Efficient Convergence to Pure Nash Equilibria in Weighted Network Congestion Games*

Panagiota N. Panagopoulou and Paul G. Spirakis

Computer Technology Institute, Riga Feraiou 61, 26221, Patras, Greece
Computer Engineering and Informatics Department, Patras University, Greece
panagopp@hermes.cti.gr; spirakis@cti.gr

Abstract. In large-scale or evolving networks, such as the Internet, there is no authority possible to enforce a centralized traffic management. In such situations, Game Theory and the concepts of Nash equilibria and Congestion Games [8] are a suitable framework for analyzing the equilibrium effects of selfish routes selection to network delays.
We focus here on *layered* networks where selfish users select paths to route their loads (represented by arbitrary integer *weights*). We assume that individual link delays are equal to the total load of the link. We focus on the algorithm suggested in [2], i.e. a potential-based method for finding *pure* Nash equilibria (PNE) in such networks. A superficial analysis of this algorithm gives an upper bound on its time which is polynomial in n (the number of users) and the sum of their weights. This bound can be exponential in n when some weights are superpolynomial. We provide strong experimental evidence that this algorithm actually converges to a PNE in strong *polynomial time* in n (independent of the weights values). In addition we propose an initial allocation of users to paths that dramatically accelerates this algorithm, compared to an arbitrary initial allocation. A by-product of our research is the discovery of a weighted potential function when link delays are *exponential* to their loads. This asserts the existence of PNE for these delay functions and extends the result of [2].

1 Introduction

In large-scale or evolving networks, such as the Internet, there is no authority possible to employ a centralized traffic management. Besides the lack of central regulation, even cooperation of the users among themselves may be impossible due to the fact that the users may not even know each other. A natural assumption in the absence of central regulation and coordination is to assume

* This work was partially supported by the EU within the Future and Emerging Technologies Programme under contract IST200133135 (CRESCCO) and within the 6th Framework Programme under contract 001907 (DELIS).

S.E. Nikoletseas (Ed.): WEA 2005, LNCS 3503, pp. 203–215, 2005.

that network users behave selfishly and aim at optimizing their own individual welfare. Thus, it is of great importance to investigate the selfish behavior of users so as to understand the mechanisms in such non-cooperative network systems.

Since each user seeks to determine the shipping of its own traffic over the network, different users may have to optimize completely different and even conflicting measures of performance. A natural framework in which to study such multi-objective optimization problems is (non-cooperative) game theory. We can view network users as independent agents participating in a non-cooperative game and expect the routes chosen by users to form a Nash equilibrium in the sense of classical game theory: a Nash equilibrium is a state of the system such that no user can decrease his individual cost by unilaterally changing his strategy.

Users selfishly choose their private strategies, which in our environment correspond to paths from their sources to their destinations. When routing their traffics according to the strategies chosen, the users will experience an expected latency caused by the traffics of all users sharing edges (i.e the latency on the edges depends on their congestion). Each user tries to minimize his private cost, expressed in terms of his individual latency. If we allow as strategies for each user all probability distributions on the set of their source-destination paths, then a Nash equilibrium is guaranteed to exist. It is very interesting however to explore the existence of *pure* Nash equilibria (PNE) in such systems, i.e. situations in which each user is deterministically assigned on a path from which he has no incentive to unilaterally deviate.

Rosenthal [8] introduced a class of games, called *congestion games*, in which each player chooses a particular subset of resources out of a family of allowable subsets for him (his strategy set), constructed from a basic set of primary resources for all the players. The *delay* associated with each primary resource is a non-decreasing function of the number of players who choose it, and the total delay received by each player is the sum of the delays associated with the primary resources he chooses. Each game in this class possesses at least one Nash equilibrium in pure strategies. This result follows from the existence of a real-valued function (an *exact potential* [6]) over the set of pure strategy profiles with the property that the gain of a player unilaterally shifting to a new strategy is equal to the corresponding increment of the potential function.

In a *multicommodity network congestion game* the strategy set of each player is represented as a set of origin-destination paths in a network, the edges of which play the role of resources. If all origin-destination pairs of the users coincide we have a *single commodity network congestion game*. In a *weighted congestion game* we allow users to have different demands for service, and thus affect the resource delay functions in a different way, depending on their own weights. Hence weighted congestion games are not guaranteed to possess a PNE.

Related Work. As already mentioned, the class of (unweighted) congestion games is guaranteed to have at least one PNE. In [1] it is proved that a PNE for any (unweighted) single commodity network congestion game can be constructed in polynomial time, no matter what resource delay functions are considered (so long as they are non-decreasing functions with loads). On the other hand, it is

shown that even for an unweighted multicommodity network congestion game it is PLS-complete to find a PNE, though it certainly exists.

