International Journal of Smart Computing and Artificial Intelligence
International Institute of Applied Informatics
2019, Vol. 3, No. 2, –

Pruning Algorithms for A Replicator Dynamics Method in Multiple OD Selfish Routing Games

Guu Kofujita *, Satoshi Takahashi *

Abstract

A traffic flow allocation has been studied by many researches. This problem is treated by both urban planning research and game theoretical approaches. We stand on game theory to consider the traffic flow allocation problem by treating a class of congestion games on the network. The traffic flow allocation is called, in context the congestion game, a selfish routing game. In this game, our proposal is to find an equilibria of decision making of the players. The player's decision is amounts of flows for each origin-destination path. It is known that the equilibrium searching problem as edge based modeling is able to compute easily by using Frank-Wolfe method, however, the edge based model has weak expressiveness. Thus we employ a path based modeling that can treat some complex phenomena. In this model, since we need to handle many paths in a network, it is known that the equilibrium searching problem is difficult.

In this paper, we study a solving method for a multi OD selfish routing game, and a method for solving standard routing games and its high speeding method. Our algorithm employs a replicator dynamics which is one of iterative optimization techniques. In the solution based on the replicator dynamics, the calculation time is very large, since the calculation is also performed for all paths. Therefore, as a preprocessing of solving by replicator dynamics, the policy of the proposed method is to make computation time faster by deleting unused paths. This paper evaluates the algorithm by numerical experiment.

Keywords: multi-od selfish routing, pruning method, replicator dynamics.

1 Introduction

Many events can be designed and analyzed by a mathematical model using a network in real world systems such as transportation networks and social networks. Particularly, a traffic flow allocation has been studied by many researches. This problem is studied by both urban planning research and game theoretical approaches. We stand on game theory to consider the traffic flow allocation problem by treating a class of congestion games on the network. The traffic flow allocation is called, in context the congestion game, a routing game.

* University of Electro-Communications, Tokyo, Japan

The congestion game can be viewed by one of allocation problem. In the congestion game, players choose a subset of resources. Each resource has a cost function which depends on a number of users. Our goal is to find an equilibria of choice of the players. In the routing game, resources are a set of path which is from origin to destination. The routing game is applied to data transportation in the computer network and a traffic volume control of automobiles, for example, the road pricing problem in London and Singapore[1][2]. In this example, the governments impose tolls on vehicles traveling in a certain section in order to eliminate traffic jams in the city center. The routing game is a valid theory when setting the charging section and the charge amount in these cases.

The routing game can be formulated as a class of potential games[3]. By the property of the potential game, we can compute an equilibria in the selfish routing game. A standard formulation of the selfish routing game is given by Roughgarden[4]. Now many studies have been proposed[5][6]. In the routing game, there are two types of models. One is a single OD model, the other is multi OD model. In the single OD model, each player has the same route can-didate, whereas in the multi OD model, the OD for each player is different, it is not a candidate for the same route. Since the strategy space is not symmetrical, the description of the model becomes complicated.

Our contribution of this study is the following

- To extend a replicator dynamics method for a single OD routing game to a multiple OD routing game. This extension can treat a traffic control problem such as route design of inside of station.

- To propose two pruning algorithms for high speeding calculation of a replicator dynamics method for a multiple OD routing game.

The rest of the paper is organized as follows. Section 2 introduces a selfish routing game. In Section 3, we introduce some examples of the selfish routing game and the price of anarchy. In section 4, we propose a speed up method to compute an equilibrium flow in the selfish routing game. In Section 5, we evaluate our method by numerical experiments. After that we remark our research.

2 Selfish routing game

2.1 Model

The selfish routing game is a class of a non-cooperating game in which each player chooses some paths and its flows between a pair of vertices on the graph. Route selection games are classified into two types of games, nonatomic and atomic, depending on whether or not one player can divide the flow into a plurality of paths and form a flow.

In this paper, we treat only nonatomic selfish routing game. In the following discussion, when describing it as a routing game, unless there is any special description, it means a nonatomic selfish routing game.

We define a selfish routing game Γ on a directed graph $G = (V,E)$ where V is a set of vertices and E is a set of directed edges. Let $N = \{1, \cdots, n\}$ be a set of players and $OD = \{(s_i,t_i) \in V \times V \mid i \in N, s_i \neq t_i\}$ be a set of OD pairs on the graph G. For each OD pair (s_i,t_i), $P_i \subseteq 2^E$ is a set of $s_i - t_i$ paths. The player $i \in N$ wish to pour the quantity of flow $X_i \in \mathbb{R}_+$ into the OD pair (s_i,t_i). Let $S_i = \{\boldsymbol{x} \in \mathbb{R}^{|P_i|} \mid \sum_{p \in P_i} x_p = X_i\}$ be a strategy set for a player i, $\mathscr{S} = S_1 \times S_2 \times \cdots \times S_n$ be a strategy space, and $\boldsymbol{s} \in \mathscr{S}$ be a strategy vector. A strategy vector $\boldsymbol{s}$ consists of elements which represent how much flow should pour to each path. Also let $c_e : \mathbb{R}_+ \to \mathbb{R}_+$ be a cost function of each edge $e \in E$. The selfish routing game is denoted by $\Gamma = (G,N,OD,\mathscr{S},\boldsymbol{c})$, where $\boldsymbol{c} = (c_e)_{e \in E}$.

