, and hence the PoA of the system. So they considered the problem of designing a set of local scheduling policies such that the PoA of the system is small. Such a set of scheduling policies are called coordination mechanism, and the PoA of the system with a coordination mechanism is also called the PoA of this mechanism.

Given our Theorem 1, we directly know that the PoA of Makespan mechanism for λ-good restricted related instances is $\Theta\big(\min\{\frac{\log \lambda m}{\log\log \lambda m}, m\}\big)$. Furthermore, using our main result, we propose a new coordination mechanism, named Group-Makespan mechanism, for scheduling unrelated machines. This Group-Makespan mechanism ensures the existence of a pure Nash equilibrium and its PoA is $O\big(\frac{\log^2 m}{\log\log m}\big)$, improving the best known result $O(\log^2 m)$ by Azar, Jain and Mirrokni in [6].

Theorem 2. *The Group-Makespan mechanism for scheduling m unrelated machines ensures the existence of pure Nash equilibria, and the PoA of the task scheduling game with this mechanism is $O\big(\frac{\log^2 m}{\log\log m}\big)$.*

Now we talk about the high level ideas of our new mechanism. This new mechanism is inspired by the mechanism Split&Shortest in [6]. However, we change some languages to fit in our framework well. Given an instance $\boldsymbol{t}$ for scheduling with unrelated machines, we can define $t_i = \min_{j\in[m]} t_{ij}$ as the weight of task i, and define the speed s_{ij} of a machine j with respect to a task i as $s_{ij} = t_i/t_{ij}$, namely the minimum running time of task i on all the machines over the running time of task i on machine j. In our Group-Makespan mechanism, every machine simulates $\log m$ submachines and submachine k of machine j only runs those tasks i for which machine j has speed $s_{ij} \in [2^{-k}, 2^{-k+1})$. We artificially delay a task so that the k-th submachines of different machines all have fixed speed 2^{-k}. Each machine simulates its submachines by round-robin, and for each submachine we use the Makespan scheduling policy. In the submachine level, each submachine has a fixed speed, and a task can only be assigned to some of the submachines. So it becomes a problem of scheduling with restricted related machines. Furthermore, all the instances obtained in this way have a very good structure, namely they are 1-good. Therefore in the submachine level, the PoA is bounded by $\Theta\big(\frac{\log m}{\log\log m}\big)$. Since each machine has to simulate $\log m$ machines all the time, this may loss a factor of at most $\log m$.

1.1 Related Work

The line of work on studying PoA of routing problem was first initiated by Koutsoupias and Papadimitriou in [1]. In their paper, they showed an exact ratio of $\frac{3}{2}$ for two identical links, and a lower bound of $\frac{1+\sqrt{5}}{2}$ for any two links with possible different speeds. For m identical links, they also showed a lower bound of $\Omega\big(\frac{\log m}{\log\log m}\big)$ and an upper bound of $3+\sqrt{4m\ln m}$. For the related case, they showed an upper bound of $O\Big(\sqrt{\frac{s_1}{s_m}\sum_{j\in[m]}\frac{s_j}{s_m}\log m}\Big)$, where s_j is the speed of link j, and the links are ordered so that $s_1 \geqslant s_2 \geqslant \cdots \geqslant s_m$.

Czumaj and Vöcking[2], Awerbuch *et al.*[3] also proved a tight bound of $\Theta\big(\frac{\log m}{\log\log m}\big)$ for the related, restricted routing models respectively in mixed strategies case. Gairing *et al.*[4] also gave a comprehensive collection of bounds on the PoA for several special cases of the restricted routing model in pure Nash equilibria case.

There is also another approach in studying the routing problem. In the KP model, the network only consists of m-parallel links between two nodes, and the users are atomic. In another model, the underlying network can be a general network, but with nonatomic users, which means that each user only controls a negligible fraction of the total traffic. Furthermore, each edge in the network is associated with a delay function, which defines the common delay incurred to all the users who select this edge, as a function of the congestion in this edge. This routing model dates back to Wardrop[7], and is also known as Wardrop model. The PoA of this model is first studied by Roughgarden and Tardos[8], who proved that the price of anarchy in networks with linear edge delay functions is precisely 4/3. After that, a lot of work appears in this model with different delay functions[9-12].