For the special case of single commodity network congestion games where the network consists of parallel edges from a unique origin to a unique destination and users have varying demands, it was shown in [3] that there is always a pure Nash equilibrium which can be constructed in polynomial time.

[5] deals with the problem of weighted parallel-edges congestion games with user-specific costs: each allowable strategy of a user consists of a single resource and each user has his own private cost function for each resource. It is shown that all such games involving only two users, or only two possible strategies for all the users, or equal delay functions, always possess a PNE. However, it is shown that even a 3-user, 3-strategies, weighted parallel-edges congestion game may not possess a PNE.

In [2] it is proved that even for a weighted single commodity network congestion game with resource delays being either linear or 2-wise linear functions of their loads, there may be no PNE. Nevertheless, it is proved that for the case of a weighted single commodity ℓ-layered network congestion game (to be defined later) with resource delays identical to their loads, at least one PNE exists and can be computed in pseudo-polynomial time.

Our Results. We focus our interest on weighted ℓ-layered network congestion games with resource delays equal to their loads. As already mentioned, any such game possesses a PNE, and the algorithm suggested in [2] requires at most a pseudo-polynomial number of steps to reach an equilibrium; this bound however has not yet been proven to be tight. The algorithm starts with any initial allocation of users on paths and iteratively allows each unsatisfied user to switch to any other path where he could reduce his cost. We experimentally show that the algorithm actually converges to a PNE in polynomial time for a variety of networks and distributions of users' weights. In addition, we propose an initial allocation of users onto paths that, as our experiments show, leads to a significant reduction of the total number of steps required by the algorithm, as compared to an arbitrary initial allocation.

Moreover, we present a **b**-potential function for any single commodity network congestion game with resource delays exponential to their loads, thus assuring the existence of a PNE in any such game (Theorem 2).

2 Definitions and Notation

Games, Congestion Games and Weighted Congestion Games. A *game* $\Gamma = \langle N, (\Pi_i)_{i\in N}, (u_i)_{i\in N}\rangle$ in strategic form is defined by a finite set of *players* $N = \{1, \ldots, n\}$, a finite set of *strategies* Π_i for each player $i \in N$, and a *payoff function* $u_i : \Pi \to \mathbb{R}$ for each player, where $\Pi \equiv \times_{i\in N}\Pi_i$ is the set of *pure strategy profiles* or *configurations*. A game is *symmetric* if all players are indistinguishable, i.e. all Π_i's are the same and all u_i's, considered as a function of the choices of the other players, are identical symmetric functions of $n-1$ variables. A *pure*

Nash equilibrium (PNE) is a configuration $\pi = (\pi_1, \ldots, \pi_n)$ such that for each player i, $u_i(\pi) \geq u_i(\pi_1, \ldots, \pi'_i, \ldots, \pi_n)$ for any $\pi'_i \in \Pi_i$. A game may not possess a PNE in general. However, if we extend the game to include as strategies for each i all possible probability distributions on Π_i and if we extend the payoff functions u_i to capture expectation, then an equilibrium is guaranteed to exist [7].

A *congestion model* $\langle N, E, (\Pi_i)_{i\in N}, (d_e)_{e\in E}\rangle$ is defined as follows. N denotes the set of players $\{1, \ldots, n\}$. E denotes a finite set of resources. For $i \in N$ let Π_i be the set of strategies of player i, where each $\varpi_i \in \Pi_i$ is a nonempty subset of resources. For $e \in E$ let $d_e : \{1, \ldots, n\} \to \mathbb{R}$ denote the delay function, where $d_e(k)$ denotes the cost (e.g. delay) to each user of resource e, if there are exactly k players using e. The *congestion game* associated with this congestion model is the game in strategic form $\langle N, (\Pi_i)_{i\in N}, (u_i)_{i\in N}\rangle$, where the payoff functions u_i are defined as follows: Let $\Pi \equiv \times_{i\in N}\Pi_i$. For all $\varpi = (\varpi_1, \ldots, \varpi_n) \in \Pi$ and for every $e \in E$ let $\sigma_e(\varpi)$ be the number of users of resource e according to the configuration ϖ: $\sigma_e(\varpi) = |\{i \in N : e \in \varpi_i\}|$. Define $u_i : \Pi \to \mathbb{R}$ by $u_i(\varpi) = -\sum_{e\in\varpi_i} d_e(\sigma_e(\varpi))$.