Next we define a flow of the network. The flow of a path $p \in P_i$ of the player $i \in N$ is denoted as x_p^i which is an element of the strategy set S_i. Hence we denote a strategy vector as $\boldsymbol{s} = (\boldsymbol{x}^1, \boldsymbol{x}^2 \cdots, \boldsymbol{x}^n) \in \mathscr{S}$. Moreover, a flow of an edge $e \in E$ under the strategy $\boldsymbol{s}$ is denoted as

$$f_e(\boldsymbol{s}) = \sum_{i \in N} \sum_{p \in P_i : e \in p} x_p^i. \tag{1}$$

From the above definition, a unit cost of the path p is denoted as

$$\tilde{c}_p(\boldsymbol{s}) = \sum_{e \in p} c_e(f_e(\boldsymbol{s})). \tag{2}$$

We can denote an average cost of the player i under the strategy $\boldsymbol{s}$ as

$$C_i(\boldsymbol{s}) = \frac{1}{X_i} \sum_{p \in P_i} x_p^i \tilde{c}_p(\boldsymbol{s}). \tag{3}$$

An equilibrium flow is a feasible flow $\boldsymbol{s}$ satisfies all path cost c_p is the same for any path $p \in P_i$ of each player. In other words, the equilibrium flow is defined as follows.

Definition 1 (Chap.18, Def. 18.1 in [4])**.** *Let $\boldsymbol{s}$ be a feasible flow for the instance (G, N, OD, $\mathscr{S}$, $\boldsymbol{c}$), $\boldsymbol{s}$ is an equilibrium flow if, for every player $i \in N$ and every pair $p, \tilde{p} \in P_i$ with $x_p^i > 0$,*

$$c_p(\boldsymbol{s}) \leq c_{\tilde{p}}(\boldsymbol{s}). \tag{4}$$

From a Wardrop's two extremal principles[7], a path cost of each player is the same in the equilibrium flow. On the other hand, we can consider a minimum total cost flow called social optimum flow (system optimum assignment[8]). It is known the following property for the equilibria of the selfish routing game.

- Any instance $(G,N,OD,\mathscr{S},\boldsymbol{c})$ has at least one equilibrium flow.
- Suppose that $\boldsymbol{s}$ and $\tilde{\boldsymbol{s}}$ are different equilibrium flow of $(G,N,OD,\mathscr{S},\boldsymbol{c})$. For any edge $e \in E$, we have $c_e(f_e) = c_e(\tilde{f}_e)$.

The selfish routing game $\Gamma = (G, N, OD, \mathscr{S}, \boldsymbol{c})$ is characterized by a potential function which is a characteristic function[3]. The potential function of Γ is

$$\Phi(\boldsymbol{s}) = \sum_{e \in E} \int_0^{f_e(\boldsymbol{s})} c_e(y) dy. \tag{5}$$

We can compute the equilibrium flow by solving the following an optimization problem.

$$\text{(P)} \left| \begin{array}{lll} \min\limits_{\boldsymbol{s} \in \mathscr{S}} & \Phi(\boldsymbol{s}) & \\ \text{s.t.} & \sum\limits_{p \in P_i} x_p^i = X_i, & \forall i \in N \\ & x_p^i \geq 0, & \forall i \in N, \forall p \in P_i. \end{array} \right. \tag{6}$$

Now we have a Lagrange relaxation of the optimization problem (6).

$$\text{(LRP)} \left| \begin{array}{ll} \min\limits_{\boldsymbol{s} \in \mathscr{S}} & \Phi(\boldsymbol{s}) - \sum\limits_{i \in N} \phi_i \left(\sum\limits_{p \in P_i} x_p^i - X_i \right) \\ \text{s.t.} & x_p^i \geq 0, \ \forall i \in N, \forall p \in P_i. \end{array} \right. \tag{7}$$

We consider a partial differential of the objective function of (7) with x_p^i,

$$\frac{\partial \Phi(\boldsymbol{s})}{\partial x_p^i} - \phi_i, \ \forall i \in N. \tag{8}$$

Next, for each player $i \in N$, we have,

$$\begin{aligned} \frac{\partial \Phi(\boldsymbol{s})}{\partial x_p^i} &= \sum_{e \in p} c_e(f_e(\boldsymbol{s})) \\ &= \tilde{c}_p(\boldsymbol{s}). \end{aligned} \tag{9}$$

From the complementary condition of (7), we get

$$\left(\frac{\partial \Phi(\boldsymbol{s})}{\partial x_p^i} - \phi_i \right) = 0, \ \forall i \in N. \tag{10}$$

Moreover, we consider a dual problem of (7).

$$\text{(LRD)} \left| \begin{array}{ll} \max & \phi_i \\ \text{s.t.} & \dfrac{\partial \Phi(\boldsymbol{s})}{\partial x_p^i} - \phi_i \geq 0, \ \forall i \in N, \forall p \in P_i. \end{array} \right. \tag{11}$$

From the above discussion, the following formula is held on the equilibrium flow for each player $i \in N$:

$$\left\{ \begin{array}{ll} \dfrac{\partial \Phi(\boldsymbol{s}*)}{\partial x_p^i} = \phi_i & (x_p^i > 0), \\ \dfrac{\partial \Phi(\boldsymbol{s}*)}{\partial x_p^i} \geq \phi_i & (x_p^i = 0). \end{array} \right. \tag{12}$$