Coordination mechanism was first studied by Christodoulou, Koutsoupias and Nanavati in [5], and there have been a lot of results in this area. Several simple scheduling policies are well studied, such as the Shortest/Longest First policy, in which each machine performs the jobs in the non-decreasing/non-increasing order of their processing times on this machine. Most of the previously best results about PoA in coordination mechanism for job scheduling game are listed in Table 1 (the main part of this table appears in [13]).

The Randomized policy is for each machine to run tasks on this machine in a random order. In the InefficiencyBased mechanisms, each task i's inefficiency e_{ij} is defined as $\min_{k\in[m]} t_{ik}/t_{ij}$, and each machine schedules its jobs based on their inefficiencies. The result of $\Theta(\log m)$ marked by $*$ in the column of InefficiencyBased means that this mechanism does not always possess a pure Nash equilibrium. This is demonstrated by an example in [6]. In their work, they modify this mechanism so that Nash equilibrium always exists, however the PoA loses a factor of $\log m$.

2 Preliminaries and Notations

In this section, we define our problem formally. There are m independent links from certain origin to destination, and n independent users. We use $[m]$ and $[n]$ to denote the link set $\{1,\ldots,m\}$ and user set $\{1,\ldots,n\}$ respectively. Each link $j\in[m]$ has a speed s_j and w.l.o.g, we assume $s_1 \geqslant s_2 \geqslant \cdots \geqslant s_m$. Each user $i\in[n]$ has a traffic of weight w_i, which can only be assigned to a link from a set $S_i \subseteq [m]$. We use $\langle w, s, \mathcal{S}\rangle$ to denote an instance of the problem, where $w = (w_1,\ldots,w_n)$, $s = (s_1,\ldots,s_m)$ and $\mathcal{S} = \{S_1,\ldots,S_n\}$ denote the weights, speeds and allowable link sets. We introduce the property of λ-goodness for an instance $\langle w, s, \mathcal{S}\rangle$.

Definition 1 (λ-Goodness). *An instance $\langle w, s, \mathcal{S}\rangle$ is λ-good if and only if the following condition holds: for any user $i\in[n]$, there exists a machine $j\in S_i$ such that the speed s_j is at least s_1/λ.*

Table 1. Summary of Results

	Makespan	ShortestFirst	LongestFirst	Randomized	InefficiencyBased
$P\|\|C_{\max}$	$\frac{2m}{m+1}$[14-15]	$\frac{2m}{m+1}$[14-15]	$\frac{4}{3}-\frac{1}{3m}$[5]	$2-\frac{2}{m}$[14-15]	
$Q\|\|C_{\max}$	$\Theta(\frac{\log m}{\log\log m})$[2]	$\Theta(\log m)$[13,16]	$> \frac{4m-1}{3m}$ $< \frac{2m}{m+1}$ [13]	$\Theta(\frac{\log m}{\log\log m})$[2]	
$B\|\|C_{\max}$	$\Theta(\frac{\log m}{\log\log m})$[14]	$\Theta(\log m)$[13,16]	$\Theta(\log m)$[13,17]	$\Theta(\frac{\log m}{\log\log m})$[18]	
$R\|\|C_{\max}$	Unbounded[15]	m[6,13]	Unbounded	$\Theta(m)$[13]	$\Theta(\log m)^*$ [6] $\Theta(\log^2 m)$

We consider pure strategies for users, and each user's strategy is to decide which link to assign his/her traffic. We use $a = (a_1, \ldots, a_n) \in S_1 \times \cdots \times S_n$ to denote a combination of all users' strategies, where user i selects a link $a_i \in S_i$. We also use a_{-i} to denote the strategies of all the other users except user i. In a state a, the delay of link j, denoted by l_j^a, is the total weights on it over its speed, and the delay of the system, denoted by l^a, is the maximum delay over all the links. That is

$$l_j^a = \frac{1}{s_j} \sum_{i:a_i=j} w_i, \quad l^a = \max_j l_j^a.$$

We consider the optimum when there is centralized coordination, that is, the minimal delay of the system over all the possible states. We use *opt* to denote the optimum as well as an optimal solution, so we have

$$opt = \min_{a \in S_1 \times \cdots \times S_n} l^a.$$

We assume the users are all non-cooperative and each one wishes to minimize his/her own cost, without any regard to performance the system. The cost of user i in a state a is the delay of link a_i and we use c_i^a to denote it. We have

$$c_i^a = l_{a_i}^a.$$

Now we define the Nash equilibriums of the system formally.

