Algorithms for Pure Nash Equilibria in Weighted Congestion Games

Panagopoulou and Spirakis

Review by Chris Fawcett
CPSC 536H
April 2, 2008

Outline

- Game theory background.
- Bird's eye view of the paper.
- Weighted congestion games.
- Overview of the algorithm.
- Experimental design and empirical results.
- Comments and criticism.

Game Theory

- Models interaction between multiple agents in a structured system.
- Defined by:
  - A set of players.
  - A set of strategies for each player.
  - A payoff function for each player (a function of the strategy chosen).

Game Theory

- At each step of a game, each player is allowed to change strategies.
- Each player aims to maximise their own payoff function.

Game Theory

- A pure strategy for a given player uses only a single strategy at each step from the available set.
- A mixed strategy for a given player is a probability distribution over the set of available strategies.
- This paper only deals with pure strategies.

Game Theory

- A Nash Equilibrium is where:
  - No player can change strategies to improve their own payoff function.
  - Must assume the strategies of other players stay fixed.

Game Theory

- A Nash equilibrium is guaranteed to exist when players can use mixed strategies.
- If all players use pure strategies, a pure Nash equilibrium *may* exist.

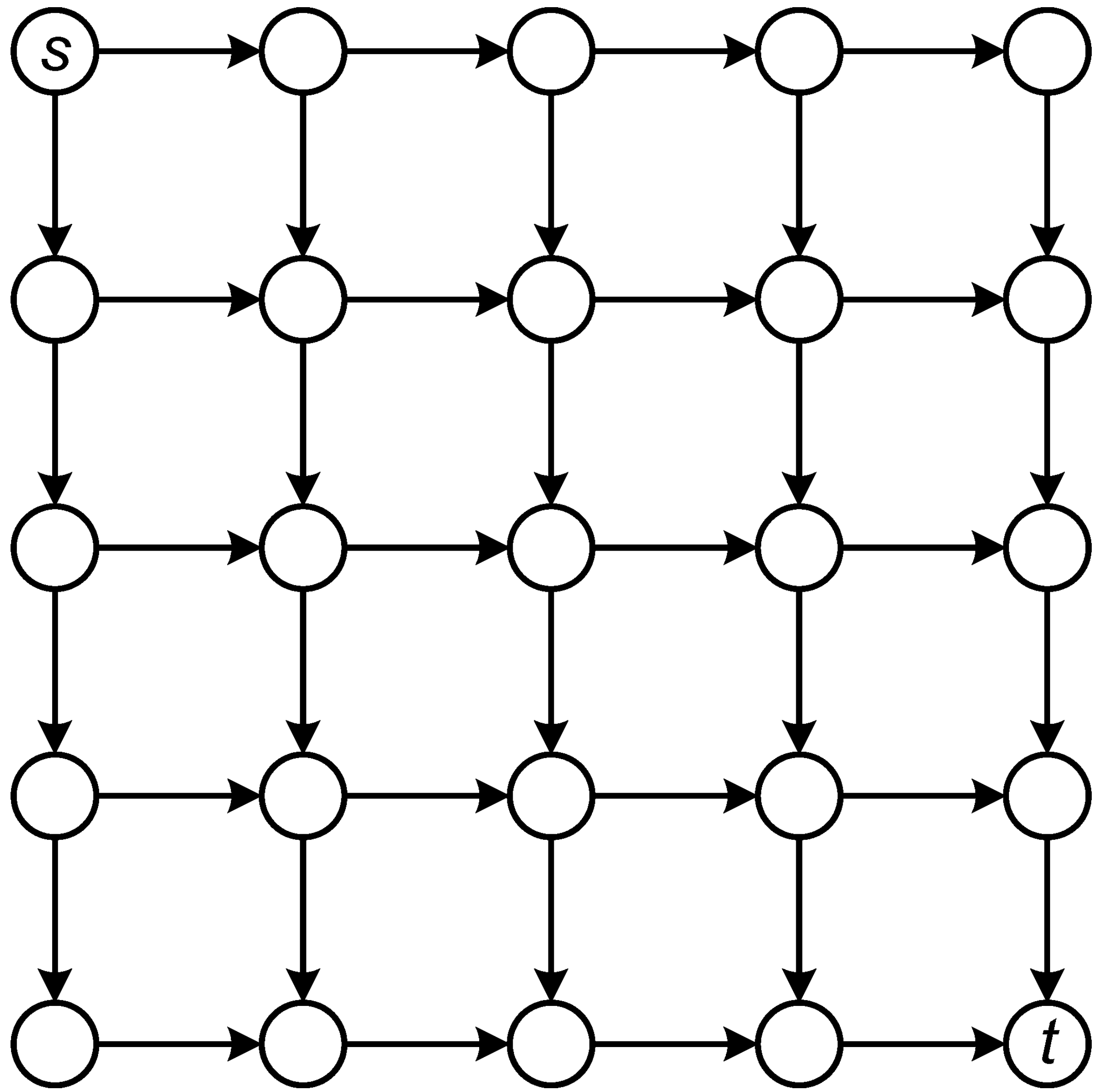
Bird's Eye View

- Weighted congestion games model the experience of users in a shared network.
- A pure Nash equilibrium always exists in these games.
- No mathematical proof that a pure Nash Equilibrium is computable in polynomial time for all instances.

Congestion Games

- Given a directed network G = (V,E)
- Every player wants to route traffic from a source node to a sink node in the network.
  - If these source/sink nodes are the same for every player, we have a single commodity network congestion game.
  - Strategy sets assumed to be equal.

Congestion Games



Congestion Games

- Each player has a set of paths from their source node to their sink node.
  - These are the strategies.
- The payoff for a given strategy is based on the the number of players sharing edges.