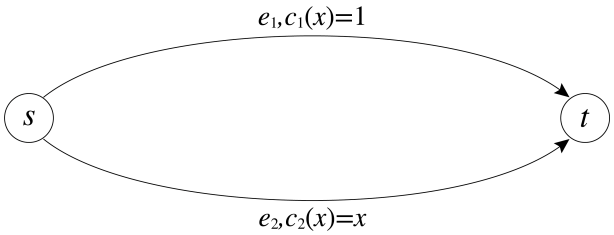


Figure 1: Pigou's paradox

Now we consider an average cost of each player on the solution $\boldsymbol{s}*$. If $x_p^i = 0$, the average cost is

$$\begin{aligned} C_i(\boldsymbol{s}*) &= \frac{1}{X_i} \sum_{p \in P_i; x_p^i > 0} x_p^i \phi_i \\ &= \phi_i, \end{aligned} \tag{13}$$

since $x_p^i \tilde{c}_p(\boldsymbol{s}*) = 0$. Therefore $\tilde{c}_p(\boldsymbol{s}*) = C_i(\boldsymbol{s}*)$ is held on a path with $x_p^i > 0$ in the equilibrium flow $\boldsymbol{s}*$. The other, $\tilde{c}_p(\boldsymbol{s}*) \geq \phi_i$ is hold on a path with $x_p^i = 0$. Thus it shows that the equilibrium flow $\boldsymbol{s}*$ satisfies the definition 1.

2.2 Price of Anarchy

There are two important flows in the selfish routing game, one is a social optimum flow and an equilibrium flow which establishes the same unit cost c_p $(\forall p \in P_i)$ of the player i. The social optimum flow can be computed by transforming to a minimum cost multi-flow. For an instance $(G, N, OD, \mathscr{S}, \boldsymbol{c})$, we consider an equilibrium flow $\boldsymbol{s}^* \in \mathscr{S}$ and an optimal flow $\tilde{\boldsymbol{s}} \in \mathscr{S}$. The total cost of the instance of each flow holds $C(\boldsymbol{s}^*) \geq C(\tilde{\boldsymbol{s}})$. We define a price of anarchy as follows:

$$\frac{C(\boldsymbol{s}^*)}{C(\tilde{\boldsymbol{s}})}. \tag{14}$$

It is known the following property of the price of anarchy[9].

Property 2. *When the edge cost is linear or constant, i.e.* $c_e(x) = ax + b, (a, b \geq 0)$, *the price of anarchy is at most* $\frac{5}{2}$.

2.3 Example of the selfish routing game

We consider two examples of the selfish routing game, one is Pigou's paradox and the other is Breass's paradox.

We show a Pigou's paradox in a figure 1. In this figure, there are two edges, e_1 and e_2. A cost of edge e_1 is 1 and e_2 has a linear cost function. We consider to pour a flow $X = 1$ from s to t.

Table 1: Flow and cost of equilibrium flow and optimum flow.

	e_1's flow	e_1's cost	e_2's flow	e_2's cost	Total cost
Eq. flow	0	1	1	1	1
Opt. flow	0.5	1	0.5	0.5	0.75

Table 2: Comparison of each cost

	Path cost	Total cost	Opt. total cost
Fig.2a	1.5	1.5	1.5
Fig.2b	2	2	1.5

From the table 1, we can see that each edge's cost is same in the equilibrium flow. The total cost of the equilibrium flow is 1. On the other hand, each edge's cost is not same in the optimum flow, however, the total cost is 0.75.

We consider the potential function of this example. Suppose that x_i and $c_i\,(i=1,2)$ are flow and cost, respectively.

$$\begin{aligned}\Phi(\boldsymbol{s}) &= \sum_{e\in E}\int_0^{x_e} c_e(y)dy \\ &= \int_0^{x_1} c_1(y)dy + \int_0^{x_2} c_2(y)dy \\ &= \int_0^{x_1} dy + \int_0^{x_2} ydy \\ &= x_1 + \frac{1}{2}x_2^2. \end{aligned} \tag{15}$$

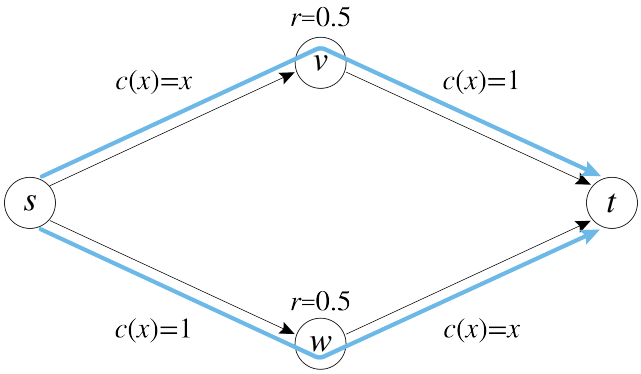
We compute a flow which minimizes this potential function. Since $x_1 + x_2 = 1$, we have

$$\begin{aligned} x_1 + \frac{1}{2}x_2^2 &= (1-x_2) + \frac{1}{2}x_2^2 \\ &= \frac{1}{2}(x_2-1)^2 + \frac{1}{2}. \end{aligned} \tag{16}$$