Definition 2 (Nash Equilibrium). *A state a is called a Nash equilibrium* (*NE for short*) *of the system if and only if no user can decrease his/her cost by unilaterally changing a link. That is, for any user $i \in [n]$, any strategy $a_i' \in S_i$ and $a' = (a_{-i}, a_i')$, we have $c_i^a \leqslant c_i^{a'}$.*

For any instance of the problem, pure Nash equilibrium always exists. The proof of this fact is using a quite common method with an elegant potential function, which is pointed out in several places (see [19] for example).

Theorem 3 (Existence of Nash Equilibrium) *For $\lambda \geqslant 1$ and any λ-good instance $\langle w, s, \mathcal{S} \rangle$, there exists an Nash equilibrium state a of it.*

To compare the performance of Nash equilibrium with optimum, we have the definition of Price of Anarchy.

Definition 3 (Price of Anarchy). *For instance of restricted routing problem, the Price of Anarchy* (*PoA for short*) *is defined as the performance ratio between the worst-case Nash equilibrium and the optimal solution. That is*

$$PoA = \max_{\substack{a \in S_1 \times \cdots \times S_n \\ a \text{ is an NE}}} \frac{l^a}{opt}.$$

And for any family of instances, its Price of Anarchy is defined to be the largest PoA among all its possible instances.

3 PoA of λ-Good Restricted Routing

In this section, we prove our main result Theorem 1. If $\lambda > (m-1)!$, Gairing *et al.* gave a tight bound $\Theta(m)$[4]. So in this section, we always assume $\lambda \leqslant (m-1)!$ and prove that the PoA of the system for λ-good instances is $\Theta\left(\frac{\log \lambda m}{\log \log \lambda m}\right)$. We prove the upper bound and lower bound in the following two subsections respectively.

3.1 Proof of the Upper Bound

Theorem 4 (Upper Bound). *Given any λ-good instance $\langle w, s, \mathcal{S} \rangle$ and a state $a \in S_1 \times \cdots \times S_n$ which is a Nash equilibrium, delay of the system l^a is at most $opt \cdot O\left(\frac{\log \lambda m}{\log \log \lambda m}\right)$.*

For notational simplicity, we scale the speeds and weights such that $s_1 = 1$ and $opt = 1$. We also define several notations used in the proof. For any $k \in \mathbb{R}^+$ and $j \in [m]$, let $W_j^k = \max\{l_j^a - k, 0\} \cdot s_j$ and $W^k = \sum_{j \in [m]} W_j^k$. Especially, we use $W_j = W_j^0$ to denote the total weight assigned to link j, and $W = W^0$ to denote the total weight of all the users. Fix an optimal solution *opt*, let O_j be the set of users assigned to link j in *opt*. We also define O_j^k to be the set of users who choose link j in *opt* and have cost at least k, that is, $O_j^k = \{i \in O_j, c_i^a \geqslant k\}$.

Our proof of the upper bound theorem comes from the following lemmas. In Lemma 1, we give an initial condition of W^k and this is the only point we use the condition that the instance is λ-good. Then Lemma 2 and Lemma 3 give recursive relations between W^ks, which basically says that W^k should increase significantly when k becomes small. So we can bound the total weight W from below in terms of makespan l^a and λ. And on the other hand, the total weight is bounded from above by m. Putting things together, we can bound l^a.

Lemma 1. *For any λ-good instance and any Nash equilibrium a, we have $W^{l^a-2} \geqslant \frac{1}{\lambda}$.*

Proof. Consider a link whose delay achieves l^a, say link j^*. Let i be a user on link j^*, and let link $j \in S_i$ has the maximum speed in S_i. Now if $j = j^*$, we have $l_j^a = l^a$. If $j \neq j^*$, since a is a Nash equilibrium, i cannot decrease his/her cost by changing from link j^* to link j. We have:

$$l^a = c_{j^*}^a \leqslant l_j^a + \frac{w_i}{s_j}.$$

As in the optimal solution, task i can only be assigned to a link from S_i, whose speed is at most s_j, we have $w_i/s_j \leqslant opt = 1$. Therefore, we have $l_j^a \geqslant l^a - 1$. So no matter whether $j = j^*$ or not, we have $l_j^a \geqslant l^a - 1$, hence

$$W^{l^a-2} \geqslant W_j^{l^a-2} \geqslant 1 \cdot s_j \geqslant \frac{1}{\lambda},$$

where the last inequality is because the instance is λ-good. □

Lemma 2. *For any Nash equilibrium a and $0 \leqslant k \leqslant l^a - 2$, we have $W^k \geqslant \frac{l^a}{l^a-(k+2)} W^{k+2}$.*

