Static and dynamic graph algorithms with applications to infrastructure and content management of modern networks

Gerasimos G. Pollatos*

National and Kapodistrian University of Athens
Department of Informatics and Telecommunications
gpol@di.uoa.gr

Abstract. This dissertation focuses on networks primarily used for the dissemination of content. We model these networks using graph theory and study algorithms for the replication of content as well as for their infrastructure. In terms of infrastructure, we propose the first fully dynamic algorithm for maintaing a minimum spanning tree on a directed graph. In terms of content replication and focusing on networks with constant number of clients we propose optimal algorithms that solve the basic problem of replicating data over a network of clients with constrained local storage and extend our results to various important extensions. These algorithms constitute the first research work on networks with non-metric distances among clients. In addition and in order to study implications of the selfish behaviour of users participating in such networks, we define an appropriate non-cooperative strategic game and study existence of pure Nash equilibria as an indication of stabilization of the networks performance. We provide tight bounds for the prices of anarchy and stability, the standard measures of efficiency of such networks. We identify conditions under which equilibria might not be expensive and extend our results to more complex hierarchical networks.

1 Introduction

Typically speaking the term *content network* refers to a large-scale system of computers containing copies of data, placed at various points in a network so as to maximize bandwidth for access to the data from clients throughout the network. A client accesses a copy of the data near to the client, as opposed to all clients accessing the same central server, so as to avoid bottleneck near that server. This *maximization* of the bandwidth dedicated for data access can also be seen as a *minimization* of the access cost each client has to *pay* in order to gain access to desired content.

Content types include ordinary downloadable objects, such as media files, software and documents, as well as web objects and applications. Other components of internet delivery, such as DNS queries, routes and database queries, can

* Dissertation Advisor: Vassilis Zissimopoulos, Professor

also be considered as content. In order to refer to such content, we will use the abstract notion of a data object or simply *object*.

The most fundamental problem a designer of a large-scale content network has to face is the *quality of service*. A system is meaningful if it achieves to serve the purpose it was built for, and succesful if it manages to accomplish that as best as possible. The quality of service relies among others, in two major factors; reliability and performance. Reliability amounts to maintaining the following invariant: "problems arising in the system should have the smallest possible impact, if any, on the offered services". That is, the clients of a system should ideally be unaware of problems related to server availability issues, or sudden power failures in some of the system's database locations. On the other hand, performance is a more complicated issue which involves the basic aims of the network. If speed of access is the aim, then optimization of performance is equivalent to fast access rates, while if up-to-date content availability is the goal, then performance amounts to be able to efficiently update content in all cooperating locations.

In this thesis, we address from an algorithmic perspective both reliability and performance in content networks. We utilize the field of dynamic graph algorithms in order to surpass difficulties in terms of reliability. As for performance, we turn our attention to *cooperative* caching of objects among distributed computers and design algorithms for their coordination by external authorities as well as study implications in their operability due to absence of any external authority.

2 Infrastructure on content networks: the DMST problem

In terms of infrastructure, we study connectivity issues among nodes of the content network. We use graph theory and model the content network as a graph, with each vertex of the graph representing a single user and each edge representing a connection among two users. We assume that each edge is weighted and directed.

In the environment of a content network, the derived graph is obviously subject to discrete changes. Such changes happen if for example a user voluntarily exits the content network, or if a sudden power or hardware failure forces the user to abandon the network. Using graph theoretic terms this is a dynamic graph. We focus on the classic problem of computing a directed minimum spanning tree (DMST) on such a graph. Such a tree is a maximal subset of the edges of the graph, with minimum cost that is (a) acyclic, i.e. does not contain directed cycles, and (b) no vertex of the directed graph has more than one incoming edge in this subset. The need for maintaining such a tree in content networks is apparent due to the fact that the spanning tree essentially maintains a minimum set of links among network nodes (i.e. the minimum requirements for connectivity of all nodes).

