Computing the Banzhaf Power Index in Network Flow Games

Yoram Bachrach Jeffrey S. Rosenschein
School of Engineering and Computer Science
The Hebrew University of Jerusalem, Israel
{yori,jeff}@cs.huji.ac.il

ABSTRACT

Preference aggregation is used in a variety of multiagent applications, and as a result, voting theory has become an important topic in multiagent system research. However, power indices (which reflect how much "real power" a voter has in a weighted voting system) have received relatively little attention, although they have long been studied in political science and economics. The Banzhaf power index is one of the most popular; it is also well-defined for any simple coalitional game.

In this paper, we examine the computational complexity of calculating the Banzhaf power index within a particular multiagent domain, a network flow game. Agents control the edges of a graph; a coalition wins if it can send a flow of a given size from a source vertex to a target vertex. The relative power of each edge/agent reflects its significance in enabling such a flow, and in real-world networks could be used, for example, to allocate resources for maintaining parts of the network.

We show that calculating the Banzhaf power index of each agent in this network flow domain is #P-complete. We also show that for some restricted network flow domains there exists a polynomial algorithm to calculate agents' Banzhaf power indices.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity;
I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent Systems*;
J.4 [**Computer Applications**]: Social and Behavioral Sciences—*Economics*

General Terms

Algorithms, Theory, Economics

Keywords

Computational complexity, Voting, Power indices

1. INTRODUCTION

Social choice theory can serve as an appropriate foundation upon which to build multiagent applications. There is a rich literature on the subject of voting[1] from political science, mathematics, and economics, with important theoretical results, and builders of automated agents can benefit from this work as they engineer systems that reach group consensus.

Interest in the theory of economics and social choice has in fact become widespread throughout computer science, because it is recognized as having direct implications on the building of systems comprised of multiple automated agents [16, 4, 22, 17, 14, 8, 15]. What distinguishes computer science work in these areas is its concern for computational issues: how are results arrived at (e.g., equilibrium points)? What is the complexity of the process? Can complexity be used to guard against unwanted phenomena? Does complexity of computation prevent realistic implementation of a technique?

The practical applications of voting among automated agents are already widespread. Ghosh et al. [6] built a movie recommendation system; a user's preferences were represented as agents, and movies to be suggested were selected through agent voting. Candidates in virtual elections have also been beliefs, joint plans [5], and schedules [7]. In fact, to see the generality of the (automated) voting scenario, consider modern web searching. One of the most massive preference aggregation schemes in existence is Google's PageRank algorithm, which can be viewed as a vote among indexed web pages on candidates determined by a user-input search string; winners are ranked (Tennenholtz and Altman [21] consider the axiomatic foundations of ranking systems such as this).

In this paper, we consider a topic that has been less studied in the context of automated agent voting, namely power indices. A power index is a measure of the power that a subgroup, or equivalently a voter in a weighted voting environment, has over decisions of a larger group. The Banzhaf power index is one of the most popular measures of voting power, and although it has been used primarily for measuring power in weighted voting games, it is well-defined for any simple coalitional game.

We look at some computational aspects of the Banzhaf power index in a specific environment, namely a network flow game. In this game, a coalition of agents wins if it can send a flow of size k from a source vertex s to a target vertex t, with the relative power of each edge reflecting its significance in allowing such a flow. We show that calculating the Banzhaf power index of each agent in this general network flow domain is #P-complete. We also show that for some restricted network flow domains (specifically, of con-

[1]We use the term in its intuitive sense here, but in the social choice literature, "preference aggregation" and "voting" are basically synonymous.

nectivity games on bounded layer graphs), there *does exist* a polynomial algorithm to calculate the Banzhaf power index of an agent. There are implications in this scenario to real-world networks; for example, the power index might be used to allocate maintenance resources (a more "powerful" edge being more critical), in order to maintain a given flow of data between two points.

The paper proceeds as follows. In Section 2 we give some background concerning coalitional games and the Banzhaf power index, and in Section 3 we introduce our specific network flow game. In Section 4 we discuss the Banzhaf power index in network flow games, presenting our complexity result in the general case. In Section 5 we consider a restricted case of the network flow game, and present results. In Section 6 we discuss related work, and we conclude in Section 7.

2. TECHNICAL BACKGROUND

A coalitional game is composed of a set of n agents, I, and a function mapping any subset (coalition) of the agents to a real value $v : 2^I \rightarrow \mathbb{R}$. In a *simple* coalitional game, v only gets values of 0 or 1 ($v : 2^I \rightarrow \{0, 1\}$). We say a coalition $C \subset I$ *wins* if $v(C) = 1$, and say it *loses* if $v(C) = 0$. We denote the set of all winning coalitions as $W(v) = \{C \subset 2^I | v(C) = 1\}$.