Weighted Version

- Each player can now demand more than one unit of traffic on a link.
- The delay on an edge is now a function of the demands of each user sharing that edge.

The Problem

- This paper considers only weighted, single-commodity network congestion games.
- Edge delays are allowed to be either polynomial or exponential in their loads (the sum of the demands).

Theoretical Results

- Proof is given that at least one pure Nash equilibrium always exists for these games.
- One of these equilibria can be computed in time polynomial in the number of players and the magnitude of the weights.

Theoretical Results

- It is conjectured by the authors that a pure Nash equilibrium is computable in polynomial time.
  - Even when the edge delays are exponential.

The Algorithm

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

Input: $\triangle$ network $G = (V, E)$ with a unique source–destination pair (s, t)
$\triangle$ 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. $\quad\varpi_i :=$ Shortest_Path$_i(\varpi_{-i})$
5. return ϖ
6. end

Experimental Design

- Nashify() was implemented in C++ using data structures in the LEDA library.
- Nine different networks of varying structure were used.
- Nashify() was run on each network, with {10,11,...,100} players.

Experimental Design

- Two different methods for choosing an initial set of strategies.
- Four different distributions of weights.

Initial Strategies

- Random Allocation
  - Each user assigns traffic on an s-t path chosen uniformly at random.
- Shortest-Path
  - Users sorted in non-increasing order of their demands.
  - Each selects the best possible s-t path, in order.

Weight Distributions

- Four different allocations of weights were examined.

1. 10% of players have weight $10^{n/10}$ and 90% of players have weight 1.
2. 50% of players have weight $10^{n/10}$ and 50% of players have weight 1.
3. 90% of players have weight $10^{n/10}$ and 10% of players have weight 1.
4. Each player has a weight selected uniformly at random from $[1, 10^{n/10}]$.

Networks Used

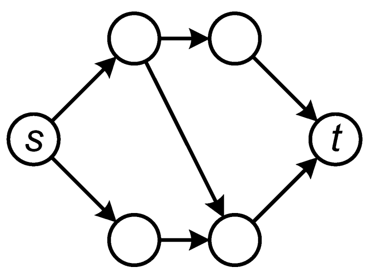


Fig. 1. Network 1.