The dynamic version of the problem amounts to exploting a previously computed solution so that we can make use of it after a node failure, without having to recompute the new solution from scratch. The problem is studied for the first time here as opposed to the widely studied undirected version. A relevant result proved in this thesis for the hardness of the problem is the following:

Theorem 1. *[1] Dynamic maintenance of a DMST under edge deletions and/or insertions is as hard as recomputing a DMST from scratch if only the DMST information is retained and used between updates.*

The dynamic algorithm for the DMST is based on an appropriate data structure, which the algorithm utilizes for recording redundant edges and all vertices during the execution of the only known algorithm for the static version of the problem i.e. Edmonds' algorithm [2], [3], [4]. The augmented data structure, namely the *Augmented Tree* (ATree), appropriately encodes the redundant edge set H along with all vertices (contracted vertices or c-vertices and simple vertices) processed during execution of Edmonds' algorithm. Simple vertices are represented in the ATree by *simple nodes* while c-vertices are represented by *c-nodes*.

Since a digraph can be always transformed to be strongly connected, all vertices (simple and intermediate c-vertices) will be eventually contracted to a single c-vertex by the end of the algorithm's execution. This c-vertex is represented by the root node of the ATree, which has no parent. The parent of each other node N is the intermediate c-node N^c to which it was contracted. The parent of each node is unique, hence the described structure is indeed a tree.

The ATree has at most $O(n)$ nodes since the algorithm handles at most $O(n)$ contractions. Construction of an ATree can be easily embedded into the functionality of Edmonds' algorithm, without affecting its complexity. Furthermore, all intermediate c-vertices created by Edmonds' algorithm are made explicit in the ATree: For a given c-vertex v_c, we can engage a Breadth-First Search (BFS) starting from its representing c-node in the ATree and collect all encountered simple nodes, hence we gather all vertices of the original digraph that were eventually contracted to v_c. Since the ATree is of $O(n)$ size, BFS takes $O(n)$ time.

For any edge we want to delete from the original digraph, two cases must be considered: (a) the edge does not belong to H, in which case only a simple update on the recorded lists is needed, and (b) the edge belongs in H, in which case we proceed by *decomposing* the ATree, initialize Edmonds' algorithm w.r.t. the remainders of the ATree and execute it.

The purpose of engaging a decomposition of the ATree is to identify surviving c-nodes and hence surviving c-vertices. By this way we can re-execute Edmonds' algorithm on a partially contracted digraph with less vertices, considering less edges than re-evaluating all contractions from scratch.

The decomposition of the ATree begins from node N which is the head of the deleted edge and follows a path from N towards the ATree root, removing all c-nodes on this path except N. Each one of the children of a removed c-node forms its own subtree. By the end of this procedure, the initial structure

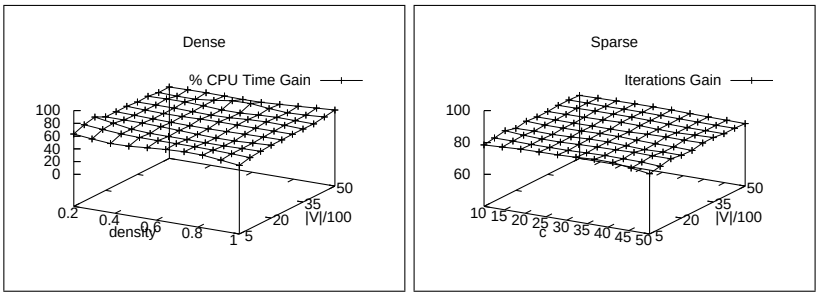

Fig. 1. Experimental evaluation of the dynamic algorithm for the DMST problem.

is decomposed into smaller ATrees, each corresponding to a contracted subset of the original digraph's vertices. All these ATrees remain intact after decomposition. Having performed the decomposition of the ATree, we proceed with a proper initialization and re-execution of Edmonds' algorithm, on a new partially contracted digraph.