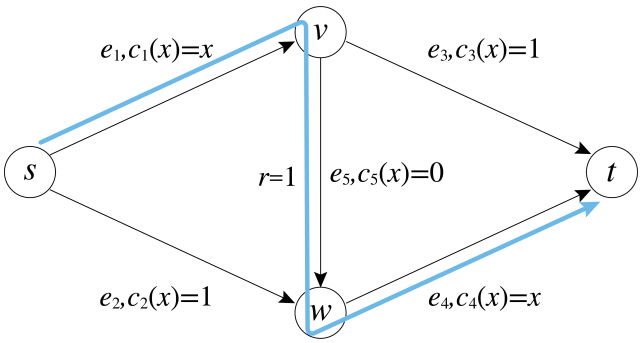
This result is the same as the equilibrium flow. Also a price of anarchy of this example is

$$\frac{C(\boldsymbol{s}^*)}{C(\tilde{\boldsymbol{s}})} = \frac{1}{0.75} = \frac{4}{3}. \tag{17}$$

The second example is Breass's paradox (figure 2a and 2b). From the table 2, the equilibrium flow and optimum flow is same in the initial network. On the other hand, after the adding an edge (v,w), the three paths $s\to v\to t$, $s\to v\to w\to t$, and $s\to w\to t$ has the same path cost 2. From this fact, we can see this flow is the equilibrium flow. However, the total cost is increasing from the initial network. We consider the potential function.



(a) Initial network.



(b) After the adding an edge (v,w)

Figure 2: Breass's paradox

Suppose that $\boldsymbol{s}^*$ is an equilibrium flow and $\tilde{\boldsymbol{s}}$ is an optimum flow. The values of potential functions are

$$\begin{cases} \Phi(\boldsymbol{s}^*) = 1.3, \\ \Phi(\tilde{\boldsymbol{s}}) = 1. \end{cases} \tag{18}$$

The potential function value of the equilibrium flow is smaller than the optimum flow's. Also the price of anarchy is 1 in the initial network, but after the adding, the value is

$$\frac{C(\boldsymbol{s}^*)}{C(\tilde{\boldsymbol{s}})} = \frac{2}{1.5} = \frac{4}{3}. \tag{19}$$

3 Replicator dynamics

In this research, we employ replicator dynamics method for equilibrium solution search[6]. The replicator dynamics is one of evolutionally computation methods[10][11] for solving a differential equation.

For the single OD model, speeding up method using replicator dynamics is already given[12]. Replicator dynamics is an iterative method that updates the flow rates in all conceivable routes until a condition of the equilibrium solution is satisfied by a defined equation. The detail of iteration is the follows.

In the $k+1$ iterations, update the player i's path p's flow,

$$x_p^i(k+1) = x_p^i(k) - \alpha x_p^i(k)(\tilde{c}_p(\boldsymbol{s}(k)) - C(\boldsymbol{s}(k))), \tag{20}$$

until the termination condition is held:

$$\begin{aligned} x_p^i &= 0, \\ \tilde{c}_p(\boldsymbol{s}^*) &= C(\boldsymbol{s}^*). \end{aligned} \tag{21}$$

In this method, it is required that an amount of flows on each OD pair satisfies a flow conservation law in each iteration of replicator dynamics. For example, we show a difference of flows between k th and $k+1$ st iteration as follows:

$$\begin{aligned} &\sum_{p\in P_i} \frac{x_p(k+1) - x_p(k)}{\alpha} \\ &= \sum_{p\in P_i} \{-x_p(k)\,(\tilde{c}_p(\boldsymbol{s}(k)) - C_i(\boldsymbol{s}(k)))\} \\ &= -\sum_{p\in P} x_p(k)\tilde{c}_p(\boldsymbol{s}(k)) + \sum_{p\in P} x_p(k)C(\boldsymbol{s}(k) \\ &= -XC(\boldsymbol{s}(k)) + XC(\boldsymbol{s}(k) \\ &= 0, \end{aligned} \tag{22}$$

where each path $p \in P_i$ holds $x_p(0) > 0$ in an initial solution. In an equilibrium solution, there are many paths such that the flow becomes zero. In the solution based on the replicator dynamics, the calculation time is very large, since the calculation is also performed for the route of the 0 flow every time. Therefore, as a preprocessing of solving by replicator dynamics, the policy of the proposed method is to make computation time faster by deleting unused paths. Yoshida et. al have been proposing a redundant path deletion method for searching an equilibrium flow in a single OD selfish routing game[12]. Our proposing method is an extension to multiple OD model.

3.1 Pruning method 1

To remove redundant paths in each path set P_i, we wish to set a criterion for path cost. This method accelerates an equilibrium search based on the replicator dynamics by removing some paths in P_i that satisfies the constant cost is greater than the criterion cost C_{cut}^i for each player i. We show a method 1 as algorithm 1.

We show that method 1 does not impair the completeness of the solution. It means that the algorithm does not delete paths with necessary to make an equilibrium flow.

Proposition 3. *Path deletion based on a criterion cost C_{cut}^i does not impair the completeness of the solution.*

Algorithm1 Pathremovingalgorithmbasedonmaximumflowonthenetwork.

Require: A directed graph $G = (V,E)$, n OD pairs $(s_1,t_1),(s_2,t_2),\cdots,(s_n,s_n)$, flows ofeachODpair$X_1,X_2,\cdots,X_n$,andeachedgecost$c$.

Ensure: Anequilibriumflow$s$ s^*.

1: Using Dijkstra method for computing a minimum cost of each terminal t_i fromevery vertex when we consider the constant cost. The estimated cost, from a vertex v to t_i is denoted by C^{min}_{v,t_i}.

2: UsingDijkstramethodforcomputingaminimumcostwhenwepour $\sum_{i2N} X_i$ *UPBO* OD pair (s_i,t_i). The estimated cost is denoted by C^i_{cut}.