Proof. Firstly, we want to prove that

$$W_j^k \geqslant \sum_{i \in O_j^{k+2}} w_i.$$

If O_j^{k+2} is empty, then we are done. Otherwise, for any task $i \in O_j^{k+2}$, $c_i^a \geqslant k+2$, by the definition of Nash equilibrium, we have

$$k+2 \leqslant c_i^a \leqslant l_j^a + w_i/s_j \leqslant l_j^a + 1.$$

The last inequality is because that the task i is assigned to link j in *opt*. Therefore, $l_j^a \geqslant k+1$ and

$$W_j^k \geqslant 1 \times s_j \geqslant \sum_{i \in O_j} w_i \geqslant \sum_{i \in O_j^{k+2}} w_i.$$

Noticing that $\bigcup_j O_j^k = \{i : c_i^a \geqslant k\}$, we can bound W^k as follows:

$$\begin{aligned} W^k &= \sum_{j \in [m]} W_j^k \geqslant \sum_{j \in [m]} \sum_{i \in O_j^{k+2}} w_i \\ &= \sum_{i : c_i^a \geqslant k+2} w_i = \sum_{j : l_j^a \geqslant k+2} W_j. \end{aligned} \quad (1)$$

By the definition of W_j and W_j^{k+2}, for any j, $l_j^a > k+2$, we have:

$$W_j = \frac{l_j^a}{l_j^a - (k+2)} W_j^{k+2} \geqslant \frac{l^a}{l^a - (k+2)} W_j^{k+2}. \quad (2)$$

The last inequality is because the function $f(x) = \frac{x}{x-(k+2)}$ is monotone decreasing when $x > k+2$ and for all j, we have $l_j^a \leqslant l^a$.

So from (1) and (2), we have:

$$\begin{aligned} W^k &\geqslant \frac{l^a}{l^a - (k+2)} \sum_{j : l_j^a > k+2} W_j^{k+2} \\ &= \frac{l^a}{l^a - (k+2)} W^{k+2}. \end{aligned}$$ □

From Lemma 1 and Lemma 2, we have recursive relation about W^k and an initial condition. These ensure us to prove an upper bound on l^a, which is $O(\log \lambda m)$. There is a little gap between our expected bound. The reason is that in the above estimation in (2), we bounded all the l_j^a from above by l^a. This is a little weak since there cannot be too many links with large l_j^a. The following lemma uses a more careful estimation, and explores a recursive relation between W^k, W^{k+2}, and W^{k+4}, which helps us to obtain a better bound on l^a.

Lemma 3. *For any λ-good instance and any Nash equilibrium a, we have*

$$W^k \geqslant \frac{k+6}{4}(W^{k+2} - 2W^{k+4}).$$

Proof. First, we omit some links in the summation of the last term in (1), and have:

$$W^k \geqslant \sum_{j : l_j^a > k+2} W_j \geqslant \sum_{j : k+6 \geqslant l_j^a > k+2} W_j.$$

Now, the estimation occurred in (2) can be more tight: for any j, $k+6 \geqslant l_j^a > k+2$, we have

$$\begin{aligned} W_j &= \frac{l_j^a}{l_j^a - (k+2)} W_j^{k+2} && (3) \\ &\geqslant \frac{k+6}{k+6-(k+2)} W_j^{k+2} && (4) \\ &= \frac{k+6}{4} W_j^{k+2}. && (5) \end{aligned}$$

So, we can bound W^k as

$$\begin{aligned} W^k &\geqslant \frac{k+6}{4} \sum_{j : k+6 \geqslant l_j^a > k+2} W_j^{k+2} \\ &= \frac{k+6}{4}\Big(W^{k+2} - \sum_{j : l_j^a > k+6} W_j^{k+2}\Big). && (6) \end{aligned}$$

For $\forall j$, $l_j^a > k+6$, we have $W_j^{k+2} = (l_j^a - (k+2)) \cdot s_j$ and $W_j^{k+6} = (l_j^a - (k+6)) \cdot s_j$, hence

$$2W_j^{k+4} = W_j^{k+2} + W_j^{k+6}.$$

Using this equality, we can bound the negative term in (6) as follows:

$$\sum_{j : l_j^a > k+6} W_j^{k+2} \leqslant \sum_{j : l_j^a > k+6} 2W_j^{k+4} \leqslant 2W^{k+4}.$$