We handle edge insertions by reducing them to edge deletions. We first check whether the newly inserted edge should replace some edge encoded in the ATree. This check involves *only* c-nodes of the ATree corresponding to c-vertices containing the head of the new edge. Given that we have found a candidate node N which should replace its current incoming edge with the new one, we proceed by engaging a virtual deletion of the old edge and re-execution of the algorithm on the remaining graph augmented with the newly inserted edge.

To analyze the theoretical performance of our algorithm we used the widely known output complexity model [5]. If we denote by ρ the set of changes in a previously computed solution (the number of affected constituents), made by an algorithm as a response to an update in the dynamic graph, then $\Omega(|\rho|)$ is a lower bound on the complexity of an update, where $|\rho|$ is the cardinality of the set ρ. In the output complexity model, the complexity of updates for a dynamic algorithm is measured as an asymptotic upper bound of a function of $|\rho|$, where the set of affected vertices is ρ, $|\rho|$ being its size and the cardinality of affected edges is denoted with $||\rho||$. Our basic result is:

Theorem 2. *[1] Fully dynamic maintenance of a directed minimum costs spanning tree can be achieved in $O(n + ||\rho|| + |\rho| \log |\rho|)$ output complexity per edge operation.*

In order to measure the practical performance, we employed our algorithm on sequences of edge operations on digraphs of varying order and density and recorded and compared the following two measures: (a) **average `CPU` time**, needed to complete computation of the new DMST and (b) **number of iterations**, performed by each one of the two algorithms. Sample experimental results are depicted in figure 1.

Our dynamic algorithm substantially achieves an update time reduced by a factor of more than 2 as opposed to solving the problem statically (from scratch)

on dense digraphs. However it is our belief that the case of sparse digraphs merits further theoretical investigation from an average case complexity perspective, since as the results indicate, there appears to be an asymptotic improvement on average.

3 Cooperative content replication

One of the most fundamental techniques for improving the performance of a large-scale content network is to cache popular objects close to the clients that potentially request them. This enables requests to be satisfied by a nearby copy and hence reduces not only the access latency but also the burden on the network as well as the remote servers, offering the objects. In the simplest caching scheme, nodes operating as caches never consult one another and when a cache miss occurs, the server is contacted directly. Improvement of the effectiveness of caching is usually accomplished through a powerful paradigm; cooperation. Nodes cooperate both in serving each other's requests as well as in making storage decisions.

Roughly speaking, the problem we study is the following: given a set of cooperating nodes, the amount of local storage at each node, the network distances between the nodes and the predictions of access rates from each node to a set of objects, determine where to place the copies of each object in order to minimize the total access cost over all nodes. We assume that an underlying mechanism exists that can route each node to the closest other node on the network that holds a replica of an object, when an object miss occurs.

From an optimization point of view, this problem is NP-hard since it is a direct generalization of the well known uncapacitated facility location problem where multiple different types of facilities are considered. In this dissertation we focused on the case where the number of cooperating nodes is constant and designed algorithms that efficiently address this problem.

The data placement problem ([6], [7]), is abstracted as follows: there is a network $\mathcal{N}$ consisting of a set $\mathcal{M}$ of $M = |\mathcal{M}|$ users (also referred to as clients) and a universe $\mathcal{O}$ of $N = |\mathcal{O}|$ objects. Each object $o \in \mathcal{O}$ has length l_o and each user $j \in \mathcal{M}$ has a local capacity C_j for the storage of objects. The distance between the users can be represented by a distance matrix D, not necessarily symmetric, where d_{ij} denotes the distance from j to i. The matrix D models the underlying topology. In our work we do not assume any restrictions on the distances, e.g. metric, which is usually the case in the literature.