An agent i is a *swinger* (or "pivot") in a winning coalition C if the agent's removal from that coalition would make it a losing coalition: $v(C) = 1$, $v(C \setminus \{i\}) = 0$. A *swing* is a pair $< i, S >$ such that agent i is a swinger in coalition S.

A question that arises in this context is that of measuring the influence a given agent has on the outcome of a simple game. One approach to measuring the power of individual agents in simple coalitional games is the Banzhaf index.

2.1 The Banzhaf Index

A common interpretation of the power an agent possesses is that of its *a priori* probability of having a significant role in the game. Different assumptions about the formation of coalitions, and different definitions of "having a significant role," have caused researchers to define different power indices, one of the most prominent of which is the Banzhaf index [1]. This index has been widely used, though primarily for the purpose of measuring individual power in a weighted voting system. However, it can also easily be applied to any simple coalitional game.

The Banzhaf index depends on the number of coalitions in which an agent is a swinger, out of all possible coalitions.[2] The Banzhaf index is given by $\beta(v) = (\beta_1(v), ..., \beta_n(v))$ where

$$\beta_i(v) = \frac{1}{2^{n-1}} \sum_{S \subset N | i \in S} [v(S) - v(S \setminus \{i\})].$$

Different probabilistic models on the way a coalition is formed yield different appropriate power indices [20]. The Banzhaf power index reflects the assumption that the agents are independent in their choices.

3. NETWORK FLOW GAMES

3.1 Motivation

Consider a communication network, where it is crucial to be able to send a certain amount of information between two sites. Given limited resources to maintain network links, which edges should get those resources?

[2]Banzhaf actually considered the percentage of such coalitions out of all *winning* coalitions. This is called the *normalized* Banzhaf index.

We model this problem by considering a *network flow game*. The game consists of agents in a network flow graph, with a certain source vertex s and target vertex t. Each agent controls one of the graph's edges, and a coalition of agents controls all the edges its members control. A coalition of agents wins the game if it manages to send a flow of at least k from source s to target t, and loses otherwise.

To ensure that the network is capable of maintaining the desired flow between s and t, we may choose to allocate our limited maintenance resources to the edges according to their impact on allowing this flow. In other words, resources could be devoted to the links whose failure is most likely to cause us to lose the ability to send the required amount of information between the source and target.

Under a reasonable probabilistic model, the Banzhaf index provides us with a measure of the impact each edge has on enabling this amount of information to be sent between the sites, and thus provides a reasonable basis for allocation of scarce maintenance resources.

3.2 Formal Definition

Formally, a network flow game is defined as follows. The game consists of a network flow graph $G =< V, E >$, with capacities on the edges $c : E \rightarrow \mathbb{R}$, a source vertex s, a target vertex t, and a set I of agents, where agent i controls the edge e_i. Given a coalition C, which controls the edges $E_C = \{e_i | i \in C\}$, we can check whether the coalition allows a flow of k from s to t. We define the *simple coalitional game of network flow* as the game where the coalition wins if it allows such a flow, and loses otherwise:

$$v(C) = \begin{cases} 1 & \text{if } E_C \text{ allows a flow of } k \text{ from } s \text{ to } t; \\ 0 & \text{otherwise;} \end{cases}$$

A simplified version of the network flow game is the *connectivity game*; in a connectivity game, a coalition wants to have some path from source to target. More precisely, a connectivity game is a network flow game where each of the edges has identical capacity, $c(e) = 1$, and the target flow value is $k = 1$. In such a scenario, the goal of a coalition is to have at least one path from s to t:

$$v(C) = \begin{cases} 1 & \text{if } E_C \text{ contains a path from } s \text{ to } t; \\ 0 & \text{otherwise;} \end{cases}$$

Given a network flow game (or a connectivity game), we can compute the power indices of the game. When a coalition of edges is chosen at random, and each coalition is equiprobable, the appropriate index is the Banzhaf index.[3] We can use the Banzhaf value of an agent $i \in I$ (or the edge it controls, e_i), $\beta_{e_i}(v) = \beta_i(v)$, to measure its impact on allowing a given flow between s and t.

4. THE BANZHAF INDEX IN NETWORK FLOW GAMES

We now define the problem of calculating the Banzhaf index in the network flow game.

DEFINITION 1. *NETWORK-FLOW-BANZHAF: We are given a network flow graph $G =< V, E >$ with a source vertex s and a target vertex t, a capacity function $c : E \rightarrow \mathbb{R}$, and a target flow value k. We consider the network flow game, as defined above in Section 3. We are given an agent i, controlling the edge e_i, and are asked to calculate the Banzhaf index for that agent. In the network*

[3]When each *ordering* of edges is equiprobable, the appropriate index is the Shapley-Shubik index.

flow game, let C_{e_i} be the set of all subsets of E that contain e_i: $C_{e_i} = \{C \subset E | e_i \in C\}$. In this game, the Banzhaf index of e_i is:

$$\beta_i(v) = \frac{1}{2^{|E|-1}} \sum_{E' \subset C_{e_i}} [v(E') - v(E' \setminus \{e_i\})].$$

