Algorithms for Pure Nash Equilibria in Weighted Congestion Games

PANAGIOTA N. PANAGOPOULOU and PAUL G. SPIRAKIS
Patras University, Greece

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 especially the concepts of Nash equilibria and congestion games [Rosenthal 1973] are a suitable framework for analyzing the equilibrium effects of selfish routes selection to network delays. We focus here on *single-commodity* 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 then focus on the algorithm suggested in Fotakis et al. [2005], i.e., a potential-based method for finding *pure* Nash equilibria 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 W. This bound can be exponential in n when some weights are exponential. We provide strong experimental evidence that this algorithm actually converges to a pure Nash equilibrium in *polynomial time*. More specifically, our experimental findings suggest that the running time is a polynomial function of n and $\log W$. 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 pure Nash equilibria for these delay functions and extends the result of Fotakis et al. [2005].

Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Routing and layout*

General Terms: Algorithms, Theory, Experimentation

Additional Key Words and Phrases: Congestion games, game theory, pure Nash equilibria

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

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), and by the General Secretariat for Research and Technology of the Greek Ministry of Development within the programme PENED 2003.
Authors' address: Research Academic Computer Technology Institute, N. Kazantzaki Str., Patras University, Rion, GR 26 500 Patras, Greece; email: {panagopp,spirakis}@cti.gr.

because of 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 that network users behave selfishly and aim at optimizing their own individual welfare. Thereupon, it is of great importance to investigate the selfish behavior of users in order to understand the mechanisms in such noncooperative network systems.

Since each user seeks to determine the shipping of her 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 multiobjective optimization problems is (noncooperative) game theory. We can view network users as independent agents participating in a noncooperative 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 her individual cost by unilaterally changing her strategy.

Users selfishly choose their private strategies, which, in our setting, correspond to paths from their sources to their destinations. When routing their traffics according to the strategies chosen, the users will experience a 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 her private cost, expressed in terms of her 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 in such systems, i.e., situations in which each user is deterministically assigned on a path from which she has no incentive to unilaterally deviate.

Rosenthal [1973] 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 her (her strategy set), constructed from a basic set of primary resources for all the players. The *delay* associated with each primary resource is a nondecreasing 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 she 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* [Monderer and Shapley 1996]) over the set of pure strategy profiles with the property that the gain (i.e. the increment of the payoff function) 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* and then all users share the same strategy set. 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 pure Nash equilibrium.

1.1 Related Work

As already mentioned, the class of (unweighted) congestion games is guaranteed to have at least one pure Nash equilibrium. In Fabrikant et al. [2004], it is proved that a pure Nash equilibrium 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 nondecreasing 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 pure Nash equilibrium, although 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 Fotakis et al. [2002] that there is always a pure Nash equilibrium, which can be constructed in polynomial time. Moreover, in Gairing et al. [2004], it was shown that a pure Nash equilibrium can be computed in polynomial time even under the restriction that each user may only be routed on a link from a certain set of allowed links for the user.

Milchtaich [1996] 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 her 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 pure Nash equilibrium. On the other hand, it is shown that even a three-user, three-strategies, weighted parallel-edges congestion game may not possess a pure Nash equilibrium.

In Libman and Orda [1997] and in Fotakis et al. [2005] it is (independently) 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 pure Nash equilibrium. Nevertheless, in Fotakis et al. [2005], it is proved that for the case of a weighted network congestion game with resource delays linear to their loads, at least one pure Nash equilibrium exists and can be computed in pseudopolynomial time. In Fotakis et al. [2005], it is shown that the "shortest-path-allocation" rule that we consider in our experiments maintains a pure Nash equilibrium if the network is layered, series-parallel, and has identical resource delays.

The algorithm Nashify() that we present and experimentally evaluate in this work is motivated by Feldmann et al. [2003] and Even-Dar et al. [2003], where *nashification*, i.e., the efficient transformation of an assignment of weights on paths to a pure Nash equilibrium, was first considered.

1.2 Our Results

We focus our interest on weighted single-commodity network congestion games with resource delays equal to their loads. As already mentioned, any such game possesses a pure Nash equilibrium, and the algorithm suggested in Fotakis et al. [2005] requires, at most, a pseudopolynomial number of steps to reach an equilibrium; this bound, however, has not yet been proved 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 she could reduce her cost. We experimentally show that the algorithm actually converges to a pure Nash equilibrium 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 network congestion game with resource delays being exponential to their loads, thus assuring the existence of a pure Nash equilibrium in any such game (Theorem 5.1).

2. DEFINITIONS AND NOTATION

2.1 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* $N = \{1, 2, \ldots, n\}$, a finite set of *strategies* Π_i for each player $i \in N$, and a *payoff function* $u_i : \Pi \to \mathcal{R}$ 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* 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 pure Nash equilibrium, 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 [Nash 1950].

2.2 Congestion Games

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 $d_e : \{1, \ldots, n\} \to \mathcal{R}$ 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 $u_i : \Pi \to \mathcal{R}$ by $u_i(\varpi) = -\sum_{e\in\varpi_i} d_e(\sigma_e(\varpi))$.

In a *network congestion game* the families of subsets Π_i are represented 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.