Substituting this into (6), and we finish the proof. □

Proof of Theorem 4. Let $k_0 = \lfloor \frac{l^a}{6} \rfloor$. For any $k \geqslant l^a - 2k_0 \geqslant \frac{2l^a}{3}$, we have:

$$\begin{aligned} W^k &\geqslant \frac{k+6}{4}(W^{k+2} - 2W^{k+4}) \\ &\geqslant \frac{k+6}{4}\Big(W^{k+2} - 2 \times \frac{l^a - (k+4)}{l^a} W^{k+2}\Big) \\ &= \frac{2(k+4) - l^a}{4l^a} \times (k+6) W^{k+2} \\ &\geqslant \frac{2\Big(\frac{2l^a}{3} + 4\Big) - l^a}{4l^a} \times \Big(\frac{2l^a}{3} + 6\Big) W^{k+2} \geqslant \frac{l^a}{18} W^{k+2}. \end{aligned}$$

The first inequality is by Lemma 3 and the second inequality is by Lemma 2.

So using this recursive relation and lemma 1, we have:

$$W^{l^a-2k_0} \geqslant W^{l^a-2} \times \left(\frac{l^a}{18}\right)^{\frac{l^a}{6}} \geqslant \frac{1}{\lambda} \times \left(\frac{l^a}{18}\right)^{\frac{l^a}{6}}.$$

Since $\forall j$, $s_j \leqslant s_1 = 1$, and $opt = 1$, we have $W \leqslant opt \cdot \sum_j s_j \leqslant m$. By $W \geqslant W^{l^a-2k_0}$, we have

$$\left(\frac{l^a}{18}\right)^{\frac{l^a}{6}} \leqslant \lambda m.$$

Since the solution to the equation $x^x = y$ is $x = \Theta(\frac{\log y}{\log\log y})$, we can obtain that l^a is at most

$$O\Big(\frac{\log \lambda m}{\log\log \lambda m}\Big).$$

3.2 Proof of the Lower Bound

In this subsection, we prove that our upper bound is also tight. First, we give a technique lemma, which is useful in expressing our lower bound. The proof is put into the Appendix.

Lemma 4.

$$\max\Big\{\frac{\log m}{\log\log m}, \frac{\log \lambda}{\log\log \lambda}\Big\} = \Omega\Big(\frac{\log \lambda m}{\log\log \lambda m}\Big).$$

Now we have the following lower bound for the PoA of λ-good instances.

Theorem 5 (Lower Bound). *For $\lambda < (m-1)!$, there exists a λ-good instance and a Nash equilibrium a of it, such that the delay of the system l^a is at least*

$$opt \cdot \Omega\Big(\frac{\log \lambda m}{\log\log \lambda m}\Big).$$

Proof. Firstly, since any 1-good instance is also λ-good, the lower bound $\Omega(\frac{\log m}{\log\log m})$ for PoA of related routing problem in [2] also holds. So given Lemma 6, it is enough to prove a lower bound of $\Omega(\frac{\log \lambda}{\log\log \lambda})$.

The following is a λ-good instance whose PoA is at least $\Omega(\frac{\log \lambda}{\log\log \lambda})$.

• There are m links and $m-1$ users. Let k be the largest integer such that $k! \leqslant \lambda$. (Note that $k = \Omega(\frac{\log \lambda}{\log\log \lambda})$ and $k \leqslant m-1$.)

• For link j, $1 \leqslant j \leqslant k+1$, $s_j = \frac{1}{(j-1)!}$ (we use the convention that $s_1 = 0! = 1$); for link $j > k+1$, $s_j = 1$. For user i, $i \leqslant k$, the weight is $w_i = s_i$; for user i, $i > k$, the weight is $w_i = s_{i+1}$.

• For user i, $i \leqslant k$, the allowed links set $S_i = \{i, i+1\}$; for user i, $i > k$, the allowed links set is $S_i = \{i+1\}$.

For this instance, we have:

• This instance is λ-good, since $s_{\max} = 1$ and $s_{\min} = \frac{1}{k!} \geqslant \frac{1}{\lambda}$.

• For this instance, the optimal solution *opt* is following: the user i goes to link $a_i = i$ if $i \leqslant k$, and to link $a_i = i+1$ if $i > k$. The optimal value is $opt = 1$.