In a *weighted congestion model* we allow the users to have different demands, and thus affect the resource delay functions in a different way, depending on their own weights. A weighted congestion model $\langle N, (w_i)_{i\in N}, E, (\Pi_i)_{i\in N}, (d_e)_{e\in E}\rangle$ is defined as follows. N, E and Π_i are defined as above, while w_i denotes the demand of player i and for each resource $e \in E$, $d_e(\cdot)$ is the delay per user that requests its service, as a function of the total usage of this resource by all the users. The *weighted congestion game* associated with this congestion model is the game in strategic form $\langle (w_i)_{i\in N}, (\Pi_i)_{i\in N}, (u_i)_{i\in N}\rangle$, where the payoff functions u_i are defined as follows. For any configuration $\varpi \in \Pi$ and for all $e \in E$, let $\Lambda_e(\varpi) = \{i \in N : e \in \varpi_i\}$ be the set of players using resource e according to ϖ. The cost $\lambda^i(\varpi)$ of user i for adopting strategy $\varpi_i \in \Pi_i$ in a given configuration ϖ is equal to the cumulative delay $\lambda_{\varpi_i}(\varpi)$ on the resources that belong to ϖ_i: $\lambda^i(\varpi) = \lambda_{\varpi_i}(\varpi) = \sum_{e\in\varpi_i} d_e(\theta_e(\varpi))$ where, for all $e \in E$, $\theta_e(\varpi) \equiv \sum_{i\in\Lambda_e(\varpi)} w_i$ is the load on resource e with respect to the configuration ϖ. The payoff function for player i is then $u_i(\varpi) = -\lambda^i(\varpi)$. A configuration $\varpi \in \Pi$ is a PNE if and only if, for all $i \in N$, $\lambda_{\varpi_i}(\varpi) \leq \lambda_{\pi_i}(\varpi_{-i}, \pi_i) \quad \forall \pi_i \in \Pi_i$, where (ϖ_{-i}, π_i) is the same configuration as ϖ except for user i that has now been assigned to path π_i. Since the payoff functions u_i can be implicitly computed by the resource delay functions d_e, in the following we will denote a weighted congestion game by $\langle (w_i)_{i\in N}, (\Pi_i)_{i\in N}, (d_e)_{e\in E}\rangle$.

In a *network congestion game* the families of subsets Π_i are presented implicitly as paths in a network. We are given a directed network $G = (V, E)$ with the edges playing the role of resources, a pair of nodes $(s_i, t_i) \in V \times V$ for each player i and the delay function d_e for each $e \in E$. The strategy set of player i is the set of all paths from s_i to t_i. If all origin-destination pairs (s_i, t_i) of the players coincide with a unique pair (s, t) we have a *single commodity network congestion game* and then all users share the same strategy set, hence the game is symmetric. If users have different demands, we refer to *weighted network con-*

gestion games in the natural way. In the case of a *weighted single commodity network congestion game* however the game is not necessarily symmetric, since the users have different demands and thus their cost functions will also differ.

Potential Functions. Fix some vector $\mathbf{b} \in \mathbb{R}^n_{>0}$. A function $F : \times_{i\in N} \Pi_i \to \mathbb{R}$ is a $\mathbf{b}$-*potential* for the weighted congestion game $\Gamma = \langle (w_i)_{i\in N}, (\Pi_i)_{i\in N}, (d_e)_{e\in E}\rangle$ if $\forall \varpi \in \times_{i\in N}\Pi_i, \forall i \in N, \forall \pi_i \in \Pi_i,\ \lambda^i(\varpi) - \lambda^i(\varpi_{-i}, \pi_i) = b_i \cdot (F(\varpi) - F(\varpi_{-i}, \pi_i))$. F is an *exact potential* for Γ if $b_i = 1$ for all $i \in N$. It is well known [6] that if there exists a $\mathbf{b}$-potential for a game Γ, then Γ possesses a PNE.

Layered Networks. Let $\ell \geq 1$ be an integer. A directed network (V, E) with a distinguished source-destination pair (s, t), $s, t \in V$, is ℓ-layered if every directed $s-t$ path has length exactly ℓ and each node lies on a directed $s-t$ path. The nodes of an ℓ-layered network can be partitioned into $\ell + 1$ layers, V_0, V_1, ..., V_ℓ, according to their distance from the source node s. Since each node lies on directed $s-t$ path, $V_0 = \{s\}$ and $V_\ell = \{t\}$. Similarly we can partition the edges E of an ℓ-layered network in ℓ subsets $E_1, \ldots, E_\ell$ where for all $j \in \{1, \ldots, \ell\}$, $E_j = \{e = (u, v) \in E : u \in V_{j-1} \text{ and } v \in V_j\}$.