2.3 Weighted Congestion Games

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 denotes the set of players $\{1, \ldots, n\}$, w_i denotes the demand of player i, and 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 each resource $e \in E$ let $d_e(\cdot)$ be 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 pure Nash equilibrium if, and only if, for all $i \in N$,

$$\lambda_{\varpi_i}(\varpi) \le \lambda_{\pi_i}(\varpi_{-i}, \pi_i) \quad \forall \pi_i \in \Pi_i$$

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 *weighted network congestion game* the strategy sets Π_i are represented implicitly as $s_i - t_i$ paths in a directed network $G = (V, E)$. If all origin-destination pairs (s_i, t_i) of the players coincide with a unique pair (s, t), we have a *weighted single-commodity network congestion game* and then all users share the same strategy set. In this case, however, the game is not necessarily symmetric, since the users have different demands and thus their cost functions will also differ.

2.4 Potential Functions

Fix some vector $\mathbf{b} \in \mathcal{R}^n_{>0}$. A function $F : \times_{i\in N}\Pi_i \to \mathcal{R}$ is a **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 [Monderer and Shapley 1996] that if there exists a **b**-potential for a game in strategic form Γ, then Γ possesses a pure Nash equilibrium.

2.5 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 single-commodity network congestion games with resource delays identical to their loads. Consider the single-commodity network $G = (V, E)$ with a unique source $s \in V$ and a unique destination $t \in V$ 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 she has no incentive to unilaterally deviate from ϖ, i.e. if for all $s-t$ paths $\pi \in \mathcal{P}$, $\lambda_{\varpi_i}(\varpi) \leq \lambda_\pi(\varpi_{-i}, \pi)$. Hence, user i is satisfied if, and only if, she is assigned to a path that minimizes her latency, with respect to the configuration ϖ_{-i} of all the users except for i. The configuration ϖ is a pure Nash equilibrium if, and only if, all users are satisfied in ϖ.

In Fotakis et al. [2005], it was shown that any weighted network congestion game with resource delays being linear functions of their loads possesses a pure Nash equilibrium that can be computed in pseudopolynomial time:

THEOREM 3.1 ([FOTAKIS ET AL. 2005]). *For any weighted multicommodity network congestion game with linear resource delays, at least one pure Nash equilibrium exists and can be computed in pseudopolynomial time.*

PROOF (SKETCH). Fix an arbitrary network $G = (V, E)$ with linear resource delays $d_e(x) = a_e x + b_e$, $e \in E$, $a_e, b_e \geq 0$. Let ϖ be an arbitrary configuration for the corresponding weighted multicommodity congestion game. The function

$$\Phi(\varpi) = \sum_{e\in E} d_e(\theta_e(\varpi))\theta_e(\varpi) + \sum_{i=1}^{n} \sum_{e\in\varpi_i} d_e(w_i)w_i$$

is then a $\mathbf{b}$-potential for the game where, $\forall i \in N, b_i = \frac{1}{2w_i}$. □

Here we focus on weighted single-commodity network congestion games with resource delays identical to their loads, for which case the above theorem yields:

COROLLARY 3.2. *For any weighted single-commodity network congestion game with resource delays equal to their loads, at least one pure Nash equilibrium exists and can be computed in pseudopolynomial time.*

PROOF (SKETCH). The **b**-potential function establishing the result is

$$\Phi(\varpi) = \sum_{e \in E} (\theta_e(\varpi))^2 + \sum_{i=1}^{n} |\varpi_i| w_i^2 \qquad (1)$$

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

In Section 4, we present the pseudopolynomial algorithm Nashify() for the computation of a pure Nash equilibrium for a weighted single-commodity network congestion game, while in Section 6 we provide experimental evidence that such a pure Nash equilibrium can actually be computed in polynomial time, as our following conjecture asserts:

CONJECTURE 3.3. *Algorithm* Nashify() *converges to a pure Nash equilibrium in polynomial time.*

4. THE ALGORITHM

The algorithm presented below converts any given nonequilibrium configuration into a pure Nash equilibrium by performing a sequence of greedy selfish steps. A greedy selfish step is a user's change of her current pure strategy (i.e., path) to her best pure strategy with respect to the current configuration of all other users. By $\text{Shortest\_Path}_i(\varpi_{-i})$ we denote the path that minimizes the latency of user i, with respect to the configuration of all other users.

Algorithm Nashify$(G, (w_i)_{i \in N}, \varpi)$

Input: △ network $G = (V, E)$ with a unique source–destination pair (s, t)
△ a set $N = \{1, \ldots, n\}$ of users, each user i having weight w_i

Output: configuration ϖ which is a pure Nash equilibrium

1. begin
2. select an initial configuration $\varpi = (\varpi_1, \ldots, \varpi_n)$
3. while $\exists$ user i that is unsatisfied
4. $\quad\varpi_i := \text{Shortest\_Path}_i(\varpi_{-i})$
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 single-commodity 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., she switches to the $s-t$ path that minimizes her latency, given the configuration ϖ_{-i}. The existence of the potential function (1) 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 pure Nash equilibrium.

4.1 Complexity Issues

Suppose that the users have arbitrary weights. Let $W = \sum_{i \in N} w_i$. Observe that, for any configuration ϖ,

$$\begin{aligned}\Phi(\varpi) &= \sum_{e \in E} (\theta_e(\varpi))^2 + \sum_{i=1}^{n} |\varpi_i| w_i^2 \\ &\leq \sum_{e \in E} W^2 + \sum_{i=1}^{n} |E| w_i^2 \\ &\leq 2|E|W^2\end{aligned}$$

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 her cost must decrease by at least 1 and, thus, the potential function (1) decreases by at least $2 \min_i w_i \geq 2$. Hence, the algorithm requires, at most, $|E|W^2$ steps so as to converge to a pure Nash equilibrium.