Let $W(C_{e_i})$ be the set of winning subsets of edges in C_{e_i}, i.e., the subsets $E' \in C_{e_i}$ where a flow of at least k can be sent from s to t using only the edges in E'. The Banzhaf index of e_i is the proportion of subsets in $W(C_{e_i})$ where e_i is crucial *to maintaining the k-flow. All the edge subsets in $W(C_{e_i})$ contain e_i and are winning, but only for some of them, $E' \in W(C_{e_i})$, do we have that $v(E' \setminus \{e_i\}) = 0$ (i.e., E' is no longer winning if we remove e_i). The Banzhaf index of e_i is the proportion of such subsets.*

4.1 #P-Completeness of Calculating the Banzhaf Index in the Network Flow Game

We now show that the general case of NETWORK-FLOW-BANZHAF is #P-complete, by a reduction from #MATCHING.

First, we note that NETWORK-FLOW-BANZHAF is in #P. There are several polynomial algorithms to calculate the maximal network flow, so it is easy to check if a certain subset of edges $E' \subset E$ contains e_i and allows a flow of at least k from s to t. It is also easy to check if a flow of at least k is no longer possible when we remove e_i from E' (again, by running a polynomial algorithm for calculating the maximal flow). The Banzhaf index of e_i is exactly the number of such subsets $E' \subset E$, so NETWORK-FLOW-BANZHAF is in #P. To show that NETWORK-FLOW-BANZHAF is #P-complete, we reduce a #MATCHING problem[4] to a NETWORK-FLOW-BANZHAF problem.

DEFINITION 2. *#MATCHING: We are given a bipartite graph $G =< U, V, E >$, such that $|U| = |V| = n$, and are asked to count the number of perfect matchings possible in G.*

4.2 The Overall Reduction Approach

The reduction is done as follows. From the #MATCHING input, $G =< U, V, E >$, we build two inputs for the NETWORK-FLOW-BANZHAF problem. The difference between the answers obtained from the NETWORK-FLOW-BANZHAF runs is the answer to the #MATCHING problem. Both runs of the NETWORK-FLOW-BANZHAF problem are constructed with the same graph $G' =< V', E' >$, with the same source vertex s and target vertex t, and with the same edge e_f for which to compute the Banzhaf index. They differ only in the target flow value. The first run is with a target flow of k, and the second run is with a target flow of $k + \epsilon$.

A choice of subset $E_c \subset E'$ reflects a possible matching in the original graph. G is a subgraph of the constructed G'. We identify an edge in G', $e \in E'$, with the same edge in G. This edge indicates a particular match between some vertex $u \in U$ and another vertex $v \in V$. Thus, if $E_c \subset E'$ is a subset of edges in G' which contains only edges in the subgraph of G, we identify it with a subset of edges in G, or with some candidate of a matching.

We say $E_c \subset E'$ *matches* some vertex $v \in V$, if E_c contains some edge that connects to v, i.e., for some $u \in U$ we have $(u, v) \in E_c$. E_c is a possible matching if it does not match a vertex $v \in V$ with more than one vertex in U, i.e., there are not two vertices $u_1 \neq u_2$ in U that both $(u_1, v) \in E_c$ and $(u_2, v) \in E_c$. A perfect matching matches all the vertices in V.

If E_c fails to match a vertex in V (the right side of the partition), the maximal possible flow that E_c allows in G' is less than k. If it matches all the vertices in V, a flow of k is possible. If it matches all the vertices in V, but matches some vertex in V more than once (which means this is not a true matching), a flow of $k+\epsilon$ is possible. ϵ is chosen so that if a single vertex $v \in V$ is unmatched, the maximal possible flow would be less than $|V|$, even if all the other vertices are matched more than once. In other words, ϵ is chosen so that matching several vertices in V more than once can never compensate for not matching some vertex in V, in terms of the maximal possible flow.

Thus, when we check the Banzhaf index of e_f when the required flow is at least k, we get the number of subsets $E' \subset E$ that match all the vertices in V at least once. When we check the Banzhaf index of e_f with a required flow of at least $k+\epsilon$, we get the number of subsets $E' \subset E$ that match all the vertices in V at least once, and match at least one vertex $v \in V$ more than once. The difference between the two is exactly the number of perfect matchings in G.

Therefore, if there existed a polynomial algorithm for NETWORK-FLOW-BANZHAF, we could use it to build a polynomial algorithm for #MATCHING, so NETWORK-FLOW-BANZHAF is #P-complete.