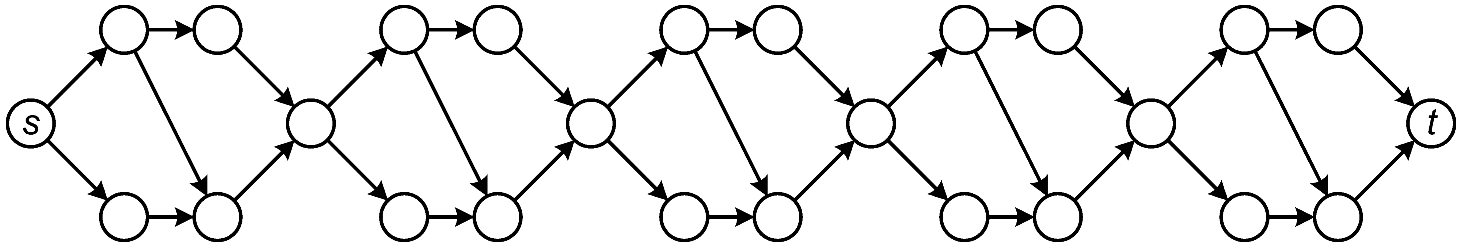


Fig. 2. Network 2.

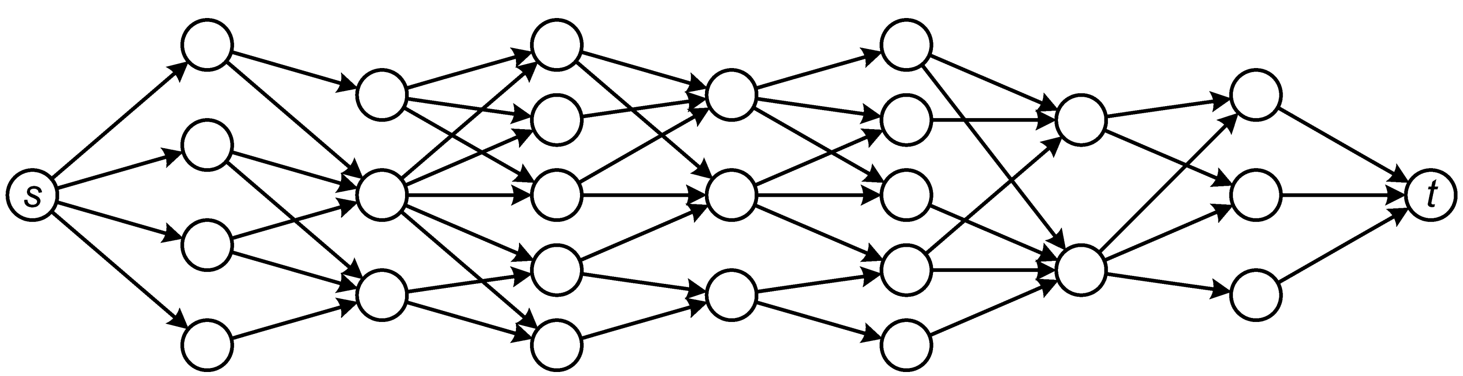


Fig. 3. Network 3.

Networks Used

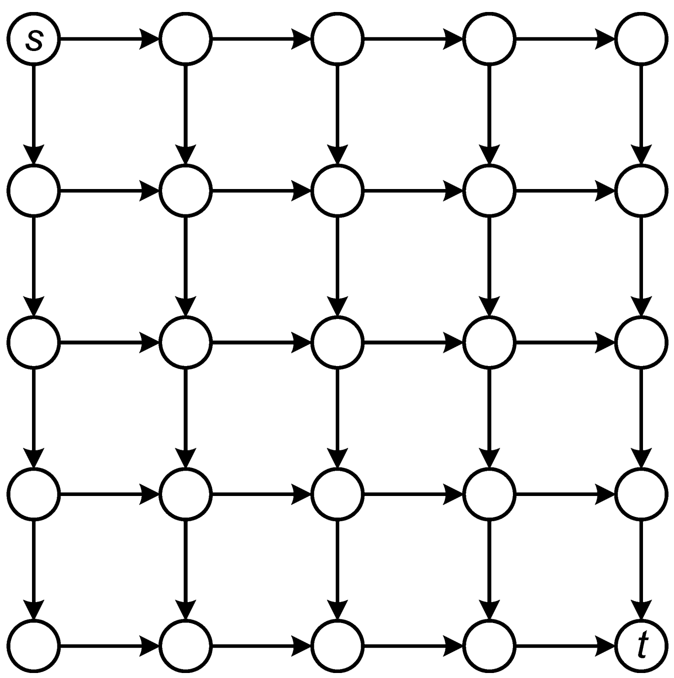


Fig. 4. Network 4.