3: ForeachODpair,decideasearchingpathbyusingdepthfirstsearch(DFS).InUIF DFS, when there are multi edges between two vertices, simultaneous use within the same route is not permitted. When DFS reached from s_i to a vertex$v \in V \setminus \{s_i\}$, denote the sum of constant cost as $c_{s_i,v}$. If $c_{s_i,v} + C^{min}_{v,t_i} > C^i_{cut}$ holds, DFS will stop searching.IfDFSreachedtot_i,addthes_i pathtoapathsetP^{use}_i.

4: LetP^{use}_i be thesetofroutesgivingtheinitialflowrateofeachODpair,searchthe equilibrium solutions^* bythereplicatordynamics,andoutputthesolution.

Proof. Let $\mathbf{s}^p$ be a strategy vector that pour all ow, i.e$X = \sum_{i2N} X_i$, to only the path p. Since each edge cost function is monotone increasing,thestrategy vector $\mathbf{s}^p$ establish a maximum cost of a path Thus

$$\tilde{c}_p(\boldsymbol{s}) \leq \tilde{c}_p(\hat{\boldsymbol{s}}^p) \forall i \in N, \forall p \in P_i, \forall \boldsymbol{s} \in \mathscr{S}. \tag{23}$$

5 I F S F G P S F G S P N B Q S P Q F S U ZU P F G U P M M R X J O H J C J S T J V F N M E P X T

$$C^i_{cut} = \tilde{c}_p(\hat{\boldsymbol{s}}^p) \geq \tilde{c}_p(\boldsymbol{s}^*) \forall i \in N, p \in P_i, \forall \boldsymbol{s} \in \mathscr{S}. \tag{24}$$

The criterion cost C^i_{cut} is always greater than the cost of the route used in the equilibrium solution, and the path deleted with C^i_{cut} never is used in the equilibrium solution. □

From the above it can be seen that the method of deleting a route using the deletion base cost C^i_{cut} does not impair the completeness of the solution.

3.2 Pruning method 2

In this method, in addition to the method 1, the route is deleted by considering the maximumflowrate of eachedge. Weshow amethod2asalgorithm2.

As same as method 1, we show that method 2 does not impair the completeness of the solution. It means that the algorithm does not delete paths with necessary to make an equilibrium flow.

Proposition 4. *Path deletion based on a criterion cost C^i_{cut} does not impair the completeness of the solution.*

Algorithm 2 Path removing algorithm based on maximum flow on each edge.

Require: A directed graph $G = (V,E)$, n OD pairs $(s_1,t_1),(s_2,t_2),\cdots,(s_n,s_n)$, flows of each OD pair $X_1,X_2,\cdots,X_n$, and each edge cost c.

Ensure: An equilibrium flow $\boldsymbol{s}^*$.

1: Using Dijkstra method for computing a minimum cost of each terminal t_i from every vertex when we consider the constant cost. The estimated cost, from a vertex v to t_i is denoted by C^{min}_{v,t_i}.
2: Using Dijkstra method for computing a minimum cost when we pour $\sum_{i\in N} X_i$ to an OD pair (s_i,t_i). The estimated cost is denoted by C^i_{cut}.
3: For each OD pair, decide a searching path by using depth first search (DFS). In the DFS, when there are multi edges between two vertices, simultaneous use within the same route is not permitted. When DFS reached from s_i to a vertex $v \in V \setminus \{s_i\}$, denote the sum of constant cost as $c_{s_i,v}$. If $c_{s_i,v} + C^{min}_{v,t_i} > C^i_{cut}$ holds, DFS will stop searching. If DFS reached to t_i, add the $s_i - t_i$ path to a path set P^{use}_i.
4: Let N_e be a set of players that use an edge $e \in E$ in the set of path P_i. For each edge $e \in E$, compute $X_e = \sum_{i\in N_e} X_i$.
5: For each path $p \in P_i$ of the player $i \in N$, compute $c^{e_{max}}_p = \sum_{e\in p} c_e(X_e)$. And for each player $i \in N$, compute $C^i_{cut} = \min_{p\in P_i} c^{e_{max}}_p$.
6: As same as step 3, make a path set P^{use}_i by using C^i_{cut}.
7: Let P^{use}_i be the set of routes giving the initial flow rate of each OD pair, search the equilibrium solution $\boldsymbol{s}*$ by the replicator dynamics, and output the solution.

Proof. Suppose that $\mathscr{S}_{cut}$ is a strategy space which contains only considering paths in P^{use}_i for every player $i \in N$. Since players using each edge $e \in E$ have already decided, the flow does not exceed $\sum_{i\in N_e} X_i$ in the edge e. Therefore $c^{e_{max}}_p$ is a maximum cost for each path after the first path deletion. Thus the following holds:

$$c^{e_{max}}_p \geq \tilde{c}_p(\boldsymbol{s}) \qquad \forall i \in N, p \in P_i, \boldsymbol{s} \in \mathscr{S}_{cut}. \tag{25}$$

Hence the equilibrium flow $\boldsymbol{s}^*$ satisfies

$$C^i_{cut} = \min_{p\in P^{use}_i} c^{e_{max}}_p \geq \tilde{c}_p(\boldsymbol{s}^*) \qquad \forall i \in N, p \in P_i. \tag{26}$$

From the above, the route deleted using the deletion base cost C^i_{cut} never is used in the equilibrium solution. □

From the above it can be seen that the method of deleting a route using the deletion base cost C^i_{cut} does not impair the completeness of the solution.