• The state a, in which user i selects link $a_i = i+1$, $\forall i$, is a Nash equilibrium and the maximum delay l^a is k. Since for link i, $i > k$, only one user can select link i in any state. we do not need to consider them in the following analysis. In the state a, for $i \leqslant k$, user i selects link $i+1$, and link $i+1$ has delay $l^a_{i+1} = \frac{w_i}{s_{i+1}} = \frac{s_i}{s_{i+1}} = i$. If user i changes to link i, then his/her cost will also be $i - 1 + w_i/s_i = i$. So the state a is a Nash equilibrium, and the system's delay is $l^a = k$.

Therefore, this instance has a Nash equilibrium a such that the system's delay l^a is at least $k \cdot opt = opt \cdot \Omega(\frac{\log \lambda}{\log\log \lambda})$. This completes the proof. □

4 Application in Coordination Mechanism

In this section, we see an application in coordination mechanism design for selfish task scheduling game. Azar, Jain and Mirrokni[6] gave the state of art result for unrelated scheduling. They showed a coordination mechanism with price of anarchy at most $O(\log m)$, though it may not possess any Nash equilibrium. They further modified this mechanism so that Nash equilibrium always exists, and the price of anarchy is $O(\log^2 m)$.

Our work focuses on restricted related routing, which corresponds to the makespan policy in restricted related scheduling game. Surprisingly, our result has an application in unrelated scheduling ($R||C_{\max}$), which is more general. We improve the above result by a mechanism with PoA of $O\Big(\frac{\log^2 m}{\log\log m}\Big)$. The overall idea is that we transform the given instance to a new one, which is restricted and related, and simply apply the Makespan mechanism on the new instance. Our Group-Makespan mechanism is adopted from the inefficiency-based mechanism in [6]. To fit our framework, we use some different language and notations. For each task i, let $t_i = \min_{j\in[m]} t_{ij}$ denote the weight of task i, and we define $s_{ij} = \frac{t_i}{t_{ij}}$ to be the speed of machine j for task i (this corresponds to the notation *inefficiency* introduced in [6]). Let $K = \lceil \log m \rceil + 1$.

Group-Makespan Mechanism. For each machine j, its scheduling policy is as follows:

1) For any task i assigned to machine j, if $s_{ij} < \frac{1}{m}$, refuse to perform this task, or equivalently, delay this task for infinite time.

2) Divide the tasks assigned to machine j into K groups. If $s_{ij} \in [2^{-k}, 2^{-k+1})$, $k = 0, 1, 2, \ldots, K-1$, put task i into group k. Note that the group 0 consists of

the tasks which have $s_{ij} = 1$.

3) Simulate K submachines. The simulation here means that the machine divides the time slots into equal tiny pieces, and use one time slot for simulating every submachine one by one. We remark that, even some submachines already finished, the machine still uses time slots for those submachines.

4) For each submachine k, since it has speed for all the tasks assigned to it in $[2^{-k}, 2^{-k+1})$, we can make this submachine have a fixed speed 2^{-k} by artificially rounding the running time of task i to $\widetilde{t}_{ij} = 2^k t_i$. This rounding can be achieved by delaying the tasks when processing them, and pretending their processing time is after rounding. All the submachines run the tasks using the Makespan policy with respect to the rounded processing time.

We firstly present another view of the mechanism. Given an instance $\boldsymbol{t} = (t_{ij})$ of the unrelated scheduling problem $R||C_{\max}$, we create a corresponding instance of those jobs to mK machines as follows: we label the machine corresponding to the k-th submachine of machine j as j_k. If a task i has running time t_{ij}, and will be put on the k-th submachine by machine j, then let the running time of task i on machine j_k be $K\widetilde{t}_{ij_k}$, otherwise it has running time $+\infty$. All the machines run the tasks in a makespan way.

Lemma 5. *Given an instance to the $R||C_{\max}$ problem and its corresponding instance on mK machines, we have*

1) *The optimum of the new instance is at most $4K$ times the optimum of the original instance.*

2) *Given an allocation for the new instance on the mK machines, we can get an allocation for the original instance with the same completion time for each task in our mechanism. In the other direction, given an allocation for the original instance in our mechanism, we can also get an allocation for the new instance with the same completion time for each task.*

Proof. The proof is similar as in [6] and we give the proof here to be self-contained.