4.3 Reduction Details

The reduction takes the #MATCHING input, the bipartite graph $G =< U, V, E >$, where $|U| = |V| = k$. It then generates a network flow graph G' as follows. The graph G is kept as a subgraph of G', and each edge in G is given a capacity of 1. A new source vertex s is added, along with a new vertex t' and a new target vertex t. Let $\epsilon = \frac{1}{k+1}$ so that $\epsilon \cdot k < 1$. The source s is connected to each of the vertices in U, the left partition of G, with an edge of capacity $1 + \epsilon$. Each of the vertices in V is connected to t' with an edge of capacity $1 + \epsilon$. t' is connected to t with an edge e_f of capacity $1 + \epsilon$.

As mentioned above, we perform two runs of NETWORK-FLOW-BANZHAF, both checking the Banzhaf index of the edge e_f in the flow network G'. We denote the network flow game defined on G' with target flow k as $v_{(G',k)}$. The first run is performed on the game with a target flow of k, $v_{(G',k)}$, returning the index $\beta_{e_f}(v_{(G',k)})$. The second run is performed on the game with a target flow of $k + \epsilon$, $v_{(G',k+\epsilon)}$, returning the index $\beta_{e_f}(v_{(G',k+\epsilon)})$. The number of perfect matchings in G is the difference between the answers in the two runs, $\beta_{e_f}(v_{(G',k)}) - \beta_{e_f}(v_{(G',k+\epsilon)})$. This is proven in Theorem 5.

Figure 1 shows an example of constructing G' from G. On the left is the original graph G, and on the right is the constructed network flow graph G'.

4.4 Proof of the reduction

We now prove that the reduction above is correct. In all of this section, we take the input to the #MATCHING problem to be $G =< U, V, E >$ with $|U| = |V| = k$, the network flow graph constructed in the reduction to be $G' =< V', E' >$ with capacities $c : E' \to \mathbb{R}$ as defined in Section 4.3, the edge for which to calculate the Banzhaf index to be e_f, and target flow values of k and $k + \epsilon$.

PROPOSITION 1. *Let $E_c \subset E'$ be a subset of edges that lacks one or more edges of the following:*

1. *The edges connected to s;*
2. *The edges connected to t';*
3. *The edge $e_f = (t', t)$.*

We call such a subset a missing *subset. The maximal flow between s and t using only the edges in the missing subset E_c is less than k.*

[4]This is one of the most well-known #P-complete problems.

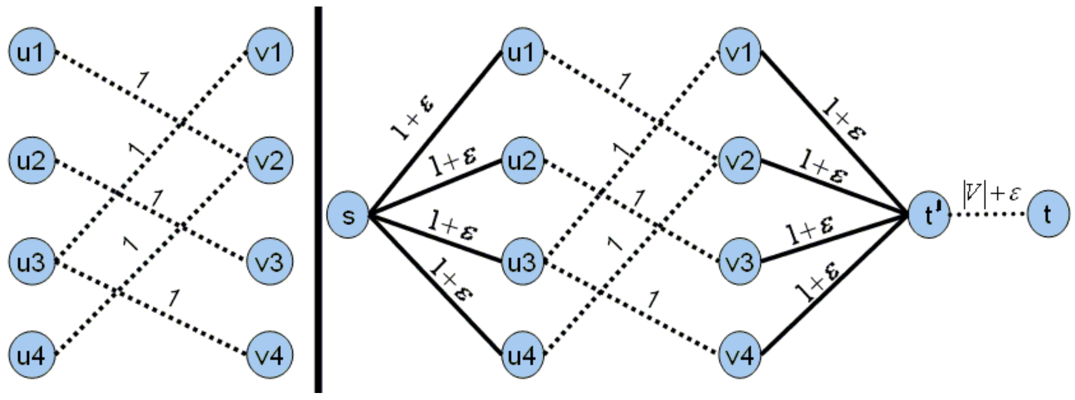

Figure 1: Reducing #MATCHING to NETWORK-FLOW-BANZHAF

PROOF. The graph is a layer graph, with s being the vertex in the first layer, U the vertices in the second layer, V the vertices in the third, t' the vertex in the fourth, and t in the fifth. Edges in G' only go between consecutive layers. The maximal flow in a layer graph is limited by the total capacity of the edges between every two consecutive layers. If any of the edges between s and U is missing, the flow is limited by $(|V| - 1)(1 + \epsilon) < k$. If any of the edges between V and t' is missing, the flow is also limited by $(|V| - 1)(1 + \epsilon) < k$. If the edge e_f is missing, there are no edges going to the last layer, and the maximal flow is 0. $\square$

Since such missing subsets of edges do not affect the Banzhaf index of e_f (they add 0 to the sum), from now on we will consider only non-missing subsets. As explained in Section 4.2, we identify the edges in G' that were copied from G (the edges between U and V in G') with their counterparts in G. Each such edge $(u, v) \in E'$ represents a match between u and v in G. E_c is a perfect matching if it matches every vertex u to a single vertex v and vice versa.