3.3 Discussion

The Frank-Wolfe algorithm is another method of solving general form of the selfish routing game and common method in application research of that game. The

computing time is mostly faster than our algorithm. However, the Frank-Wolfe algorithm use edges flow when solving. Because of using paths flow, our algorithm may solve some problems which cannot be solved existing algorithms like each edges have some kinds of costs.

4 Numerical experiment

We examine the equilibrium solution search method in the selfish routing game with multiple OD pairs using the proposed algorithm. In this experiment, we analyze the equilibrium solution search result in the grid network G. The cost function of each edge $e \in E$ is

$$\begin{aligned} c_e(\boldsymbol{s}) &= c_e^{cs} + c_e^{cgt}(\boldsymbol{s}), e \in E \\ c_e^{cs} &= a, 0.5 \leq a \leq 1.0 \\ c_e^{cgt} &= b \times f_e(\boldsymbol{s}), 0.5 \leq b \leq 1.0, \end{aligned}$$

where a, b is a random variable on the uniform distribution. Let P be a number of paths, $P*$ be a number of paths after deletion, and P^{use} be a number of paths that there is a positive flow in the equilibrium flow. The experiment environment is follows; (1)CPU: 3.2 GHz Intel Core i5, (2)Memory: 16 GB 1867 MHz DDR3, and (3)OS: macOS 10.12.6. We implemented by python 3.6.3.

4.1 Experiment 1

Verify the solution in a small scale grid network. Investigate the calculation cost of the equilibrium solution search by the proposed method 1 and the proposed method 2 and verify the resultant solution.

Here we will use a 4 × 4 size grid network. The combination of the number of OD pairs and the starting and ending points of OD pairs was changed and each experiment was carried out. The combination of the start and end points was set to be two types of the shortest route on the network and the one not intersecting with each other without considering the cost. The number of OD pairs was set to 2.

Experiments were carried out for each of the intersecting ones where the shortest path between the ODs when the cost is not considered and the intersecting ones respectively. The experiments were conducted on the same network for Method 1 and Method 2. The cost was changed and the average of the 10 results was obtained and taken as the obtained value. We show the average of 10 instances in the table 3,4 and number of paths in search shows in the fig 3,4.

Comparing the number of route deletes, it is reduced regardless of whether OD crosses or not. However, it turns out that the reduced rate is higher when it is parallel. On the other hand, when we compare the computation time, we can see that in the case of parallel, the method 2 exceeds the method 1, whereas in the intersection the calculation time is shorter than the deletion rate of the route. This is considered to change depending on the ratio of the time required for route deletion

Table 3: Experiment Results of paralell OD pairs in grid graph.

	Base Method	Method 1	Method 1+2
C_{cut}	-	6.426	6.426
$C(\boldsymbol{s}^*)$	-	4.067	4.0
P	736	127.800	127.800
C_{cut2}	-	-	4.527
P^2	-	-	87.500
P^{use}	-	3.600	3.600
Comp. time(msec)	-	6.108	2.727

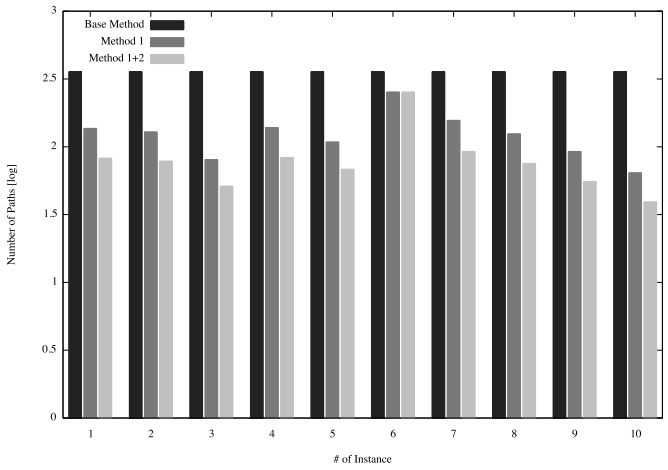


Figure 3: Number of paths in each instances.

to the solution search time by replicator dynamics. That is, when the number of routes is small, the time required for route deletion exceeds the computation time of replicator dynamics, resulting in exceeding the total computation time of method 1. It is considered that it is necessary to properly use methods 1 and 2 properly according to the number of candidate routes obtained in the initial search.

As for the result obtained from the equilibrium solution, both the number of utilization routes and the average cost are considered to be within the range of the error occurred after obtaining almost equal answers.

4.2 Experiment 2

In the following, we verify the number of candidate paths deleted, which is the key to speeding up in method 2. Using the grid network of 5×5 size, in addition to the setting made in the experiment 1, experiments were conducted for the cases where the number of OD pairs was 2 and 4, respectively. Since the difference in method 2 is only deletion of the candidate route, in this experiment, the equilibrium solution itself is not searched. The cost was changed on the same network and the average of the 100 results was taken as the obtained value. The obtained results are shown

Table 4: Experiment Results of interest OD pairs in Grid Graph.