PROPOSITION 4.1. *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 pure Nash equilibrium in polynomial time.*

PROOF. Observe that

$$\begin{aligned}\Phi(\varpi) &\leq 2|E|W^2 \\ &\leq 2|E|(n \max_i w_i)^2 \\ &= 2|E|n^2 \min_i w_i \cdot O(n^k)\end{aligned}$$

which implies that the algorithm will reach a pure Nash equilibrium in $O(|E|n^{k+2})$ steps. □

5. THE CASE OF EXPONENTIAL DELAY FUNCTIONS

In this section, we deal with the existence of pure Nash equilibria in weighted (multicommodity) network congestion games with resource delays being exponential to their loads. Let $G = (V, E)$ be any directed network and let $N = \{1, 2, \ldots, n\}$ be the set of network users. For each $i \in N$, denote by Π_i the set of all $s_i - t_i$ paths in G from the source s_i to the destination t_i. Consider the weighted network congestion game $\Gamma = \langle (w_i)_{i \in N}, (\Pi_i)_{i \in N}, (d_e)_{e \in E} \rangle$ associated with G, such that for any configuration $\varpi \in \times_{i=1}^{n} \Pi_i$ and for all $e \in E$

$$d_e(\theta_e(\varpi)) = \exp(\theta_e(\varpi))$$

We next show that $\Phi(\varpi) = \sum_{e \in E} \exp(\theta_e(\varpi))$ is a **b**-potential for such a game and some positive n-vector **b**, assuring the existence of a pure Nash equilibrium.

THEOREM 5.1. *For any weighted network congestion game with resource delays exponential to their loads, at least one pure Nash equilibrium exists.*

PROOF. Let $\varpi \in \times_{i=1}^{n} \Pi_i$ be an arbitrary configuration. Let i be a user of demand w_i and fix some path $\pi_i \in \Pi_i$. Denote $\varpi' \equiv (\varpi_{-i}, \pi_i)$. Observe that, for all $e \in \{\varpi_i \cap \pi_i\}$, it holds that $\theta_e(\varpi) = \theta_e(\varpi')$. Hence

$$\begin{aligned}
\lambda^i(\varpi) - \lambda^i(\varpi') &= \sum_{e \in \varpi_i} \exp(\theta_e(\varpi)) - \sum_{e \in \pi_i} \exp(\theta_e(\varpi')) \\
&= \sum_{e \in \varpi_i \setminus \pi_i} \exp(\theta_e(\varpi)) - \sum_{e \in \pi_i \setminus \varpi_i} \exp(\theta_e(\varpi')) \\
&= \sum_{e \in \varpi_i \setminus \pi_i} \exp(\theta_e(\varpi_{-i}) + w_i) - \sum_{e \in \pi_i \setminus \varpi_i} \exp(\theta_e(\varpi_{-i}) + w_i) \\
&= \exp(w_i) \cdot \left(\sum_{e \in \varpi_i \setminus \pi_i} \exp(\theta_e(\varpi_{-i})) - \sum_{e \in \pi_i \setminus \varpi_i} \exp(\theta_e(\varpi_{-i})) \right)
\end{aligned}$$

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

$$\begin{aligned}
\Phi(\varpi) - \Phi(\varpi') &= \sum_{e \in E} \exp(\theta_e(\varpi)) - \exp(\theta_e(\varpi')) \\
&= \sum_{e \in \varpi_i \setminus \pi_i} \exp(\theta_e(\varpi)) - \exp(\theta_e(\varpi')) \\
&\quad + \sum_{e \in \pi_i \setminus \varpi_i} \exp(\theta_e(\varpi)) - \exp(\theta_e(\varpi')) \\
&= \sum_{e \in \varpi_i \setminus \pi_i} \exp(\theta_e(\varpi_{-i}) + w_i) - \exp(\theta_e(\varpi_{-i})) \\
&\quad + \sum_{e \in \pi_i \setminus \varpi_i} \exp(\theta_e(\varpi_{-i})) - \exp(\theta_e(\varpi_{-i}) + w_i) \\
&= \sum_{e \in \varpi_i \setminus \pi_i} \exp(\theta_e(\varpi_{-i}))(\exp(w_i) - 1) \\
&\quad - \sum_{e \in \pi_i \setminus \varpi_i} \exp(\theta_e(\varpi_{-i}))(\exp(w_i) - 1) \\
&= (\exp(w_i) - 1) \left(\sum_{e \in \varpi_i \setminus \pi_i} \exp(\theta_e(\varpi_{-i})) - \sum_{e \in \pi_i \setminus \varpi_i} \exp(\theta_e(\varpi_{-i})) \right) \\
&= \frac{\exp(w_i) - 1}{\exp(w_i)} (\lambda^i(\varpi) - \lambda^i(\varpi'))
\end{aligned}$$

Thus, Φ 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 pure Nash equilibrium. □

6. EXPERIMENTAL EVALUATION

6.1 Implementation Details

We implemented algorithm Nashify() in C++ programming language using several advanced data types of LEDA [Mehlhorn and Naher 1999].

In our implementation, we considered two initial allocations of users on paths:

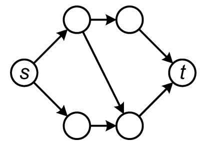


Fig. 1. Network 1.