Each user i requests access to a set of objects $R_i \subseteq \mathcal{O}$, namely its *request set*. For each object o in its request set, client i has a *demand* of access $w_{io} > 0$. This demand can be interpreted as the frequency under which user i requests object o or even as the *preference* that i has for object o. The subset P_i of its request set, that i chooses to replicate locally is referred to as its *placement*. Obviously, $|P_i| \leq C_i$ for unit-sized objects. We assume an installation cost f_i^o for each object o and each cache (user) i.

The objective of the data placement problem is to choose placements of objects for every client such as the total induced access and installation costs for all objects and all clients is minimized. In the following, we will make the reasonable assumption that each object $o \in \mathcal{O}$ is requested by at least one user. If this is not the case, one can always formulate and solve an equivalent problem which has an object set containing only requested objects.

Up to know, the only way to tackle this problem, was the 10-approximation algorithm of [7] designed for the general case. When object lengths are uniform (or equivalently unit) our algorithm finds the optimum solution in polynomial time. When object lengths are non-uniform, our algorithm returns an optimum solution which violates the capacities of the clients' caches by a small, asymptotically tight additive factor.

We show how our results can be modified to handle various important extensions of the problem such as cases when bounds are imposed on the number of maximum replicas allowed for each object (a k-median variant for DP), or cases when upper and lower bounds are imposed on the number of users a single replica of an object can serve. Furthermore, we describe ways to cope with a well known generalization of DP, the page placement problem ([8]), in which bounds are imposed on the number of clients that can connect to a client's cache, as well as cases where object updates are frequent and consistency of all replicas of each object has to be guaranteed.

The latter problem is more commonly known as the connected data placement problem [7]. Our algorithms are applicable with uniform and non-uniform object lengths and can be employed independently of the underlying topology of the network, thus giving the first non-trivial results for non-metric DP problems. Most of the results described in the dissertation appear in [9].

Our basic result for objects of uniform (equivalently unit) length is the following:

Theorem 3. *[9] The non-metric data placement problem with uniform length objects and a fixed number of clients can be solved optimally in polynomial time* $O(N^{M+1})$.

The algorithm we designed is based on dynamic programming algorithm and is in fact pseudo-polynomial, since the complexity depends on the maximum cache size. In the case of unit-sized objects we are able to bound it by the total number of objects and thus obtain a polynomial time algorithm. In the case of objects of arbitrary length the same bound does not hold and the algorithm remains pseudo-polynomial. To tackle this issue, we let $\alpha = \varepsilon l_{max}/N$ where ε is an arbitrarily small positive constant and modify the object lengths and cache sizes appropriately as $l'_o = \left\lfloor \frac{l_o}{\alpha} \right\rfloor$ and $C'_j = \left\lfloor \frac{C_j}{\alpha} \right\rfloor$ respectively. To compute a solution we use the same algorithm with the modified sizes and obtain the following result:

Theorem 4. *[9] The non-metric data placement problem, with non-uniform object lengths and a fixed number of clients, can be solved optimally in polynomial*

	Known results	In this paper
	arbitrary M, metric	**fixed M, no metric**
uniform lengths DP	APX-hard, 10-approx [6, 7, 10]	optimal
uniform lengths with installation costs DP	APX-hard, 10-approx [6, 7, 10]	optimal
connected DP	14-approx [7]	optimal
k-median DP	10-approx [6, 7]	optimal
page placement	13-approx [10] *	optimal with cache augmentation ** εl_{max}
non-uniform lengths DP	10-approx with cache augmentation $l_{\max}$ [7]	optimal with cache augmentation εl_{max}

Table 1. The main known results on the DP problem (*non-uniform lengths with constant blow-up on both capacities, **on cache capacity only).

time using εl_{max} augmentation on the machines' capacity, where ε is an arbitrarily small positive constant and l_{max} is the length of the largest object.