PROPOSITION 2. *Let $E_c \subset E'$ be a subset of edges that fails to match some vertex $v \in V$. The maximal flow between s and t using only the edges in the missing subset E_c is less than k. We call such a set* sub-matching, *and it is not a perfect matching.*

PROOF. If E_c fails to match some vertex $v \in V$, the maximal flow that can reach the vertices in the V layer is $(1+\epsilon)(k-1) < k$, so this is also the maximal flow that can reach t. $\square$

PROPOSITION 3. *Let $E_c \subset E'$ be a subset of edges that is a perfect matching in G. Then the maximal flow between s and t using only the edges in E_c is exactly k.*

PROOF. A flow of k is possible. We send a flow of 1 from s to each of the vertices in U, send a flow of 1 from each vertex $u \in U$ to its match $v \in V$, and send a flow of 1 from each $v \in V$ to t'. t' gets a total flow of exactly k, and sends it to t. A flow of more than k is not possible since there are exactly k edges of capacity 1 between the U layer and the V layer, and the maximal flow is limited by the total capacity of the edges between these two consecutive layers. $\square$

PROPOSITION 4. *Let $E_c \subset E'$ be a subset of edges that contains a perfect matching $M \subset E$ in G and at least one more edge e_x between some vertex $u_a \in U$ and $v_a \in V$. Then the maximal flow between s and t using only the edges in E_c is at least $k+\epsilon$. We call such a set* super-matching, *and it is not a perfect matching.*

PROOF. A flow of k is possible, by using the edges of the perfect match as in Proposition 3. We send a flow of 1 from s to each of the vertices in U, send a flow of 1 from each vertex $u \in U$ to its match $v \in V$, and send a flow of 1 from each $v \in V$ to t'. t' gets a total flow of exactly k, and sends it to t. After using the edges of the perfect matching, we send a flow of ϵ from s to u_a (this is possible since the capacity of the edge (s, u_a) is $1 + \epsilon$ and we have only used up 1). We then send a flow of ϵ from u_a to v_a. This is possible since we have not used this edge at all—it is the edge which is not a part of the perfect matching. We then send a flow of ϵ from v_a to t'. Again, this is possible since we have used 1 out of the total capacity of $1 + \epsilon$ which that edge has. Now t' gets a total flow of $k + \epsilon$, and sends it all to t, so we have achieved a total flow of $k + \epsilon$. Thus, the maximal possible flow is at least $k + \epsilon$. $\square$

THEOREM 5. *Consider a #MATCHING instance $G =< U, V, E >$ reduced to a BANZHAF-NETWORK-FLOW instance G' as explained in Section 4.3. Let $v_{(G',k)}$ be the network flow game defined on G' with target flow k, and $v_{(G',k+\epsilon)}$ be the game defined with a target flow of $k+\epsilon$. Let the resulting index of the first run be $\beta_{e_f}(v_{(G',k)})$, and $\beta_{e_f}(v_{(G',k+\epsilon)})$ be the resulting index of the second run. Then the number of perfect matchings in G is the difference between the answers in the two runs, $\beta_{e_f}(v_{(G',k)}) - \beta_{e_f}(v_{(G',k+\epsilon)})$.*

PROOF. Consider the game $v_{(G',k)}$. According to Proposition 1, in this game, the Banzhaf index of E_f does not count missing subsets $E_c \in E'$, since they are losing in this game. According to Proposition 2, it does not count subsets $E_c \in E'$ that are sub-matchings, since they are also losing. According to Proposition 3, it adds 1 to the count for each perfect matching, since such subsets allow a flow of k and are winning. According to Proposition 3, it adds 1 to the count for each super-matching, since such subsets allow a flow of k (and more than k) and are winning.

Consider the game $v_{(G',k+\epsilon)}$. Again, according to Proposition 1, in this game the Banzhaf index of E_f does not count missing subsets $E_c \in E'$, since they are losing in this game. According to Proposition 2, it does not count subsets $E_c \in E'$ that are sub-matchings, since they are also losing. According to Proposition 3, it adds 0 to the count for each perfect matching, since such subsets allow a flow of k but not $k + \epsilon$, and are thus losing. According to Proposition 3, it adds 1 to the count for each super-matching, since such subsets allow a flow of $k + \epsilon$ and are winning.

Thus the difference between the two indices,

$$\beta_{e_f}(v_{(G',k)}) - \beta_{e_f}(v_{(G',k+\epsilon)}),$$

is exactly the number of perfect matchings in G. $\square$

We have reduced a #MATCHING problem to a NETWORK-FLOW-BANZHAF problem. This means that given a polynomial algorithm to calculate the Banzhaf index of an agent in a general network flow game, we can build an algorithm to solve the #MATCHING problem. Thus, the problem of calculating the Banzhaf index of agents in general network flow games is also #P-complete.