3 The Problem

We focus our interest on the existence and tractability of pure Nash equilibria in weighted ℓ-layered network congestion games with resource delays identical to their loads. Consider the ℓ-layered network $G = (V, E)$ with a unique source-destination pair (s, t) and the weighted single commodity network congestion game $\langle (w_i)_{i\in N}, \mathcal{P}, (d_e)_{e\in E}\rangle$ associated with G, such that $\mathcal{P}$ is the set of all directed $s-t$ paths of G and $d_e(x) = x$ for all $e \in E$. Let $\varpi = (\varpi_1, \ldots, \varpi_n)$ be an arbitrary configuration and recall that $\theta_e(\varpi)$ denotes the load of resource $e \in E$ under configuration ϖ. Since resource delays are equal to their loads, for all $i \in N$ it holds that $\lambda^i(\varpi) = \lambda_{\varpi_i}(\varpi) = \sum_{e\in\varpi_i} \theta_e(\varpi) = \sum_{e\in\varpi_i} \sum_{j\in N | e\in\varpi_j} w_j$.

A user $i \in N$ is *satisfied* in the configuration $\varpi \in \mathcal{P}^n$ if he has no incentive to unilaterally deviate from ϖ, i.e. if for all $s-t$ paths $\pi \in \mathcal{P}$, $\lambda_{\varpi_i}(\varpi) \leq \lambda_\pi(\varpi_{-i}, \pi)$. The last inequality can be written equivalently

$$\lambda_{\varpi_i}(\varpi_{-i}) + \ell w_i \leq \lambda_\pi(\varpi_{-i}) + \ell w_i \quad \Longleftrightarrow \quad \lambda_{\varpi_i}(\varpi_{-i}) \leq \lambda_\pi(\varpi_{-i}) \ ,$$

hence user i is satisfied if and only if he is assigned to the shortest $s-t$ path calculated with respect to the configuration ϖ_{-i} of all the users except for i. The configuration ϖ is a PNE if and only if all users are satisfied in ϖ. In [2] it was shown that any such weighted ℓ-layered network congestion game possesses a PNE that can be computed in pseudo-polynomial time:

Theorem 1 ([2]). *For any weighted ℓ-layered network congestion game with resource delays equal to their loads, at least one PNE exists and can be computed in pseudo-polynomial time.*

Proof (sketch). The **b**-potential function establishing the result is

$$\Phi(\varpi) = \sum_{e \in E} (\theta_e(\varpi))^2$$

where, $\forall i \in N, b_i = \frac{1}{2w_i}$. □

In Sect. 4 we present the pseudo-polynomial algorithm *Nashify*() for the computation of a PNE for a weighted ℓ-layered network congestion game, while in Sect. 6 we experimentally show that such a PNE can actually be computed in polynomial time, as our following conjecture asserts:

Conjecture 1. Algorithm *Nashify*() converges to a PNE in polynomial time.

4 The Algorithm

The algorithm presented below converts any given non-equilibrium configuration into a PNE by performing a sequence of greedy selfish steps. A greedy selfish step is a user's change of his current pure strategy (i.e. path) to his best pure strategy with respect to the current configuration of all other users.

Algorithm *Nashify*$(G, (w_i)_{i \in N})$
Input: ℓ-layered network G and a set N of users, each user i having weight w_i
Output: configuration ϖ which is a PNE

1. begin
2. select an initial configuration $\varpi = (\varpi_1, \ldots, \varpi_n)$
3. while $\exists$ user i such that $\lambda_{\varpi_i}(\varpi_{-i}) > \lambda_s(\varpi_{-i})$ where $s = \text{Shortest\_Path}(\varpi_{-i})$
4. $\varpi_i := \text{Shortest\_Path}(\varpi_{-i})$
5. return ϖ
6. end

The above algorithm starts with an initial allocation of each user $i \in N$ on an $s-t$ path ϖ_i of the ℓ-layered network G. The algorithm iteratively examines whether there exists any user that is unsatisfied. If there is such a user, say i, then user i performs a greedy selfish step, i.e. he switches to the shortest $s-t$ path according to the configuration ϖ_{-i}. The existence of the potential function Φ assures that the algorithm will terminate after a finite number of steps at a configuration from which no user will have an incentive to deviate, i.e. at a PNE.

Complexity Issues. Let $W = \sum_{i \in N} w_i$. Note that in any configuration $\varpi \in \mathcal{P}^n$ and for all $j \in \{1, \ldots, \ell\}$ it holds that $\sum_{e \in E_j} \theta_e(\varpi) = W$. It follows that

$$\Phi(\varpi) = \sum_{e \in E} (\theta_e(\varpi))^2 = \sum_{j=1}^{\ell} \sum_{e \in E_j} (\theta_e(\varpi))^2 \leq \sum_{j=1}^{\ell} \left(\sum_{e \in E_j} \theta_e(\varpi) \right)^2 = \ell W^2 \ .$$

Without loss of generality assume that the users have integer weights. At each iteration of the algorithm *Nashify*() an unsatisfied user performs a greedy selfish

step, so his cost must decrease by at least 1 and thus the potential function Φ decreases by at least $2 \min_i w_i \geq 2$. Hence the algorithm requires at most $\frac{1}{2}\ell W^2$ steps so as to converge to a PNE.