	Base Method	Method 1	Method 1+2
C_{cut}	-	12.199	12.199
$C(\boldsymbol{s}^*)$	-	6.634	6.634
P	736	734.600	734.600
C_{cut2}	-	-	10.380
P^2	-	-	695
P^{use}	-	31.300	31.300
Comp. time(msec)	-	615.357	545.271

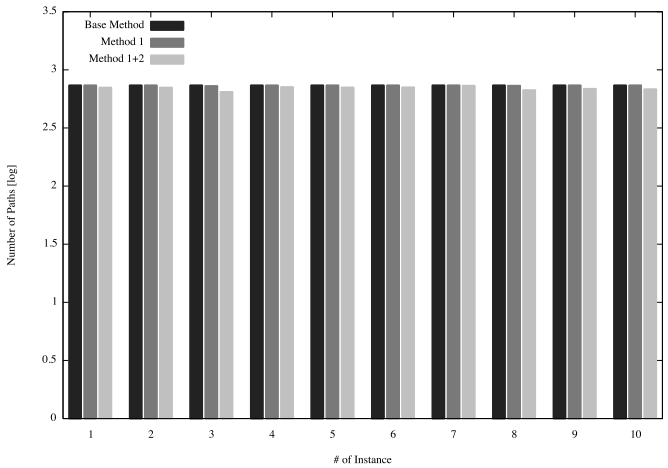


Figure 4: Number of paths in each instance.

Table 5: Result of experiment 2

	Parallel		Intersect	
# of ODs	2	4	2	4
P	534.650	45957.125	70893.835	290805.510
P^*	390.380	45957.125	68982.980	289377.693
$P-P^*$	144.27	0	1910.855	1427.817
$\frac{P^*}{P}$	0.730	1	0.973	0.995

in table 5.

In the 5×5 grid network it turned out that on average it cannot be deleted at all or that the deletion rate is small. When the OD which was able to delete was looking at the result of the intersect, it was found that the number of deletions greatly changed depending on the cost given. Moreover, even if the shape of the graph is the same, we also found that the result of route deletion greatly varies depending on cost. In future it is necessary to additionally verify in which case efficient path deletion is possible and what kind of properties exist depending on the shape of the graph.

Table 6: Experiment Results of random graph.

Method	Base Method	Method 1	Method 1+2
C_{cut}	-	10.815	10.815
$C(\boldsymbol{s}^*)$	-	6.739	6.739
P	3622.600	562.400	562.400
C_{cut2}	-	-	8.808
P^2	-	-	373.900
P^{use}	-	5.600	5.600
Comp. time(msec)	-	233.495	165.529

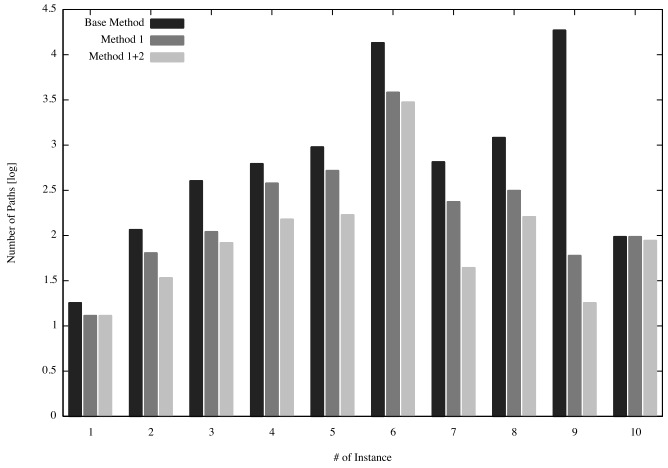


Figure 5: Number of paths in each instance.

4.3 Experiment 3

Next, we use another type of network. In this experiment, so many type of instances are applicated. We will use 10 random graphs made by 40 nodes and 0.05 density of edges. The setting of cost is same as before experiments. The difference of before experiments is the type of graph. In this experiment, topology of graphs are different. The average of 10 instances are shown in table 6 and number of paths in search shows in figure 5. Here we will use a random graph. It made by 40 nodes and the density of edges is 0.05.

From table 6, method 1 and 2 have reduced redundant paths. However from figure 6, there are large differences between instances. In addition, it can be seen that there are multiple cases such as those in which there is almost no difference between Method 1 and 1 + 2 and there is no difference between Base Method and Method 1. Since the difference for each instance is much larger than the result obtained in Experiment 1, it can be considered that the present method has a big difference depending on the nature of the instance.

Table 7: Experiment Results of JPN25.

Method	Base Method	Method 1	Method 1+2
C_{cut}	-	7.050	7.050
$C(\boldsymbol{s}^*)$	-	4.242	4.242
P	12076.400	6003.500	6003.500
C_{cut2}	-	-	5.607
P^2	-	-	1084.700
P^{use}	-	31.300	31.300
Comp. time(msec)	-	3672.217	382.704

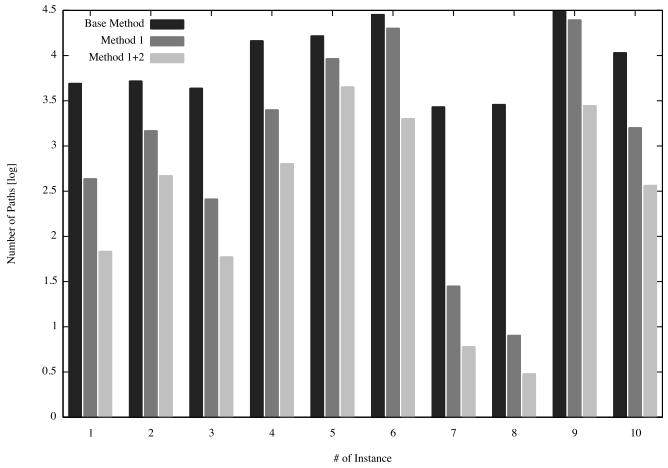


Figure 6: Number of paths in each instance.