5. CALCULATING THE BANZHAF INDEX IN BOUNDED LAYER GRAPH CONNECTIVITY GAMES

We here present a polynomial algorithm to calculate the Banzhaf index of an edge in a connectivity game, where the network is a bounded layer graph. This positive result indicates that for some *restricted* domains of network flow games, it is possible to calculate the Banzhaf index in a reasonable amount of time.

DEFINITION 3. *A* layer graph *is a graph $G =< V, E >$, with source vertex s and target vertex t, where the vertices of the graph are partitioned into $n + 1$ layers, $L_0 = \{s\}, L_1, ..., L_n = \{t\}$. The edges run only between consecutive layers.*

DEFINITION 4. *A* c-bounded *layer graph is a layer graph where the number of vertices in each layer is bounded by some constant number c.*

Although there is no limit on the number of layers in a bounded layer graph, the structure of such graphs makes it possible to calculate the Banzhaf index of edges in connectivity games on such graphs. The algorithm provided below is indeed polynomial in the number of vertices given that the network is a c-bounded layer graph. However, there is a constant factor to the running time, which is exponential in c. Therefore, this method is only tractable for graphs where the bound c is small. Bounded layer graphs may occur in networks when the nodes are located in several ordered segments, where nodes can be connected only between consecutive segments.

Let v be a vertex in layer L_i. We say an edge e occurs before v if it connects two vertices in v's layer or a previous layer: $e = (u, w)$ connects vertex $u \in L_j$ to vertex $w \in L_{j+1}$ and $j + 1 \leq i$. Let $Pred_v \subset E$ be the subset of edges that occur before v. Consider a subset of these edges, $E' \subset Pred_v$. E' may contain a path from s to v, or it may not. We define P_v as the number of subsets $E' \subset Pred_v$ that contain a path from s to v.

Similarly, let $V_i \in V$ be the subset of *all* the vertices in the same layer L_i. Let $Pred_{V_i} \subset E$ be the subset of edges that occur before V_i (all the vertices in V_i are in the same layer, so any edge that occurs before some $v \in V_i$ occurs before any other vertex $w \in V_i$). Consider a subset of these edges, $E' \subset Pred_V$. Let $V_i(E')$ be the subset of vertices in V_i that are reachable from s using only the edges in E': $V_i(E') = \{v \in V_i | E' \text{ contains a path from s to v}\}$. We say $E' \in Pred_V$ *connects exactly* the vertices in $S_i \subset V_i$ if all the vertices in S_i are reachable from s using the edges in E' but no other vertices in V_i are reachable from s using E', so $V_i(E') = S_i$.

Let $V' \subset V_i$ be a subset of the vertices in layer L_i. We define $P_{V'}$ as the number of subsets $E' \subset Pred_{V'}$ that *connect exactly* the vertices in V': $P_{V'} = |\{E' \subset Pred_{V'} | V_i(E') = V'\}|$.

LEMMA 1. *Let $S_1, S_2 \subset V_i$ where $S_1 \neq S_2$ be two different subsets of vertices in the same layer. Let $E', E'' \subset Pred_{V_i}$ be two sets of edge subsets, so that E' connects exactly the vertices in S_1 and E'' connects exactly the vertices in S_2: $V_i(E') = S_1$ and $V_i(E'') = S_2$. Then E' and E'' do not contain the same edges: $E' \neq E''$.*

PROOF. If $E' = E''$ then both sets of edges allow the same paths from s, so $V_i(E') = V_i(E'')$. □

Let $S_i \subset V_i$ be a subset of vertices in layer L_i. Let $E_i \subset E$ be the set of edges between the vertices in layer L_i and layer L_{i+1}. Let $E \subset E_i$ be some subset of these edges. We denote by $Dests(S_i, E)$ the set of vertices in layer L_{i+1} that are connected to some vertex in S_i by an edge in E:

$$Dests(S_i, E) = \{v \in V_{i+1} | \text{there exists some } w \in S_i \text{ and some } e \in E \text{ that } e = (w, v)\}.$$

Let $S_i \subset V_i$ be a subset of vertices in L_i and $E \subset E_i$ be some subset of the edges between layer L_i and layer L_{i+1}. P_{S_i} counts the number of edge subsets in $Pred_{V_i}$ that connect exactly the vertices in S_i. Consider such a subset E' counted in P_{S_i}. $E' \cup E$ is a subset of edges in $PredV_{i+1}$ that connects exactly to $Dest(S_i, E)$. According to Lemma 1, if we iterate over the different S_i's in layer L_i, the P_{S_i}'s count different subsets of edges, and thus every expansion using the edges in E is also different.

Algorithm 1 calculates P_t. It iterates through the layers, and updates the data for the next layer given the data for the current layer. For each layer L_i and every subset of edges in that layer $S_i \subset V_i$, it calculates P_{S_i}. It does so using the values calculated in the previous layer. The algorithm considers every subset of possible vertices in the current layer, and every possible subset of expanding edges to the next layer, and updates the value of the appropriate subset in the next layer.