Proposition 1. *Suppose that* $\frac{(\max_i w_i)^2}{\min_i w_i} = O(n^k)$ *for some constant* k. *Then algorithm Nashify() will converge to a PNE in polynomial time.*

Proof. Observe that $\Phi(\varpi) \leq \ell W^2 \leq \ell(n \max_i w_i)^2 = \ell n^2 \min_i w_i \cdot O(n^k)$, which implies that the algorithm will reach a PNE in $O(\ell n^{k+2})$ steps. □

5 The Case of Exponential Delay Functions

In this section we deal with the existence of pure Nash equilibria in weighted single commodity network congestion games with resource delays being exponential to their loads. Let $G = (V, E)$ be any single commodity network (not necessarily layered) and denote by $\mathcal{P}$ the set of all $s - t$ paths in it from the unique source s to the unique destination t. Consider the weighted network congestion game $\Gamma = \langle (w_i)_{i \in N}, \mathcal{P}, (d_e)_{e \in E} \rangle$ associated with G, such that for any configuration $\varpi \in \mathcal{P}^n$ and for all $e \in E$, $d_e(\theta_e(\varpi)) = \exp(\theta_e(\varpi))$. We next show that $F(\varpi) = \sum_{e \in E} \exp(\theta_e(\varpi))$ is a **b**-potential for such a game and some positive n-vector **b**, assuring the existence of a PNE.

Theorem 2. *For any weighted single commodity network congestion game with resource delays exponential to their loads, at least one PNE exists.*

Proof. Let $\varpi \in \mathcal{P}^n$ be an arbitrary configuration. Let i be a user of demand w_i and fix some path $\pi_i \in \mathcal{P}$. Denote $\varpi' \equiv (\varpi_{-i}, \pi_i)$. Observe that

$$\lambda^i(\varpi) - \lambda^i(\varpi') = \sum_{e \in \varpi_i \setminus \pi_i} \exp(\theta_e(\varpi_{-i}) + w_i) - \sum_{e \in \pi_i \setminus \varpi_i} \exp(\theta_e(\varpi_{-i}) + w_i)$$

$$= \exp(w_i) \cdot \left(\sum_{e \in \varpi_i \setminus \pi_i} \exp(\theta_e(\varpi_{-i})) - \sum_{e \in \pi_i \setminus \varpi_i} \exp(\theta_e(\varpi_{-i})) \right) .$$

Note that, for all $e \notin \{\{\varpi_i \setminus \pi_i\} \cup \{\pi_i \setminus \varpi_i\}\}$, it holds that $\theta_e(\varpi) = \theta_e(\varpi')$. Now

$$F(\varpi) - F(\varpi') = \sum_{e \in \varpi_i \setminus \pi_i} \exp\left(\theta_e(\varpi_{-i}) + w_i\right) - \exp(\theta_e(\varpi_{-i}))$$

$$+ \sum_{e \in \pi_i \setminus \varpi_i} \exp(\theta_e(\varpi_{-i})) - \exp(\theta_e(\varpi_{-i}) + w_i)$$

$$= \frac{\exp(w_i) - 1}{\exp(w_i)} \left(\lambda^i(\varpi) - \lambda^i(\varpi')\right) .$$

Thus, F is a **b**-potential for our game, where $\forall i \in N, b_i = \frac{\exp(w_i)}{\exp(w_i) - 1}$, assuring the existence of at least one PNE. □

6 Experimental Evaluation

Implementation Details. We implemented algorithm *Nashify*() in C++ programming language using several advanced data types of LEDA [4]. In our implementation, we considered two initial allocations of users on paths: (1) Random allocation: each user assigns its traffic uniformly at random on an $s-t$ path and (2) Shortest Path allocation: users are sorted according to their weights, and the maximum weighted user among those that have not been assigned a path yet selects a path of shortest length, with respect to the loads on the edges caused by the users of larger weights.

Note that, in our implementation, the order in which users are checked for satisfaction (line 3 of algorithm *Nashify*()) is the worst possible, i.e. we sort users according to their weights and, at each iteration, we choose the minimum weighted user among the unsatisfied ones to perform a greedy selfish step. By doing so, we force the potential function to decrease as less as possible and thus we maximize the number of iterations, so as to be able to better analyze the worst-case behavior of the algorithm.