4.4 Experiment 4

Finally, we use JPN25(Japan Photonic Network) Model[13] like Japan's backbone network made by 25 nodes. We made 10 network by different settings of each edge's cost. In this experiment, the effects of algorithm in real network are shown. The average of 10 instances are in 7 and number of paths in search shows in 6. The table ref Exp6 indicates the shortest path length as the number of edges of the OD pair used in each instance and whether each shortest path shares vertices.

Table ref Exp 5 shows that the number of reduction paths when applying each method is much larger than other experiments. JPN 25 handled this time has lower edge density than the instances dealt with in previous experiments and modeled Japanese cities, so it is because the reason is that the route between vertices is likely to be narrowed down when applying the method Conceivable. Looking at each instance, we can see that the shortest path length of both ODs is very short for 7 and 8 instances where the number of routes can be greatly reduced. On the other hand, it is considered that the influence of whether or not to share the vertex in the shortest route of the two OD pairs on reduction amount is not large.

Table 8: Information of Instances's shortest paths

	1	2	3	4	5	6	7	8	9	10
Distances of shortest paths	3,4	3,6	4,4	4,6	7,4	2,8	2,4	3,2	7,4	3,5
Sharing nords in ODs	Yes	No	No	Yes	Yes	No	No	Yes	No	No

5 Conclusion

In this research, we dealt with speeding up of solution by replicator dynamics for general form of the selfish routing game. Although the effect of speeding up is recognized, it is effective depending on the problem given. We also found that it is different depending on the problem which of the two methods can be solved quickly. It cannot be said that we could sufficiently consider what kind of effect it is for what kind of problem in our experiments. It is thought that it is necessary to conduct experiments on more problems in the future.

We also want to investigate whether the model using replicator dynamics can solve a model that was difficult to obtain an equilibrium solution by the existing method.

Moreover, convexity is important property in the selfish routing games and traffic assignment problems[14][15]. In the traffic assignment problems, Frank-Wolfe method[16] is usually used to compute an equilibrium flow. In this algorithm, convexity is needed. Our algorithm also needs convexity for the cost function. This paper employs a linear cost function, however, our algorithm can be applied to general type convex function. This is our most interest future work.

Acknowledgment

This research was partially supported by Grants-in-Aid for Scientific Research (B) 15H02972 and (C) 26330025.

References

[1] J. Leape: The London Congestion Charge. *Journal of Economic Perspectives*, **20**(4), (2006)157–176.

[2] K.A. Small, E.T. Verhoef, and R. Lindsey: *The economics of ueban transportation*. Routledge in Tayler & Francis, (2007).

[3] D. Monderer and L.S. Shapley: Potential Games. *GAMES AND ECONOMIC BEHAVIOR*, **14**, (1996)124-143.

[4] T. Roughgarden: Routing Games, *Algorithmic Game Theory*, Noan Nisan, Tim Roughgarden, Eva Tardos, Cijay V. Vazirani,CAMBRIDGE UNIVERSITY PRESS, (2007)461-484.

[5] K. Yoshida, T. Okamoto and S. Koakutsu: Equilibrium Solution Search on a Selfish Routing Problem with Multiple Constraints using the Variable Metric Gradient Projection Method. *2015 IEEE International Conference on Systems, Man, and Cybernetics*, (2015)3023–3029.

[6] S. Fischer and B. Vocking: On the Evolution of Selfish Routing. *Algorithms - ESA 2004*,(2004)323–334.

[7] M.J. Smith: The existence, uniqueness and stability of traffic equilibria. *Transportation Research Part B: Methodological*, **13**(4), (1979)295–304.

[8] A.K. Ziliaskopoulos: A Liner Programming Model for the Single Destination System Optimum Dynamic Traffic Assignment Problem. *Presentation preprint, Transportation Research Boarai, Washington, D. C.*, (1997).

[9] S. Suri, C.D. Toth, and Y. Zhou: Selfish Load Balancing and Atomic Congestion Games. *Algorithmica*,**47**(1), (2007)79–96.

[10] R. Cressman and Y. Tao: The replicator equation and other game dynamics. *Proceedings of the National Academy of Sciences of the United States of America*, **111**, (2014)10810–10817.

[11] P. Schuster and K. Sigmund: Replicator dynamics. *Journal of Theoretical Biology*, **100**(3), (1983)533–538.

[12] K. Yoshida, T. Okamoto and S. Koakutsu: An Efficiency Improvement of the Equilibrium Solution Search on the Selfish Routing Game by Removing Redundant Paths. *SICE Journal of Control, Measurement, and System Integration*, **9**, (2016)234–241.

[13] JPN network. IEICE: PN Homepage,*https://www.ieice.org/cs/pn/jpn/jpnm.html*. (2018/11/15 Accessed)

[14] A. Taguchi: TIME DEPENDENT TRAFFIC ASSIGNMENT MODEL FOR COMMUTER TRAFFIC IN TOKYO METROPOLITAN RAILWAY NETWORK. *Transactions of the Operations Research Society of Japan*, **48**, (2005)85–108.

[15] T. Larsson and M. Patriksson: Simplicial Decomposition with Disaggregated Representation for the Traffic Assignment Problem. *Transportation Science*, **26**(1), (1992)4–17.

[16] M. Jaggi: Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization. *Proceedings of the 30th International Conference on Machine Learning, PMLR*, **28**(1), (2013)427–435.