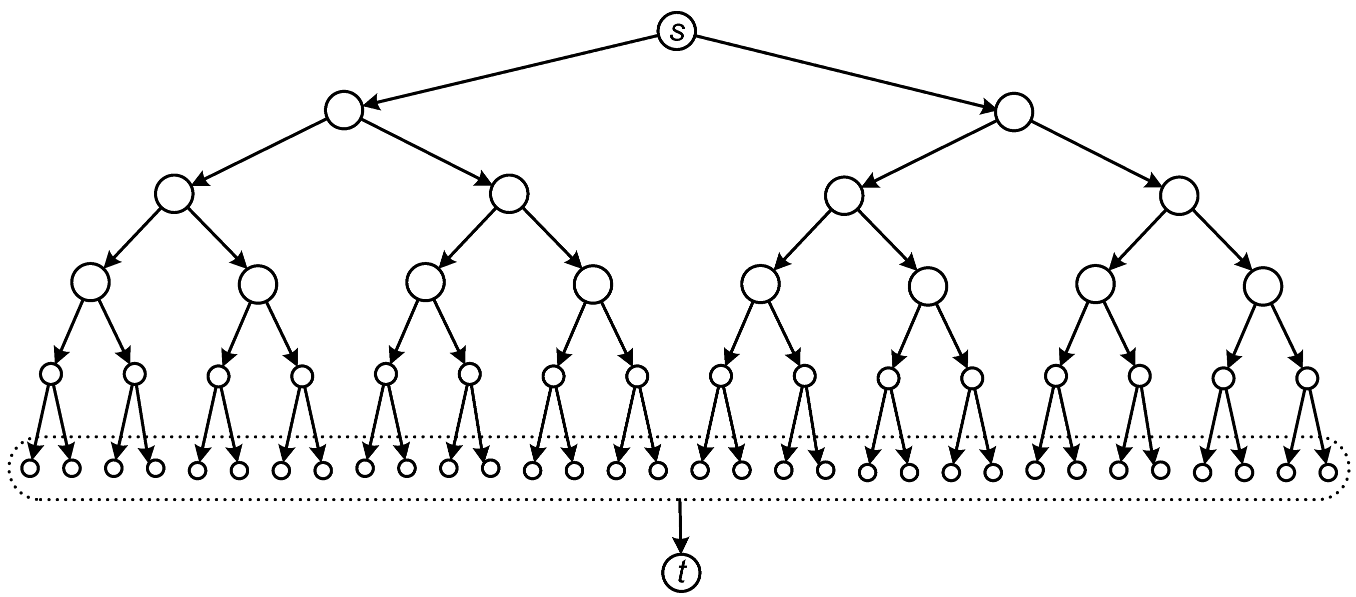
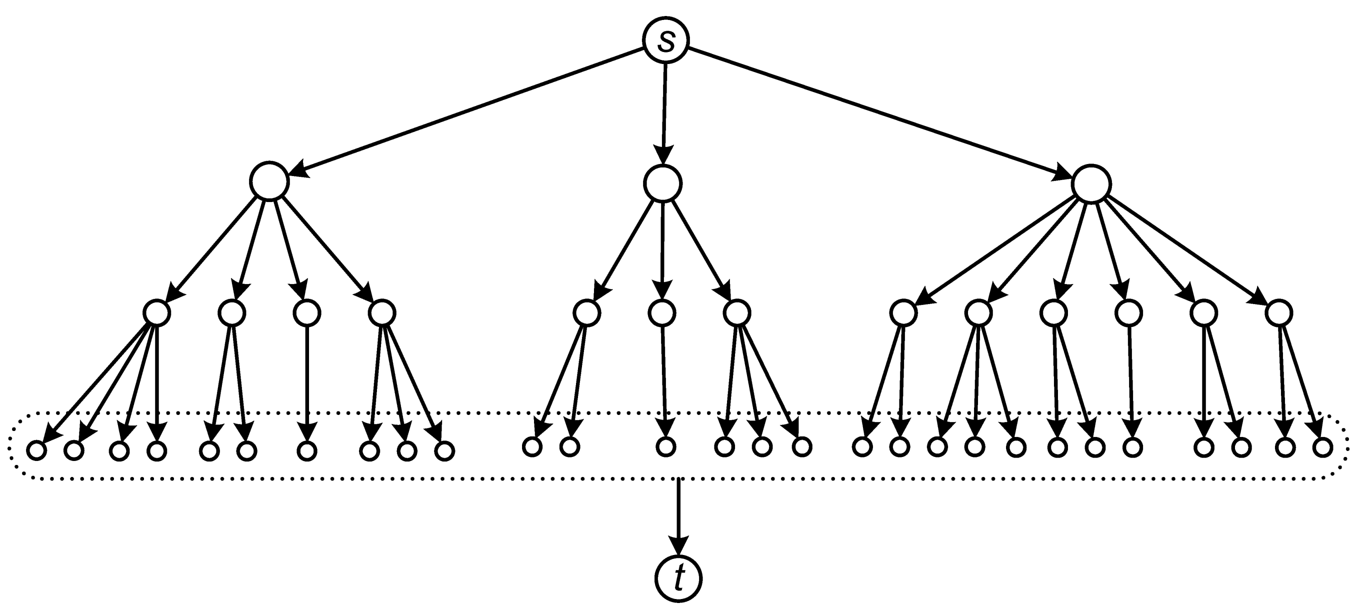