We note that the augmentation in each user's cache stated in the above theorem is asyptotically tight. Furthermore, our results can be modified to handle well-known and important extensions of the DP problem with constant number of clients. The extensions we consider are the following: (1) existense of a distant server (user) whose cache can hold all the objects, (2) there is an upper bound k_o imposed on the number of copies of an object o that are replicated in the network, (3) each user j has a set RJ_j of *rejected* objects that cannot be placed on its cache, (4) the number of users u_{o_i} served by a single copy o_i of object o is lower bounded by $k_{o_i}^{min}$ and upper bounded by $k_{o_i}^{max}$, (5) there is an upper bound k_j on the number of users that can access a given user j's cache, (6) objects updates are frequent and all copies of an object must be up-to-date. We summarize the main results on the DP problem and these extensions, in table 1.

4 Distributed selfish replication

While cooperation of nodes is an attractive and reasonable paradigm in environments where machines trust one another, such as within an Internet service provider, a cache service provider or even a corporate intranet, there are numerous content networks under which this assumption is not valid. For example, file sharing or peer-to-peer networks, that have become extremely popular nowadays, are formed by users that participate voluntarily and have no knowledge of the motives and goals of other participants.

In such networks, one cannot a-priori assume that a participant will voluntarily offer it's local storage for the replication of content that is of no interest to it. On the contrary the most logical assumption is that such users upon joining the network behave selfishly, i.e. aim to maximize their own benefit and thus

the total access cost depends on their selfish replication decisions. The two basic questions that arise under this setting are: (a) does the performance of the network ever stabilize? (b) what is the overall performance of a stable network, in absence of a central optimizing authority?

Using tools from the field of algorithmic game theory, we studied implications of this user behaviour in content networks. We formulated the basic problem of data replication as a strategic non-cooperative game and study this game in order to analyze the performance of these networks. We use the standard quantification measures, the prices of anarchy [11] and stability [12], [13], in order to measure the performance loss due to the lack of coordination.

The modelled strategic game is as follows [14], [15]:

Definition 1. *An instance of the* Distributed Selfish Replication *(*DSR*) game is specified by the tuple* $\langle N, \{P_i\}, \{c_i\}\rangle$*, where:*

- N *is the set of the* n *players, which in our case are the nodes.*
- $\{P_i\}$ *is the set of strategies available to player* i*. Each strategy corresponds to a different placement which means that there is one strategy for each* $size_i$*-cardinality subset.*
- $\{c_i\}$ *is the set of utilities for the players. The utility for each player is the access cost that the player wishes to minimize.*

A placement X is then a *strategy profile* (or *configuration*) for the DSR game. DSR is a n-player, non-cooperative, non-zerosum game [16]. For the DSR game, we investigate configurations that are pure Nash equilibria.

Definition 2. *A pure Nash equilibrium (N.E.) for the* DSR *game is a placement* X^**, such that for every node* $i \in N$*,*

$$c_i(X^*) \geq c_i(\{P_1^*, \ldots, P_{i-1}^*, P_i, P_{i+1}^*, \ldots P_n^*\}$$

for all $P_i \in \{P_i\}$*.*

The definition essentially states that, under such a placement X^*, no node can modify its individual placement unilaterally and benefit from this modification. In what follows we use the terms node and player interchangeably. Let

$$X_{-i} = \{P_1, \ldots, P_{i-1}, P_{i+1}, \ldots, P_n\}$$

refer to the strategy profile of all players but i. For the DSR game, it is easy to see that given a strategy profile X_{-i}, player i can determine optimally in polynomial time its *best response* P_i to the other players' strategies. This computation amounts to solving a special 0/1 *Knapsack* problem, in which player i chooses to replicate the $size_i$ objects with the greatest value $w_{io}d_i(o)$.

The network topologies we consider and the results we obtained are the following. The minimum and maximum distances appearing in the network are also referred to with $d_{\min} = d_k$ and $d_{\max} = d_0$.