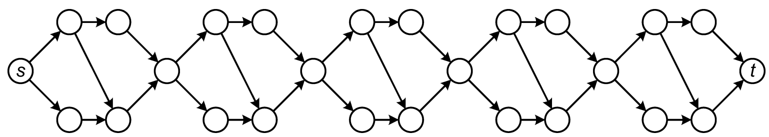


Fig. 2. Network 2.

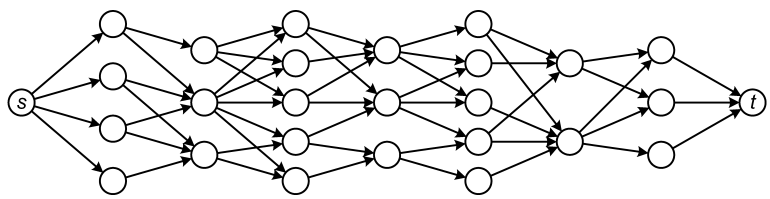


Fig. 3. Network 3.

1. Random allocation: Each user assigns its traffic uniformly at random on an $s - t$ path.
2. Shortest-path allocation: Users are sorted in nonincreasing order of their weights, and the maximum weighted user among those that have not been assigned a path yet selects a path that minimizes her latency, 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 in nondecreasing order of 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 little as possible and thus we maximize the number of iterations, so as to be able to better estimate the worst-case behavior of the algorithm.

6.2 Experimental Setup

6.2.1 *Networks.* Figures 1–9 show the single commodity networks considered in our experimental evaluation of algorithm Nashify(). Network 1 is the simplest possible layered network and Network 2 is its generalization. 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. Networks 5 and 6 are ℓ-layered networks with the property that layers $1, 2, \ldots, \ell - 1$ form a tree rooted at s and layer ℓ comprises all the edges connecting the leaves of this tree with t. Network 7 is the clique of 9 nodes, while Network 8 is a 16-node network with 15 $s - t$

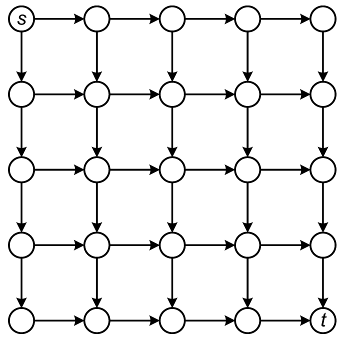


Fig. 4. Network 4.

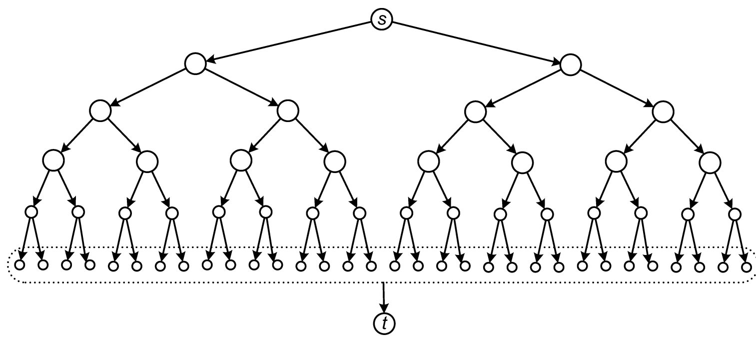


Fig. 5. Network 5.

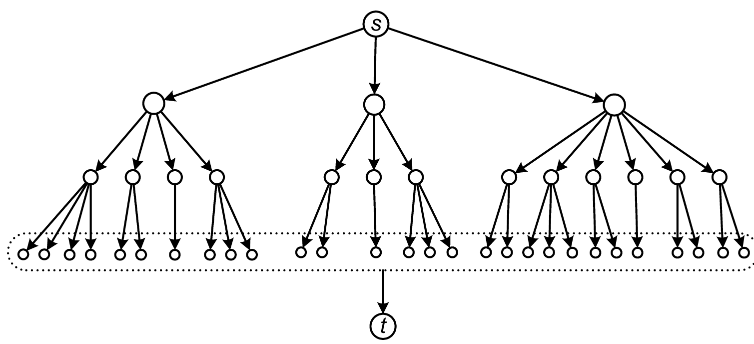


Fig. 6. Network 6.

paths, each of different length. Finally, Network 9 is an arbitrary nonlayered network.

6.2.2 *Distributions 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 4.1, we focused on instances where some usershave exponential weights. More specifically, we

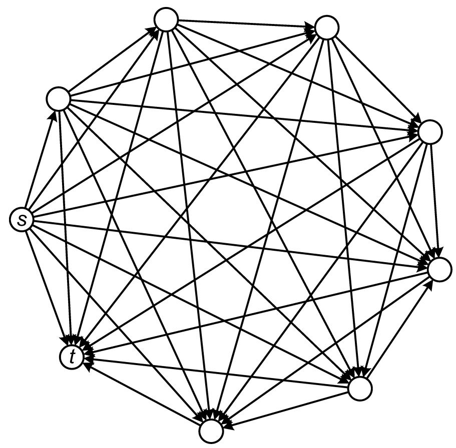


Fig. 7. Network 7.

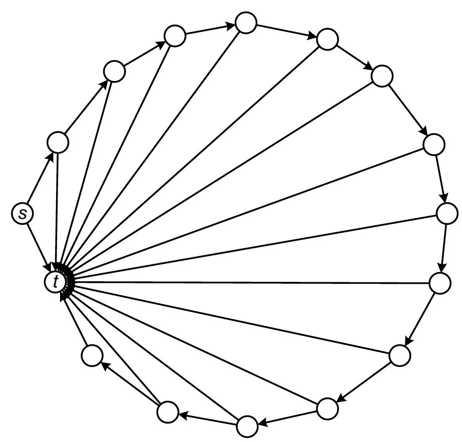


Fig. 8. Network 8.