6.1 Experimental Setup

Networks. Figure 1 shows the ℓ-layered networks considered in our experimental evaluation of algorithm *Nashify*(). Network 1 is the simplest possible layered network and Network 2 is a generalization of it. Observe that the number of possible $s-t$ paths of Network 1 is 3, while the number of possible $s-t$ paths for Network 2 is 3^5. Network 3 is an arbitrary dense layered network and Network 4 is the 5×5 grid. Network 5 is a 4-layered network with the property that layers $1, 2, 3$ form a tree rooted at s and layer 4 comprises all the edges connecting the leaves of this tree with t.

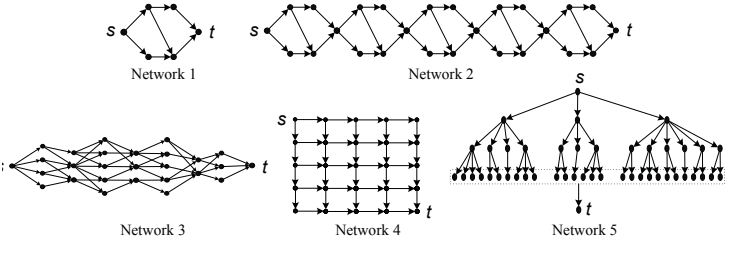

Fig. 1. The $s-t$ layered networks considered

Distribution of weights. For each network, we simulated the algorithm *Nashify*() for $n = 10, 11, \ldots, 100$ users. Obviously, if users' weights are polynomial in n then the algorithm will definitely terminate after a polynomial number of steps. Based on this fact, as well as on Proposition 1, we focused on instances

where some users have exponential weights. More specifically, we considered the following four distributions of weights: (a) 10% of users have weight $10^{n/10}$ and 90% of users have weight 1, (b) 50% of users have weight $10^{n/10}$ and 50% of users have weight 1, (c) 90% of users have weight $10^{n/10}$ and 10% of users have weight 1, and (d) users have uniformly at random selected weights in the interval $[1, 10^{n/10}]$. Distributions (a)–(c), albeit simple, represent the distribution of service requirements in several communication networks, where a fraction of users has excessive demand that outweighs the demand of the other users.

6.2 Results and Conclusions

Figures 2–6 show, for each network and each one of the distributions of weights (a)–(d), the number of steps performed by algorithm *Nashify*() over the number of users (#steps/n) as a function of the sum of weights of all users W. For each instance we considered both random and shortest path initial allocation.

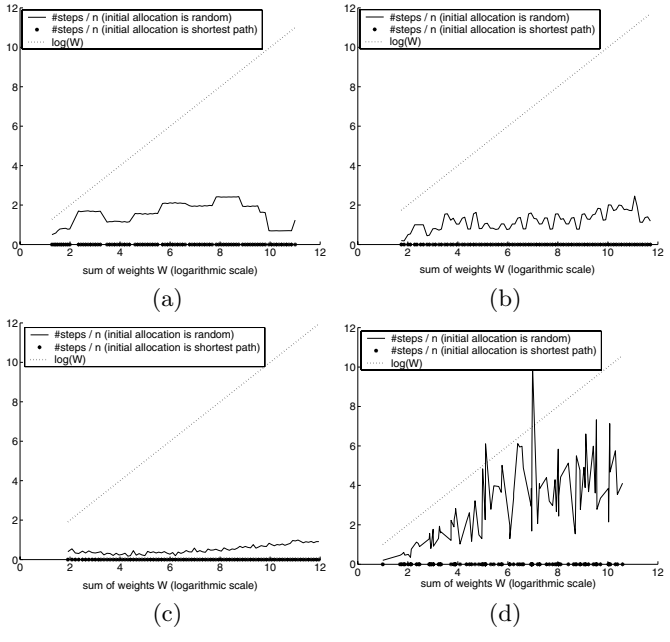

Fig. 2. Experimental results for Network 1

Observe that the shortest path initial allocation significantly outperforms any random initial allocation, no matter what networks or distributions of weights are considered. In particular, the shortest path initial allocation appears to be a PNE for sparse (Networks 1 and 2), grid (Network 4) and tree-like (Network 5) networks, while for the dense network (Network 3) the number of steps over the number of users seems to be bounded by a small constant.