Ultrametric Replication Group: this is a network model that generalizes the one introduced by Leff,Wolf and Yu in [17] and studied in [18], involving k origin servers, instead of 1. The distance of every node i from server l is d_l for $l = 0 \ldots k-1$, while two nodes i and j are at distance $d_{ij} = d_k$. In our study we assume that distances form an ultra-metric [1] such that $d_k < d_{k-1} < \cdots < d_0$. We designed a distributed protocol that upon finishind guarantees convergence to pure Nash equilibria.

Theorem 5. *[14] Pure strategy Nash equilibria exist for the* DSR *game on* $\boldsymbol{LWY}(k)$ *networks, and can be found in polynomial time.*

Furthermore we studied the quality of such equilibria obtaining the following results:

Theorem 6. *[14] The price of anarchy for the* DSR *game is upper bounded by* $\frac{d_{max}}{d_{min}}$*. The Price of Stability for the* DSR *game has a lower bound arbitrarily close to* d_{max}/d_{min} *in the worst-case, even for* 1 *server and* 0/1 *demand weights.*

We should note that both results are valid even when the maximum experienced access cost by a user is measured instead of the sum of all access costs. Furthermore we studied networks with modestly demanding participants, that is participants offering storage capacity to the network asyptotically equal to their demand, and identified that in such networks pure equilibria can be of significantly less cost.

Balanced Hierarchies: this is a network model with distances that also form an ultrametric. The network's nodes are clustered hierarchically, so that at each clustering level the maximum distance between any two nodes in the same cluster is given and is smaller than the distance of any two nodes in different clusters. We designed an extension of our basic distributed protocol that also achieves convergence to pure Nash equilibria. Our main result on the quality of the pure equilibria is the following.

Theorem 7. *[15] The Price of Anarchy of the DSR game with* 0/1 *preference weights on balanced hierarchical networks is* $O\left(\frac{\ln n}{\ln \ln n}\right)$*. The Price of Stability of the DSR game on hierarchical networks with* 0/1 *preferences is* $\Omega\left(\frac{\ln n}{\ln \ln n}\right)$*.*

General Networks: in this case the distance matrix $[d_{ij}]$ can be arbitrary. Our study shows that pure equilibria are not guaranteed to always exist when such network topologies are considered.

Theorem 8. *[15] The* DSR *game on general networks is not a potential game.*

[1] An ultrametric is a metric which satisfies the strengthened version of the triangle inequality, $d(x,z) \leq \max\{d(x,y), d(y,z)\}$ for all x, y, z. This essentially states that at least two of the $d(x,z), d(x,y)$ and $d(y,z)$ are the same.

5 Conclusions

This thesis investigated problems arising in content networks and described ways to effectively cope with them. Focusing on infrastructure problems arising in content networks and using tools from the field of dynamic graph algorithms we studied the problem of maintaining a directed minimum spanning tree on a graph that changes dynamically under edge insertions and deletions. After analyzing the hardness of maintaining such a tree, we described the first fully dynamic algorithm that maintains the DMST under edge updates and analyzed it in the output complexity model. The results of the extensive experimental evaluation, revealed the practical efficiency of the proposed algorithm. An important aspect of further research is to resolve the complexity of updates, a small step to which has been made in this thesis.

In terms of content replication we addressed the basic problem of replicating data among users of a content network, the problem most commonly referred to as the data placement problem. We focused on the case of constant number of users and described polynomial time algorithms that solve optimally the basic problem when the objects in consideration have uniform size. When object sizes are not uniform we described an algorithm that also finds the optimum solution to the problem, albeit a small and asymptotically tight augmentation in each user's local storage. The proposed technique was extended to handle various other common extensions of the basic problem such as the page-placement problem and the connected data placement problem among others. A significant characteristic of this technique is its ability to solve these problems independently of the underlying topology of the network thus giving the first non-trivial results for non-metric topologies. A challenging aspect of future work involves employing our results to models involving payments.