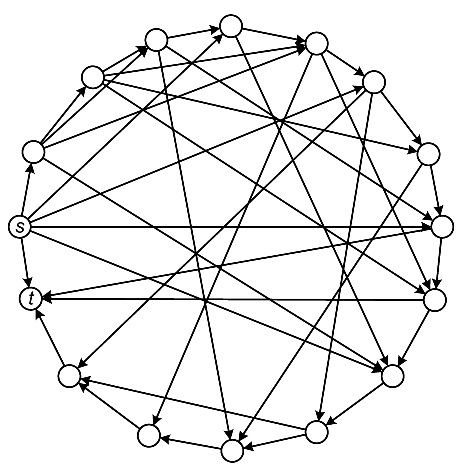


Fig. 9. Network 9.

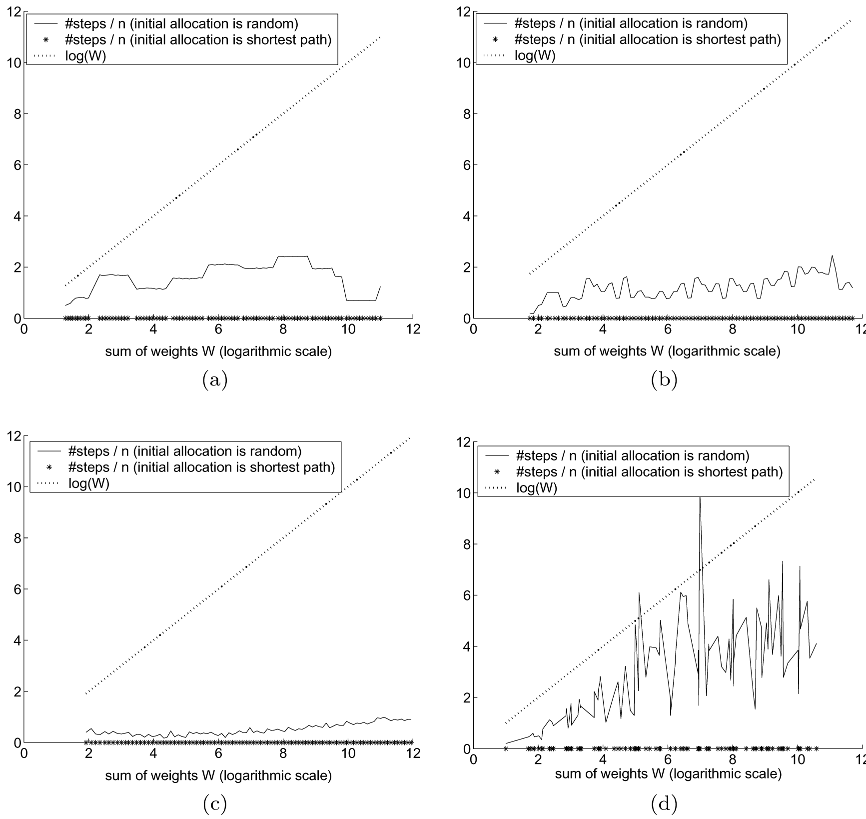


Fig. 10. Experimental results for Network 1.

considered the following four distributions of weights:

1. 10% of users have weight $10^{n/10}$ and 90% of users have weight 1,
2. 50% of users have weight $10^{n/10}$ and 50% of users have weight 1,
3. 90% of users have weight $10^{n/10}$ and 10% of users have weight 1,
4. users have uniformly at random selected weights in the interval $[1, 10^{n/10}]$.

Distributions (1–3), 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.

6.3 Results and Conclusions

Figures 10–18 show, for each network and each one of the distributions of weights (1–4), 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.

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

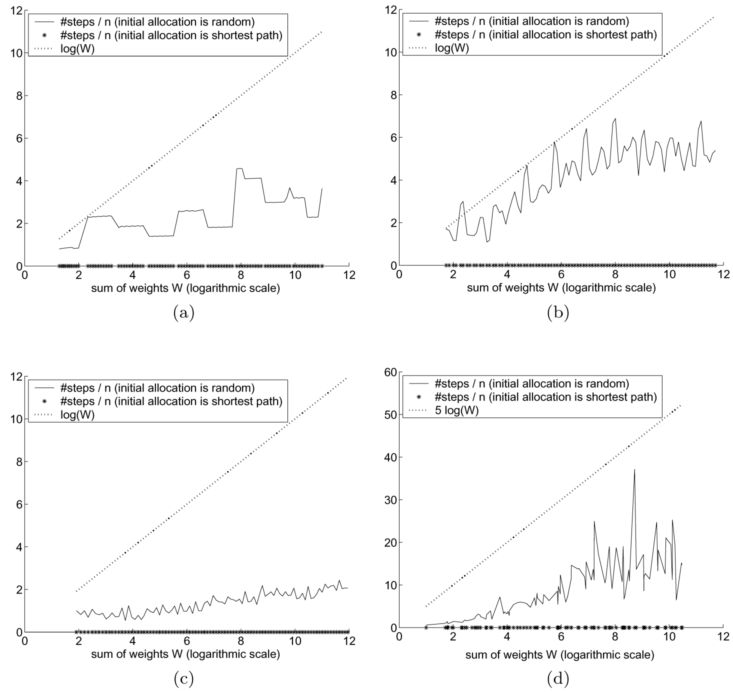


Fig. 11. Experimental results for Network 2.