1) Given any allocation a for the original instance on the m machines, we create an allocation for the corresponding instance on mK machines in two steps. Firstly, we put all the task i where $s_{ia_i} < \frac{1}{m}$ to the machines where they have the least running time, and obtain an allocation a'. Denote the set of these tasks by I. Then, the makespan can only increase additively by at most $\sum_{i\in I} t_i$. So we have:

$$l^{a'} \leqslant l^a + \sum_{i\in I} t_i \leqslant l^a + \frac{1}{m}\sum_{i\in I} t_{ia_i} \leqslant 2l^a.$$

Next, we create from an allocation a'' from a'. Assign the tasks on machine j in a' to the machine j_k if it will be put on the k-th submachine by machine j. The load will not increase since the tasks are split among K submachines. But we round the running time of tasks on each submachine and scaled it up by a factor K, which means the completion time of any task may be scaled up by a factor at most $2K$. By applying the two steps, we create an allocation a'' with the makespan which is at most increased by a factor of $4K$. In particular, the optimal solution for the original instance will be modified to a feasible solution for the new instance with the makespan increased by a factor of at most $4K$. Therefore, the optimum for the new instance is at most $4K$ times the optimum for the original instance.

2) This correspondence is easy. The tasks being assigned to machine j_k correspond to being assigned to machine j. Since our mechanism shares the time slots evenly between K submachines all the time. This just kills the K factor in the new instance. Both in the new instance and in our mechanism, the rounding exists. Therefore, the corresponding allocations have the same cost for each task. In particular, the makespan does not change. □

Now we prove that this new mechanism has a better PoA, which is Theorem 2.

Proof of Theorem 2. We firstly notice that the makespan policy always possesses Nash equilibria. So given any instance on m machines of our problem, there exists an allocation a which defines a Nash equilibrium in the corresponding instance on mK machines. By the above lemma, this allocation corresponds to an allocation a' in our mechanism, and the makespan does not change. We claim that a' still defines a Nash equilibrium, since if any task i can decrease its cost by changing machines in our mechanism, it can also do this in the new instance, which is impossible. Similarly, any allocation defining a Nash equilibrium in our mechanism also corresponds to an allocation in the new instance on mK machines.

Now, fix any allocation a defining a Nash equilibrium in our mechanism, we obtain an allocation a' for the new instance on mK machines. We have $l^a = l^{a'}$. Since in the new instance, each machine j_k has a fixed speed of 2^{-k}, and each task i can be processed on a subset of machines. So this is a restricted related scheduling case. Now, each task's allowed machines set contains a fastest machine. Therefore, this new instance can be viewed as a 1-good instance of the restricted related routing problem. Applying Theorem 4, we have

$$l^a = l^{a'} \leqslant O\Big(\frac{\log(mK)}{\log\log(mK)}\Big)\,opt',$$

where opt' is the optimum of the new instance on mK machines. Substituting $opt' \leqslant 4K\,opt$ and $K =$

$\lceil \log m \rceil + 1$ into the above inequality, we prove the theorem.

$$l^a \leqslant opt \cdot O\Big(\frac{\log^2 m}{\log\log m}\Big). \qquad \square$$

5 Conclusions and Open Problems

In this paper, we study the restricted related routing problem, a common extension model of two previous generalized models. We define a property called λ-goodness for the users' allowed links sets, and characterize the relation between this property and the price of anarchy of the system. In particular, we prove that any λ-good instance has PoA at most $O\big(\frac{\log \lambda m}{\log\log \lambda m}\big)$. We also construct a λ-good instance and a Nash equilibrium state a, whose maximum delay is $opt \cdot \Omega\big(\frac{\log \lambda}{\log\log \lambda}\big)$. Combining this lower bound with the lower bound of $\Omega\big(\frac{\log \lambda}{\log\log \lambda}\big)$ for related routing in [2], we prove that our upper bound of PoA for λ-good instances is tight.

As an important application, we use this main result and design a new coordination mechanism for the job scheduling game in the unrelated case. Our mechanism has PoA at most $O\big(\frac{\log^2 m}{\log\log m}\big)$, which improves the best previous result $O(\log^2 m)$ in [6].

The major open problem left in coordination mechanism is whether we can design a coordination mechanism with constant PoA, or whether we can achieve the PoA $O(\log m)$ with a coordination mechanism which promises the existence of Nash equilibrium.