Finally, we studied implications on the process of replicating data over a content network when the users are autonomous and selfish and participate voluntarily. We formulated a proper strategic game that modeled the data replication problem and studied conditions under which the network stabilizes. We proved inability to converge to pure Nash equilibria for general underlying topologies. For simple hierarchical networks and balanced hierarchical networks with multiple servers we described an algorithm that reaches pure Nash equilibria and admits a distributed implementation. Furthermore, we analyzed the quality of achieved equilibria by computing the prices of anarchy and stability for the game and identified conditions under which these ratios can be different. The most significant aspect of future work is investigation of whether part or all of our results can be extended in the case of arbitrary-sized objects.

References

1. Gerasimos G. Pollatos, Orestis Telelis, and Vassilis Zissimopoulos. Updating directed minimum cost spanning trees. In *Workshop on Experimental Algorithms (WEA)*, pages 291–302, 2006.

2. F. Bock. An algorithm to construct a minimum spanning tree in a directed network. *Developments in Operations Research*, pages 29–44, 1971.
3. Y.J. Chu and T.H. Liu. On the shortest arborescence of a directed graph. *Scientia Sinica*, 14:1396–1400, 1965.
4. J. Edmonds. Optimum branchings. *Journal of Research of the National Bureau for Standards*, 69(B):125–130, 1967.
5. G. Ramalingam and Thomas W. Reps. On the computational complexity of dynamic graph problems. *Theoretical Computer Science*, 158(1&2):233–277, 1996.
6. I. Baev and R. Rajaraman. Approximation algorithms for data placement in arbitrary networks. In *Proceedings of the ACM-SIAM Annual Symposium on Discrete Algorithms (SODA)*, pages 661–670, 2001.
7. Ivan D. Baev, Rajmohan Rajaraman, and Chaitanya Swamy. Approximation algorithms for data placement problems. *SIAM Journal on Computing*, 38(4):1411–1429, 2008.
8. Adam Meyerson, Kamesh Munagala, and Serge Plotkin. Web caching using access statistics. In *Proceedings of the ACM-SIAM Annual Symposium on Discrete Algorithms (SODA)*, pages 354–363, 2001.
9. Eric Angel, Evripidis Bampis, Gerasimos G. Pollatos, and Vassilis Zissimopoulos. Optimal data placement on networks with constant number of clients. *Submitted to COCOON 2010*, 2010.
10. Sudipto Guha and Kamesh Munagala. Improved algorithms for the data placement problem. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 106–107, 2002.
11. Elias Koutsoupias and Christos H. Papadimitriou. Worst-case Equilibria. In *Proceedings of the Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 404–413, 1999.
12. E. Anshelevich, A. Dasgupta, E. Tardos, and T. Wexler. Near-optimal network design with selfish agents. In *Proceedings of the ACM Annual Symposium on Theory of Computing (STOC)*, pages 511–520, 2003.
13. E. Anshelevich, A. Dasgupta, J. M. Kleinberg, E. Tardos, T. Wexler, and T. Roughgarden. The Price of Stability for Network Design with Fair Cost Allocation. In *Proceedings of IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 295–304, 2004.
14. Gerasimos G. Pollatos, Orestis Telelis, and Vassilis Zissimopoulos. On the social cost of distributed selfish content replication. In *Proceedings of IFIP-TC6 Networking*, pages 195–206, 2008.
15. Gerasimos G. Pollatos, Orestis Telelis, and Vassilis Zissimopoulos. The social cost of distributed selfish content replication. *Submitted to Computer Communications*, 2010.
16. M. J. Osborne and A. Rubinstein. *A course in game theory*. MIT Press, 1994.
17. A. Leff, J. Wolf, and P. S. Yu. Replication Algorithms in a Remote Caching Architecture. *IEEE Transactions on Parallel and Distributed Systems*, 4(11):1185–1204, 1993.
18. N. Laoutaris, O. A. Telelis, V. Zissimopoulos, and I. Stavrakakis. Distributed Selfish Replication. *IEEE Transactions on Parallel and Distributed Systems*, 17(12):1401–1413, 2006.