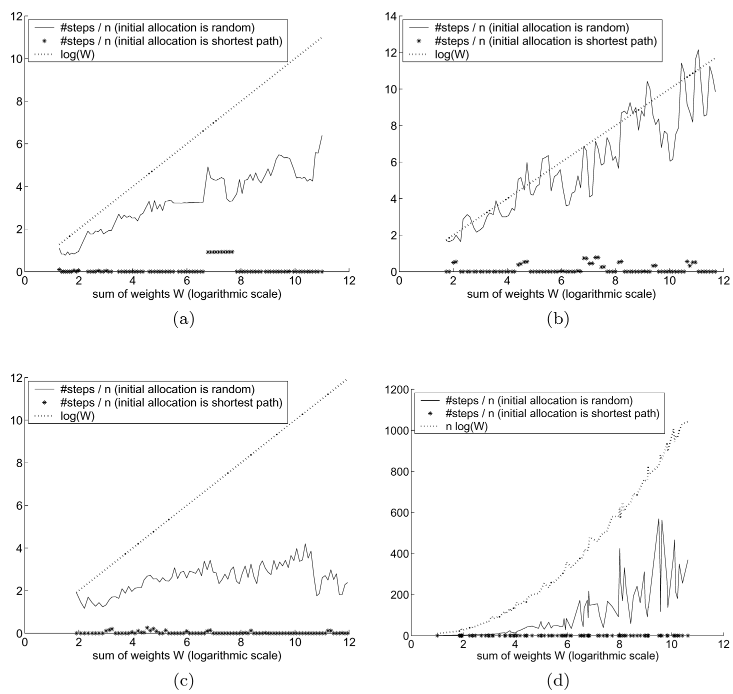


Fig. 12. Experimental results for Network 3.

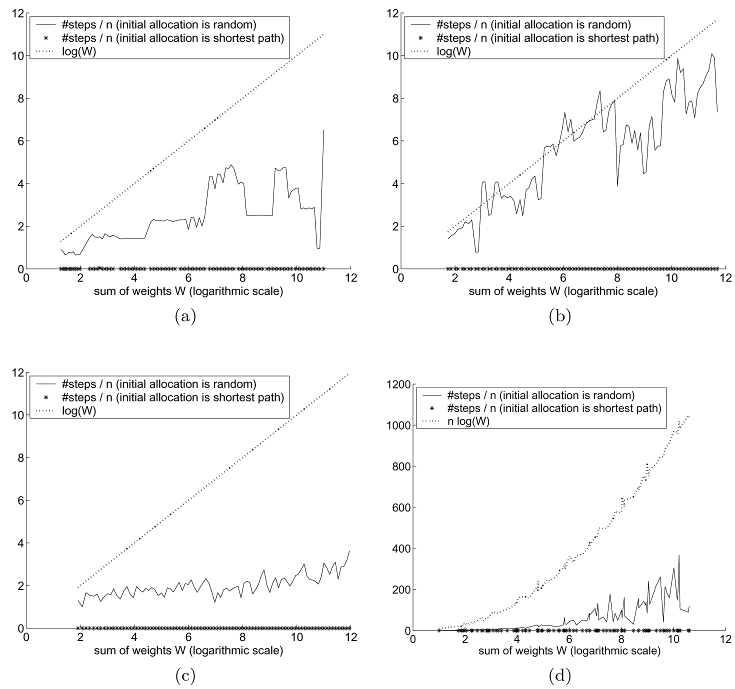


Fig. 13. Experimental results for Network 4.

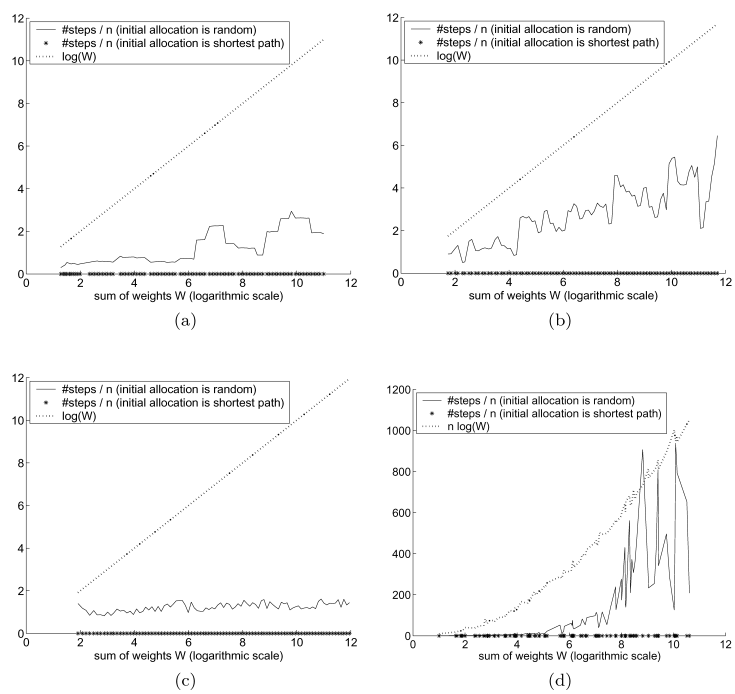
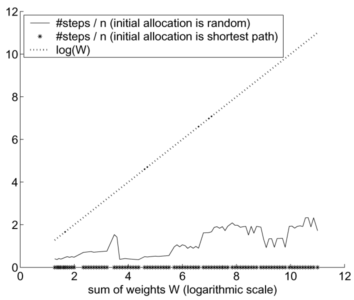
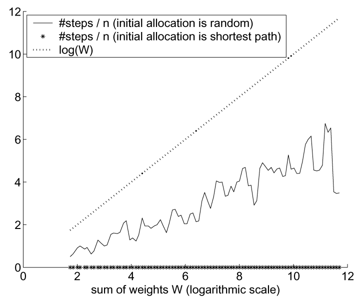