Fig. 5. Network 5.



Networks Used

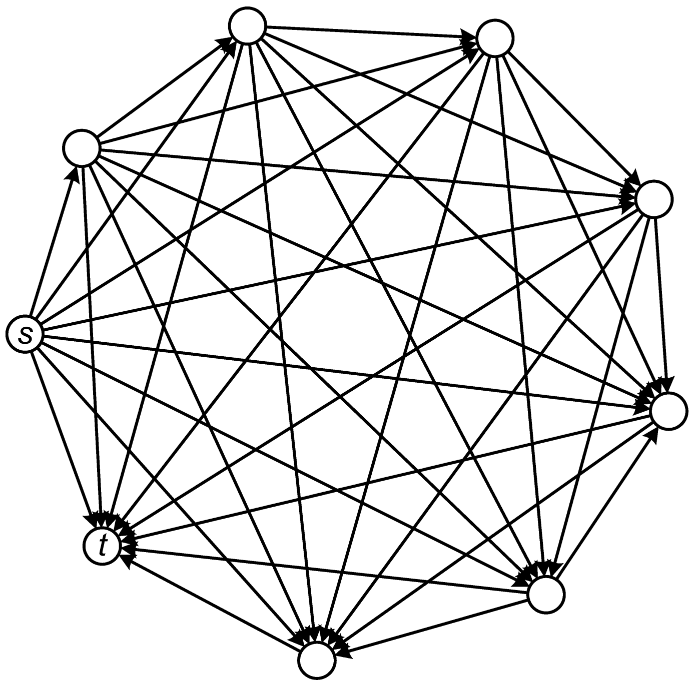


Fig. 7. Network 7.