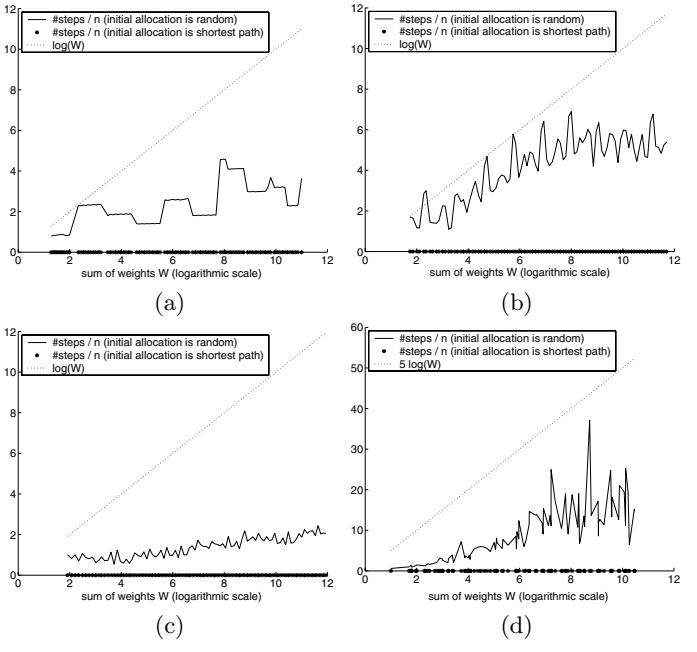

Fig. 3. Experimental results for Network 2

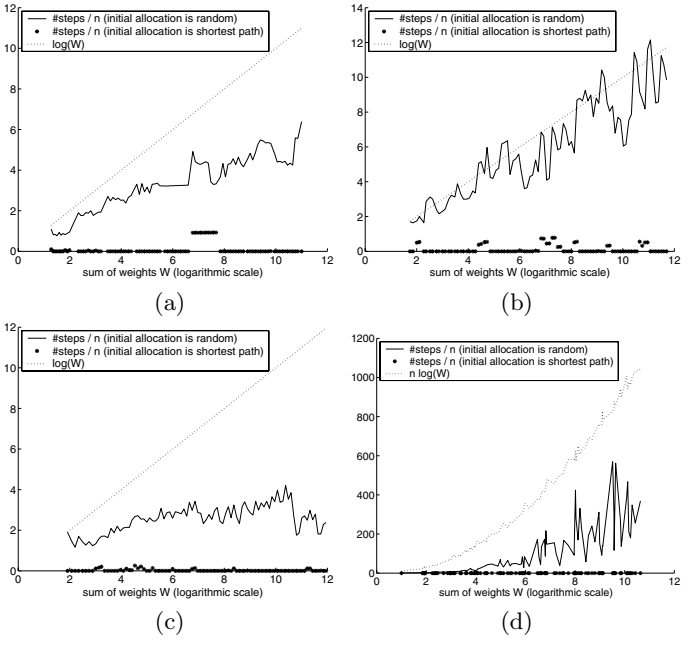

Fig. 4. Experimental results for Network 3

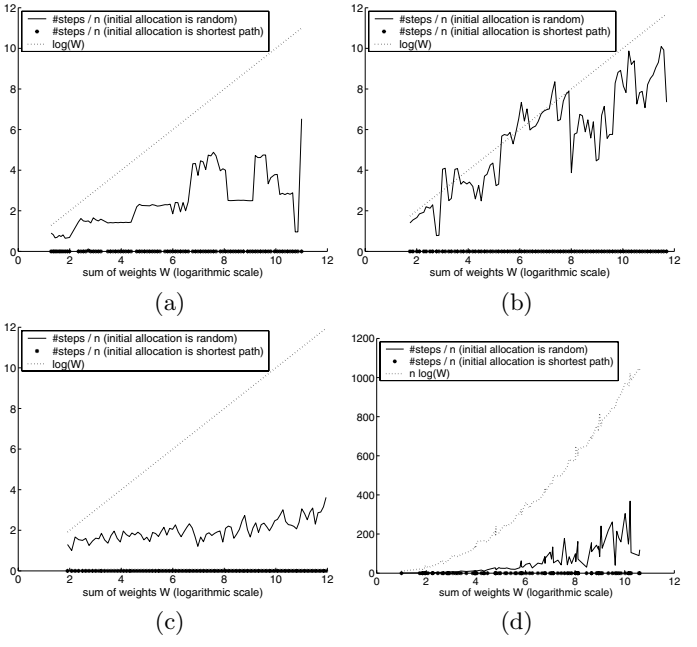

Fig. 5. Experimental results for Network 4

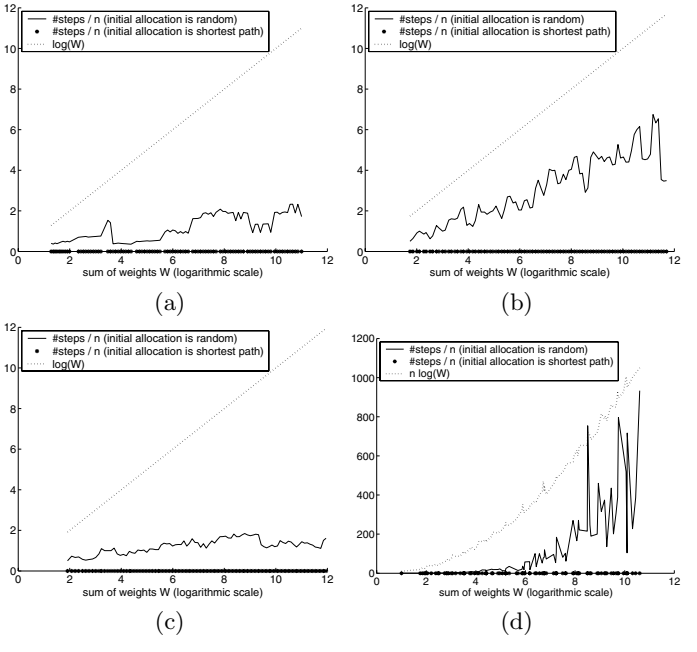

Fig. 6. Experimental results for Network 5

On the other hand, the behavior of the algorithm when starting with an arbitrary allocation is sensibly worse. First note that, in this case, the fluctuations observed at the plots are due to the randomization of the initial allocation. On the average however we can make safe conclusions as regards the way #steps/n increases as a function of W. For the distributions of weights (a)–(c) it is clear that the number of steps over the number of users is asymptotically upper bounded by the logarithm of the sum of all weights, implying that #steps $= O(n \cdot \log(W))$. Unfortunately, the same does not seem to hold for randomly selected weights (distribution (d)). In this case however, as Figs. 2–6(d) show, $n \log(W)$ seems to be a good asymptotic upper bound for #steps/n, suggesting that #steps $= O(n^2 \cdot \log(W))$.

Note that, for all networks, the maximum number of steps over the number of users occurs for the random distribution of weights. Also observe that, for the same value of the sum of weights W, the number of steps is dramatically smaller when there are only 2 distinct weights (distributions (a)–(c)). Hence we conjecture that the complexity of the algorithm does actually depend not only on the sum of weights, but also on the number of distinct weights of the input.

Also note that the results shown in Figs. 2 and 3 imply that, when starting with an arbitrary allocation, the number of steps increases as a linear function of the size of the network. Since the number of $s - t$ paths in Network 2 is exponential in comparison to that of Network 1, we would expect a significant increment in the number of steps performed by the algorithm. Figures 2 and 3 however show that this is not the case. Instead, the number of steps required for Network 2 are at most 5 times the number of steps required for Network 1.