References

[1] Koutsoupias E, Papadimitriou C H. Worst-case equilibria. In *Proc. the 16th STACS*, February 1999, pp.404-413.
[2] Czumaj A, Vöcking B. Tight bounds for worst-case equilibria. In *Proc. the 13th SODA*, January 2002, pp.413-420.
[3] Awerbuch B, Azar Y, Richter Y *et al.* Tradeoffs in worst-case equilibria. *Theor. Comput. Sci.*, 2006, 361(2): 200-209.
[4] Gairing M, Lucking T, Mavronicolas M, Monien B. The price of anarchy for restricted parallel links. *Parallel Processing Letters*, 2006, 16(1): 117-132.
[5] Christodoulou G, Koutsoupias E, Nanavati A. Coordination mechanisms. *Theor. Comput. Sci.*, 2009, 410(36): 3327-3336.
[6] Azar Y, Jain K, Mirrokni V. (Almost) optimal coordination mechanisms for unrelated machine scheduling. In *Proc. the 19th SODA*, January 2008, pp.323-332.
[7] Wardrop J G. Some theoretical aspects of road traffic research. *Proc. Inst. Civil Engineers*, 1952, Pt. II, 1: 325-378.
[8] Roughgarden T, Tardos É. How bad is selfish routing? In *Proc. the 41st FOCS*, November 2000, pp.93-102.
[9] Roughgarden T, Stackelberg scheduling strategies. In *Proc. the 33rd STOC*, July 2001, pp.104-113.
[10] Roughgarden T. The price of anarchy is independent of the network topology. In *Proc. the 34th STOC*, May 2002, pp.428-437.
[11] Roughgarden T, Tardos É. Bounding the inefficiency of equilibria in nonatomic congestion games. Technical Report TR 2002-1866, Cornell University, 2002.
[12] Schulz A S, Moses N S. On the performance of user equilibria in traffic networks. In *Proc. the 43th SODA*, January 2003, pp.86-87.
[13] Immorlica N, Li L, Mirrokni V, Schulz A. Coordination mechanisms for selfish scheduling. In *Proc. WINE*, December 2005, pp.55-69.
[14] Finn G, Horowitz E. A linear time approximation algorithm for multiprocessor scheduling. *BIT Numerical Mathematics*, 1979, 19(3): 312-320.
[15] Vredeveld T. Combinatorial approximation algorithms: Guaranteed versus experimental performance [Ph.D. Thesis]. Department of Mathematics and Computer Science, Eindhoven University of Technology, 2002.
[16] Aspnes J, Azar Y, Fiat A, Plotkin S, Waarts O. On-line routing of virtual circuits with applications to load balancing and machine scheduling. *J. ACM*, 1997, 44(3): 486-504.
[17] Azar Y, Naor J, Rom R. The competitiveness of on-line assignments. *Journal of Algorithms*, 1995, 18(2): 221-237.
[18] Gairing M, Lucking T, Mavronicolas M, Monien B. Computing Nash equilibria for scheduling on restricted parallel links. In *Proc. the 36th STOC*, June 2004, pp.613-622.
[19] Fotakis D, Kontogiannis S, Koutsoupias E, Mavronicolas M, Spirakis P. The structure and complexity of Nash equilibria for a selfish routing game. In *Proc. the 29th ICALP*, July 2002, pp.123-134.

Pin-Yan Lu is a researcher at Theory Group of Microsoft Research Asia. He studied in Tsinghua University and got his B.S. and Ph.D. degrees both in computer science in 2005 and 2009 respectively. He joined Microsoft Research Asia as an associate researcher in 2009. He is mainly interested in complexity theory, algorithms design and algorithmic game theory. Pin-Yan Lu is a member of ACM.

Chang-Yuan Yu is a project manager at Baidu Inc. He got his B.S. and Ph.D. degrees from Tsinghua University in 2005 and 2009 respectively, both in computer science. He is mainly interested in algorithms design and algorithmic game theory.

Appendix Proof of Lemma 4.

Proof. Since the function $\log x$ is a concave function, we have:

$$\log\Big(\frac{x_1+x_2}{2}\Big) \geqslant \frac{\log x_1 + \log x_2}{2}.$$

Then we have:

$$\begin{aligned}\frac{\log \lambda m}{\log\log \lambda m} &= \frac{\log m + \log\lambda}{\log\Big(\dfrac{\log m + \log \lambda}{2}\Big) + 1}\\ &\leqslant \frac{\log m + \log \lambda}{\dfrac{\log\log m + \log\log\lambda}{2} + 1}\\ &\leqslant 2\max\Big\{\frac{\log m}{\log\log m}, \frac{\log\lambda}{\log\log\lambda}\Big\}. \qquad \square\end{aligned}$$