Algorithm 1

```
 1: procedure CONNECTING-EXACTLY-SUBSETS($G, v$)
 2:     $P_{\{s\}} \leftarrow 1$                                  ▷ Initialization
 3:     for all other subsets of vertices $S$ do          ▷ Initialization
 4:         $P_S \leftarrow 0$
 5:     end for
 6:     for $i \leftarrow 0$ to $n - 1$ do                  ▷ Iterate through layers
 7:         for all vertex subsets $S_i$ in $L_i$ do
 8:             for all edge subsets $E$ between $L_i, L_{i+1}$ do
 9:                 $D \leftarrow Dests(S_i, E)$            ▷ subset in $L_{i+1}$
10:                 $P_D \leftarrow P_D + P_{S_i}$
11:             end for
12:         end for
13:     end for
14: end procedure
```

A c-bounded layer graph contains at most c vertices in each layer, so for each layer there are at most 2^c different subsets of vertices in that layer. There are also at most c^2 edges between 2 consecutive layers, and thus at most $2^{(c^2)}$ edge subsets between two layers.

If the graph contains k layers, the running time of the algorithm is bounded by $k \cdot 2^c \cdot 2^{(c^2)}$. Since c is a constant, this is a polynomial algorithm.

Consider the connectivity game on a layer graph G, with a single source vertex s and target vertex t. The Banzhaf index of the edge e is the number of subsets of edges that allow a path between s and t, but do not allow such a path when e is removed (divided by a constant). We can calculate $P_{\{t\}} = P_{\{t\}}(G)$ for G using the algorithm to count the number of subsets of edges that allow a path from s to t. We can then remove e from G to obtain the graph $G' =< V, E \setminus \{e\} >$, and calculate $P_{\{t\}} = P_{\{t\}}(G')$. The difference $P_{\{t\}}(G) - P_{\{t\}}(G')$ is the number of subsets of edges that contain a path from s to t but no longer contain such a path when e is removed. The Banzhaf index for e is $\frac{P_{\{t\}}(G) - P_{\{t\}}(G')}{2^{|E|-1}}$. Thus, this algorithm allows us to calculate the Banzhaf index on an edge in the connectivity games on bounded layer graphs.

6. RELATED WORK

Measuring the power of individual players in coalitional games has been studied for many years. The most popular indices suggested for such measurement are the Banzhaf index [1] and the Shapley-Shubik index [19].

In his seminal paper, Shapley [18] considered coalitional games and the fair allocation of the utility gained by the grand coalition (the coalition of all agents) to its members. The Shapley-Shubik index [19] is the direct application of the Shapley value to simple coalitional games.

The Banzhaf index emerged directly from the study of voting in decision-making bodies. The *normalized* Banzhaf index measures the proportion of coalitions in which a player is a swinger, out of all *winning* coalitions. This index is similar to the Banzhaf index discussed in Section 1, and is defined as:

$$\widetilde{\beta}_i = \frac{\beta_i(v)}{\sum_{k \in N} \beta_k}.$$

The Banzhaf index was mathematically analyzed in [3], where it was shown that this normalization lacks certain desirable properties, and the more natural Banzhaf index is introduced.

Both the Shapley-Shubik and the Banzhaf indices have been widely studied, and Straffin [20] has shown that each index reflects specific conditions in a voting body. [11] considers these two indices along with several others, and describes the axioms that characterize the different indices.

The naive implementation of an algorithm for calculating the Banzhaf index of an agent i enumerates all coalitions containing i. There are 2^{n-1} such coalitions, so the performance is exponential in the number of agents. [12] contains a survey of algorithms for calculating power indices of weighted majority games. Deng and Papadimitriou [2] show that computing the Shapley value in weighted majority games is #P-complete, using a reduction from KNAPSACK. Since the Shapley value of any simple game has the same value as its Shapley-Shubik index, this shows that calculating the Shapley-Shubik index in weighted majority games is #P-complete.

Matsui and Matsui [13] have shown that calculating both the Banzhaf and Shapley-Shubik indices in weighted voting games is NP-complete.

The problem of computing power indices in simple games depends on the chosen representation of the game. Since the number of possible coalitions is exponential in the number of agents, calculating power indices in time polynomial in the number of agents can only be achieved in specific domains.

In this paper, we have considered the network flow domain, where a coalition of agents must achieve a flow beyond a certain value. The network flow game we have defined is a simple game. [10, 9] have considered a similar network flow domain, where each agent controls an edge of a network flow graph. However, they introduced a non-simple game, where the value a coalition of agents achieves is the maximal total flow. They have shown that certain families of network flow games and similar games have nonempty cores.

7. CONCLUSIONS AND FUTURE DIRECTIONS

We have considered network flow games, where a coalition of agents wins if it manages to send a flow of more than some value k between two vertices. We have assessed the relative power of each agent in this scenario using the Banzhaf index. This power index may be used to decide how to allocate maintenance resources in real-world networks, in order to maximize our ability to maintain a certain flow of information between two sites.