Summarizing our results, we conclude that (i) a shortest path initial allocation is a few greedy selfish steps far from a PNE, amplifying Conjecture 1, while (ii) an arbitrary initial allocation does not assure a similarly fast convergence to a PNE, however Conjecture 1 seems to be valid for this case as well, (iii) the size of the network does not affect significantly the time complexity of the algorithm, and (iv) the worst-case input for an arbitrary initial allocation occurs when all users' weights are distinct and some of them are exponential.

References

1. Fabrikant, A., Papadimitriou, C., Talwar, K.: The Complexity of Pure Nash Equilibria. Proc. of the 36th ACM Symp. on Theory of Computing (STOC 04), 2004.
2. Fotakis, D., Kontogiannis, S., Spirakis, P.: Selfish Unsplittable Flows. 31st International Colloquium on Automata, Languages and Programming (ICALP'04), pp. 593–605, 2004.
3. Fotakis, D., Kontogiannis, S., Koutsoupias, E., Mavronicolas, M., Spirakis, P.: The Structure and Complexity of Nash Equilibria for a Selfish Routing Game. Proc. of the 29th International Colloquium on Automata, Languages and Programming (ICALP 02), Springer-Verlag, 2002, pp. 123–134.
4. Mehlhorn, K., Näher, S.: LEDA – A Platform for Combinatorial and Geometric Computing. Cambridge University Press, 1999.

5. Milchtaich, I.: Congestion Games with Player-Specific Payoff Functions. Games and Economic Behavior 13 (1996), 111–124.
6. Monderer, D., Shapley, L.: Potential Games. Games and Economic Behavior, 14:124–143, 1996.
7. Nash, J. F.: Equilibrium Points in N-person Games. Proc. of National Academy of Sciences, Vol. 36, pp. 48–49, 1950.
8. Rosenthal, R. W.: A Class of Games Poseessing Pure-Strategy Nash Equilibria. International Journal of Game Theory 2, pp. 65–67, 1973.