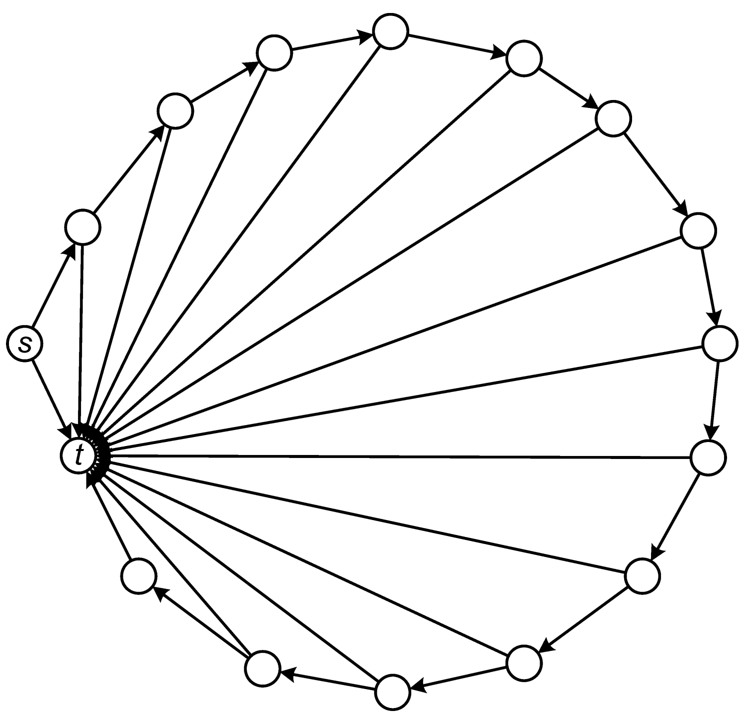


Fig. 8. Network 8.

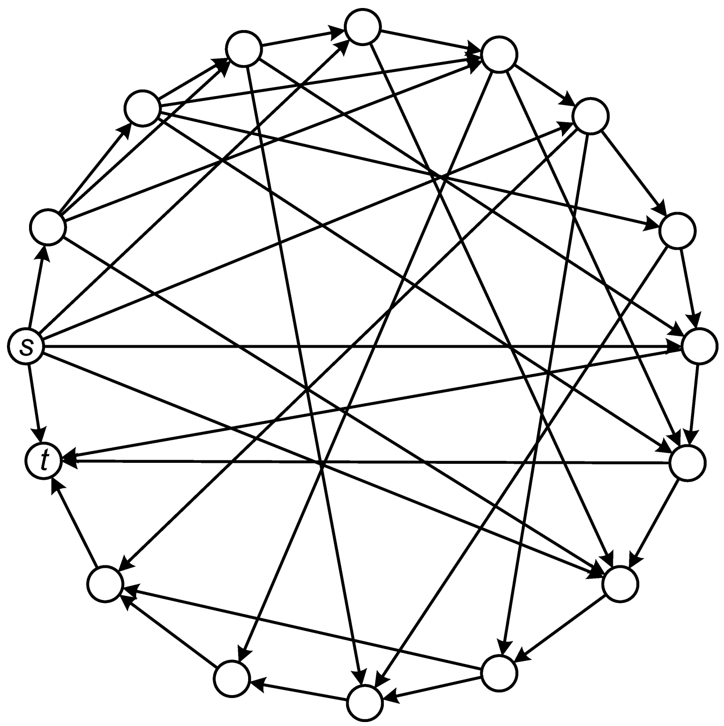


Fig. 9. Network 9.

Results

- Evidence suggests polynomial scaling on these nine networks.
- The shortest path allocation appears to dominate the random allocation.
- The authors conjecture that Nashify() will find a pure Nash equilibrium in a polynomial number of steps for any instance.

Results

- For weight distributions 1-3, #steps/n bounded above by log(W).
  - Implies O(nlog(W)) runtime.
- For weight distribution 4, #steps/n bounded above by nlog(W).
  - Implies $O(n^2\log(W))$ runtime.