Although the Banzhaf index theoretically allows us to measure the power of the agents in the network flow game, we have shown that the problem of calculating the Banzhaf index in this domain in #P-complete. Despite this discouraging result for the general network flow domain, we have also provided a more encouraging result for a restricted domain. In the case of connectivity games (where it is only required for a coalition to contain a path from the source to the destination) played on bounded layer graphs, it is possible to calculate the Banzhaf index of an agent in polynomial time.

It remains an open problem to find ways to tractably approximate the Banzhaf index in the general network flow domain. It might also be possible to find other useful restricted domains where it is possible to exactly calculate the Banzhaf index. We have only considered the complexity of calculating the Banzhaf index; it remains an open problem to find the complexity of calculating the Shapley-Shubik or other indices in the network flow domain. Finally, we believe that there are many additional interesting domains other than weighted voting games and network flow games, and it would be worthwhile to investigate the complexity of calculating the Banzhaf index or other power indices in such domains.

8. ACKNOWLEDGMENT

This work was partially supported by grant #898/05 from the Israel Science Foundation.

9. REFERENCES

[1] J. F. Banzhaf. Weighted voting doesn't work: a mathematical analysis. *Rutgers Law Review*, 19:317–343, 1965.

[2] X. Deng and C. H. Papadimitriou. On the complexity of cooperative solution concepts. *Math. Oper. Res.*, 19(2):257–266, 1994.

[3] P. Dubey and L. Shapley. Mathematical properties of the Banzhaf power index. *Mathematics of Operations Research*, 4(2):99–131, 1979.

[4] E. Ephrati and J. S. Rosenschein. The Clarke Tax as a consensus mechanism among automated agents. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 173–178, Anaheim, California, July 1991.

[5] E. Ephrati and J. S. Rosenschein. A heuristic technique for multiagent planning. *Annals of Mathematics and Artificial Intelligence*, 20:13–67, Spring 1997.

[6] S. Ghosh, M. Mundhe, K. Hernandez, and S. Sen. Voting for movies: the anatomy of a recommender system. In *Proceedings of the Third Annual Conference on Autonomous Agents*, pages 434–435, 1999.

[7] T. Haynes, S. Sen, N. Arora, and R. Nadella. An automated meeting scheduling system that utilizes user preferences. In *Proceedings of the First International Conference on Autonomous Agents*, pages 308–315, 1997.

[8] E. Hemaspaandra, L. Hemaspaandra, and J. Rothe. Anyone but him: The complexity of precluding an alternative. In *Proceedings of the 20th National Conference on Artificial Intelligence*, Pittsburgh, July 2005.

[9] E. Kalai and E. Zemel. On totally balanced games and games of flow. Discussion Papers 413, Northwestern University, Center for Mathematical Studies in Economics and Management Science, Jan. 1980. available at http://ideas.repec.org/p/nwu/cmsems/413.html.

[10] E. Kalai and E. Zemel. Generalized network problems yielding totally balanced games. *Operations Research*, 30:998–1008, September 1982.

[11] A. Laruelle. On the choice of a power index. Papers 99-10, Valencia — Instituto de Investigaciones Economicas, 1999.

[12] Y. Matsui and T. Matsui. A survey of algorithms for calculating power indices of weighted majority games. *Journal of the Operations Research Society of Japan*, 43, 2000.

[13] Y. Matsui and T. Matsui. NP-completeness for calculating power indices of weighted majority games. *Theoretical Computer Science*, 263(1–2):305–310, 2001.

[14] N. Nisan and A. Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35:166–196, 2001.

[15] A. D. Procaccia and J. S. Rosenschein. Junta distributions and the average-case complexity of manipulating elections. In *The Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 497–504, Hakodate, Japan, May 2006.

[16] J. S. Rosenschein and M. R. Genesereth. Deals among rational agents. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 91–99, Los Angeles, California, August 1985.

[17] T. Sandholm and V. Lesser. Issues in automated negotiation and electronic commerce: Extending the contract net framework. In *Proceedings of the First International Conference on Multiagent Systems (ICMAS-95)*, pages 328–335, San Francisco, 1995.

[18] L. S. Shapley. A value for n-person games. *Contributions to the Theory of Games*, pages 31–40, 1953.

[19] L. S. Shapley and M. Shubik. A method for evaluating the distribution of power in a committee system. *American Political Science Review*, 48:787–792, 1954.

[20] P. Straffin. Homogeneity, independence and power indices. *Public Choice*, 30:107–118, 1977.

[21] M. Tennenholtz and A. Altman. On the axiomatic foundations of ranking systems. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI'05)*, pages 917–922, Edinburgh, August 2005.

[22] M. P. Wellman. The economic approach to artificial intelligence. *ACM Computing Surveys*, 27:360–362, 1995.