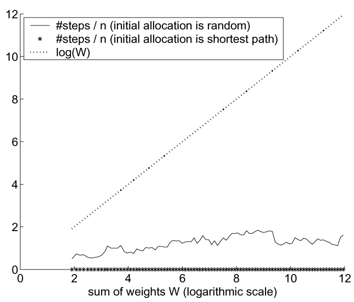
Fig. 14. Experimental results for Network 5.



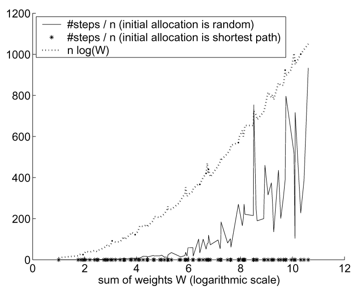
(a)



(b)

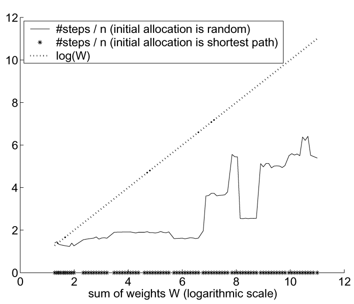


(c)

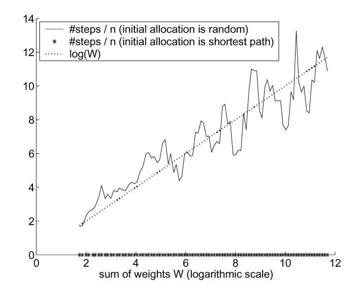


(d)

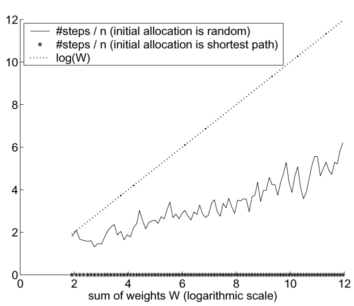
Fig. 15. Experimental results for Network 6.



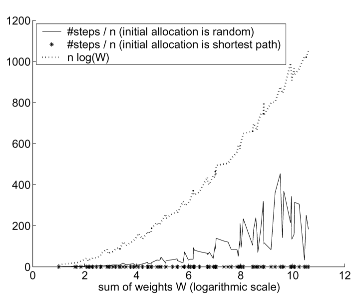
(a)



(b)



(c)



(d)

Fig. 16. Experimental results for Network 7.

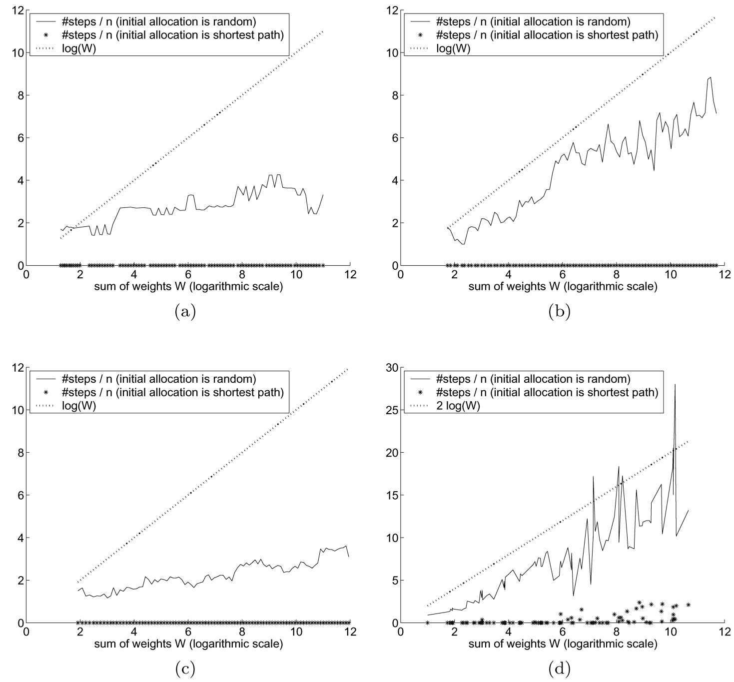


Fig. 17. Experimental results for Network 8.