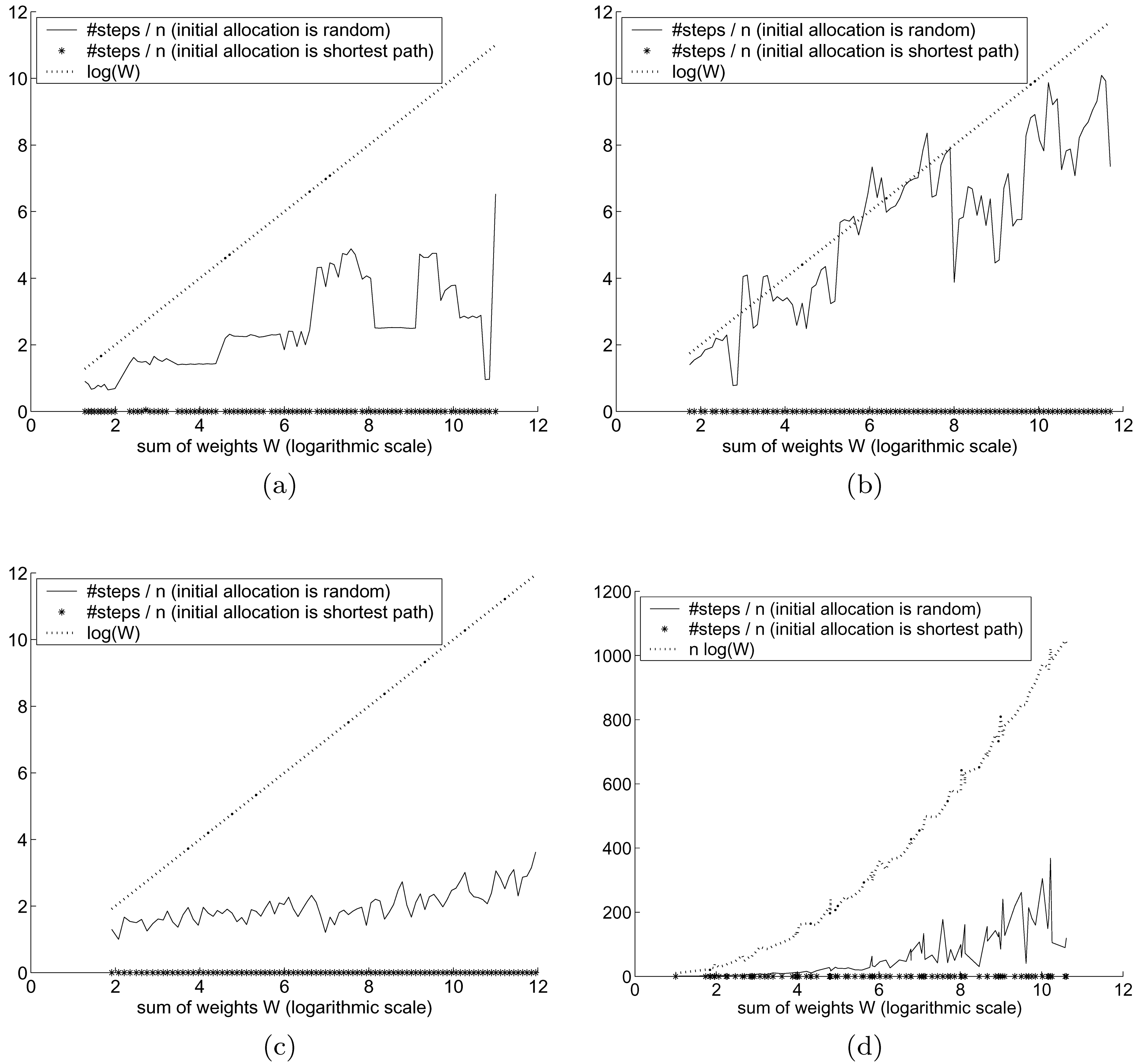


Fig. 13. Experimental results for Network 4.

Some Criticism

- May want to repeat outside of the linear spread of players (10-100).
  - Perhaps try n=200 and n=500 just to confirm.
- The networks tested had a narrow spread in terms of number of nodes.
  - What happens if we double the number of nodes in the same structures?

Some Criticism

- The experimental environment is never described in any detail whatsoever.
- The computation time is measured in terms of steps, with each step assumed to be a single greedy path selection.
- Should at least mention the basic machine characteristics for reproducibility.

Some Criticism

- The log(W) comparison for each network was different.
  - Compared against log(W), nlog(W), 2log(W), (n/3)log(W).
- Made comparing between networks difficult.

Some Criticism

- This appears to be a manual guess of the fit for each network structure.
- Would have been more informative to do an automatic fit and compare between the structures.

Questions?