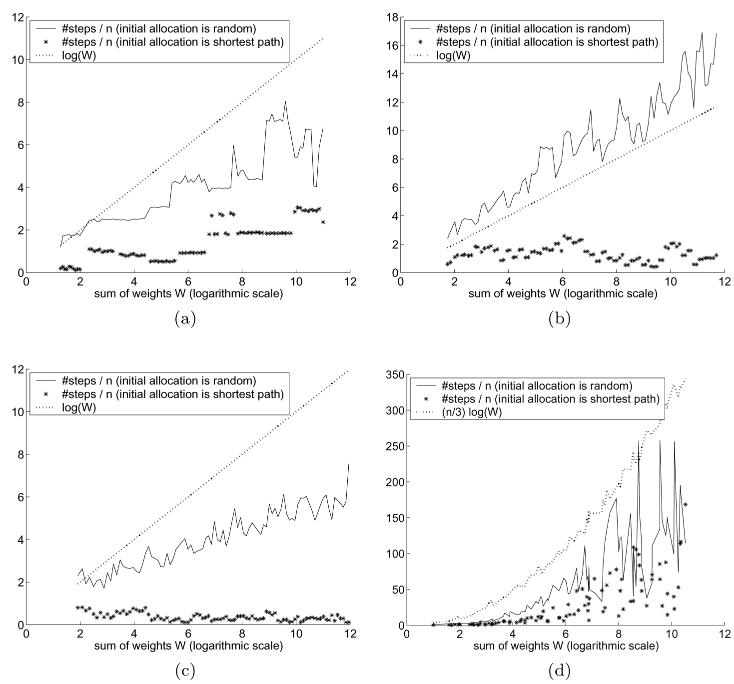


Fig. 18. Experimental results for Network 9.

appears to be a pure Nash equilibrium for sparse (Networks 1 and 2), grid (Network 4) and treelike networks (Networks 5 and 6), as well as for the clique (Network 7). As regards the dense-layered network (Network 3) and the non-layered Networks 8 and 9, the number of steps over the number of users seems to be bounded by a small constant.

On the other hand, the behavior of the algorithm when beginning with an arbitrary allocation is considerably worse. First, note that, in this case, the fluctuations observed at the plots are due to the randomization of the initial allocation. On average, however, we can make safe conclusions regarding the way # steps/n increases as a function of W. For the distributions of weights (1–3), 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 $= O(n \log(W))$. Unfortunately, the same does not seem to hold for randomly selected weights (distribution 4). In this case, however, as Figures 10–18d show, $n \log(W)$ seems to be a good asymptotic upper bound for #steps/n, suggesting that #steps $= O(n^2 \log(W))$.

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 two distinct weights (distributions 1–3). Hence, we conjecture that the complexity of the algorithm actually depends 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 Figures 10 and 11 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 are exponential in comparison to that of Network 1, we would expect a significant increase in the number of steps performed by the algorithm. Figures 10 and 11, however, show that this is not the case. Instead, the number of steps required for Network 2 are, at most, five times the number of steps required for Network 1.

Summarizing our results, we conclude that

- a shortest-path initial allocation is usually a few greedy selfish steps far from a pure Nash equilibrium, amplifying Conjecture 3.3, while
- an arbitrary initial allocation does not assure a similarly fast convergence to a pure Nash equilibrium; however, Conjecture 3.3 seems to be valid for this case as well, and
- the worst-case input for an arbitrary initial allocation occurs when all users' weights are distinct and some of them are exponential.

REFERENCES

EVEN-DAR, E., KESSELMAN, A., AND MANSOUR, Y. 2003. Convergence time to Nash equilibria. In *ICALP*. 502–513.

FABRIKANT, A., PAPADIMITRIOU, C. H., AND TALWAR, K. 2004. The complexity of pure Nash equilibria. In *Proc. of the 36th ACM Symposium on Theory of Computing*. 281–290.

FELDMANN, R., GAIRING, M., LÜCKING, T., MONIEN, B., AND RODE, M. 2003. Nashification and the coordination ratio for a selfish routing game. In *ICALP*. 514–526.

Fotakis, D., Kontogiannis, S., Koutsoupias, E., Mavronicolas, M., and Spirakis, P. 2002. The structure and complexity of Nash equilibria for a selfish routing game. In *Proc. of the 29th International Colloquium on Automata, Languages and Programming*. 123–134.

Fotakis, D., Kontogiannis, S., and Spirakis, P. 2005. Selfish unsplittable flows. In *Theoretical Computer Science (TCS) 348*, 2–3 (Dec.), 129–366. Preliminary version in the 31st International Colloquium on Automata, Languages and Programming (ICALP'04), pp. 593–605, 2004.

Fotakis, D., Kontogiannis, S., and Spirakis, P. 2006. Symmetry in network congestion games: Pure equilibria and anarchy cost. *Workshop on Approximation and online Algorithms - WAOA'05, LNCS 3879*, pp. 161–175.

Gairing, M., Lücking, T., Mavronicolas, M., and Monien, B. 2004. Computing Nash equilibria for scheduling on restricted parallel links. In *Proc. the 36th Annual ACM Symposium on Theory of Computing*. 613–622.

Libman, L. and Orda, A. 1997. Atomic resource sharing in noncooperative networks. In *INFOCOM*. 1006–1013.

Mehlhorn, K. and Naher, S. 1999. *LEDA: A Platform for Combinatorial and Geometric Computing*. Cambridge University Press, Cambridge.

Milchtaich, I. 1996. Congestion games with player-specific payoff functions. *Games and Economic Behavior 13*, 111–124.

Monderer, D. and Shapley, L. 1996. Potential games. *Games and Economic Behavior 14*, 124–143.

Nash, J. F. 1950. Equilibrium points in n-person games. *Proc. of National Academy of Sciences 36*, 48–49.

Rosenthal, R. W. 1973. A class of games poseessing pure-strategy Nash equilibria. *International Journal of Game Theory 2*, 65–67.

Received September 2005; revised January 2006; accepted January 2006