Online Network Optimization Problems

Bala Kalyanasundaram and Kirk Pruhs

Computer Science Dept., University of Pittsburgh, Pittsburgh, PA 15260, USA.
{kalyan,kirk}@cs.pitt.edu, http://www.cs.pitt.edu/{~kalyan,~kirk}

Abstract. We survey results on online versions of the standard network optimization problems, including the minimum spanning tree problem, the minimum Steiner tree problem, the weighted and unweighted matching problems, and the traveling salesman problem. The goal in these problems is to maintain, with minimal changes, a low cost subgraph of some type in a dynamically changing network.

1 Introduction

In the early 1920's Otakar Borůvka was asked by the Electric Power Company of Western Moravia (EPCWM) to assist in EPCWM's electrification of southern Moravia by solving from a mathematical standpoint the question of how to construct the most economical electric power network [9]. In 1926 Borůvka initiated the study of network optimization problems, by publishing an efficient algorithm for constructing a minimum spanning tree of a fixed network [9]. Certainly since the 1920's the underlying collection of sites that require electrification in southern Moravia has changed frequently as new sites require service, and perhaps occasionally some sites drop service. It would not be reasonable for EPCWM to continually maintain the absolute minimum spanning tree since the addition of a site might radically change the lines in the minimum spanning tree, and there would be significant cost to removing old lines. Hence, the real problem faced by EPCWM would be how to maintain, with minimal changes, a small weight spanning tree of a dynamic network.

In this paper we survey algorithmic results for problems related to maintaining, with minimal changes, a low cost subgraph H of some type in a dynamically changing network G. For example, H may be required to be a maximum matching in an online matching problem. These problems are online in nature because the algorithm is unaware of future changes to G. The particular types of subgraphs we consider are unweighted matchings, weighted matchings, tours, spanning trees, k-connected spanning graphs, Steiner trees, and generalized Steiner trees. Generally these problems are formalized in the following manner. Let N be some fixed global network which may or may not be known a priori, and G_0 some initial subgraph of N. G_{i+1} is formed from either adding a vertex to, or possibly deleting a vertex from, G_i. The goal of the online algorithm is to minimally modify the subgraph H_i of G_i to yield the subgraph H_{i+1} of G_{i+1}.

The first case generally considered for these problems is when G and H are constrained to grow monotonically, i.e. each vertex in G_i is a vertex in G_{i+1} and

$H_i \subseteq H_{i+1}$. The competitive ratio of an algorithm A in this model is then the maximum over all sequences I of changes to G, of the ratio of the cost of the final H subgraph constructed by the online algorithm divided by the cost of the optimal H subgraph.

If vertices may depart from G_i, or edges may be deleted from H_i, then there are two parameters one would like to optimize, the quality of H_i and the cost of restructuring H_i. Notice how both of these costs are captured in the cost of the final H in the monotone model. If one wishes to use competitive analysis, it seems necessary to fix one of these two parameters. Generally you can not hope to be competitive on both parameters at once since this would require that the online algorithm be competitive, in terms of restructuring cost, against an adversary that refuses to restructure its H subgraph, while at the same time, being competitive, in terms of solution quality, against an adversary that always maintains the optimal H subgraph.

In what we call the *fixed quality model*, we assume that we fix a priori a quality parameter β, and that there is a cost function $\alpha(H_i, H_{i+1})$ that gives the cost of changing from H_i to H_{i+1}. Most commonly, if N is unweighted, the cost might be the number of edges in the symmetric difference of H_i and H_{i+1}, and if N is weighted, the cost might be the aggregate weight of the edges in the symmetric difference of H_i and H_{i+1}. The problem is then to minimize the cost of handling the sequence I of changes, while maintaining an H_i that has cost at most β times the cost of optimal solution for G_i. The competitive ratio in this case is then the cost the online algorithm incurs for handling I divided by the optimal cost of maintaining a subgraph within a multiplicative factor of β of optimum. One example where it might be important to keep the quality of H high is the spanning tree example where there is a yearly maintenance cost proportional to the total line length.

Alternatively, in what we call the *fixed cost model*, we fix a cost α that we are willing to spend (in the amortized sense) each time G changes. If N is unweighted, this cost might be the number of edges in the symmetric difference between H_i and H_{i+1}. Here the problem would be to minimize the cost of the final H subgraph subject to the constraint that only $\alpha\ k$ can be spent, where k is the number of changes made to G.

When considering randomized algorithms we consider only oblivious adversaries, i.e. the adversary must specify the sequence I of changes a priori. Generally, adaptive adversaries, which can specify the ith change after seeing the online algorithm's response to the $(i-1)st$ change, rob the the randomized algorithm of the power of randomization.

This paper is organized as follows. In section 2 we consider both weighted and unweighted matching problems. In section 3 we consider subgraphs involving connectivity, such as generalized Steiner trees and spanning trees. In section 4, we consider traveling salesman tours and paths. In each section we relate the main results that have been obtained to date. Where it is possible to do so succinctly, we give a description of the proposed online algorithms, along with the bounds obtained. We assume the reader is familiar with basic graph theoretic concepts, and with basic concepts concerning online algorithms. We generally end the discussion of each paper with the most appealing questions left open in that paper. In the conclusion we briefly discuss directions for future research.

2 Matching

The online transportation (weighted matching) problem is formulated as follows. The initial setting consists of a collection $\{s_1, \ldots, s_m\}$ of server sites which are vertices in a underlying network N. Each server site s_j has a positive integer capacity c_j. The online algorithm sees over time a sequence $\{r_1, \ldots r_n\}$ of requests for service, with each request being a point in N. In response to the request r_i, A must select a site $s_{\sigma(i)}$ to service r_i. The cost/distance incurred to service r_i from s_j would be the weight $d(s_j, r_i)$ on the edge (s_j, r_i) in N. One can assume that each nonedge in N has weight $+\infty$. Each site s_j can service at most c_j requests. The goal is minimize the total cost incurred in servicing the requests. Here the acceptable H subgraphs are maximum matchings in G.

It sometimes helps to have some concrete applications of a problem. One could imagine the server sites as fire stations, the servers as fire trucks, the requests as fires, the cost as the distance/time required to get from a fire station to a fire, and the problem is to minimize the average response time to a fire. Or one could imagine the server sites as schools, the servers as seats in the schools, the requests as the arrival of a student in a school district, the cost as the distance from the student's home to the school, and the problem is to minimize the average distance a student has to travel to school.

Online matching is quite similar to the k-server problem. The difference is that the servers in the k-server problems are mobile, while in online matching they have fixed sites. See the survey on server problems for more information. Also many online load balancing problems can be phrased as online matching problems. See the survey on online load balancing for more information.

2.1 Unweighted Matching

In unweighted matching problems it is assumed that all edges have weight 1 or $+\infty$. Under this assumption it is easier to think about the $+\infty$ edges as not being present, and the problem is constructing a maximum cardinality matching. In this setting the competitive ratio is the number of request vertices matched by the online algorithm divided by the size of the maximum matching. Notice that the competitive ratio is at most 1, and to be consistent with other papers we follow this notation instead of considering the inverse of this ratio. Karp, Vazirani, and Vazirani [19] consider the standard matching problem, with each server capacity c_j is equal to 1, in the monotonic model. Since every maximal matching is at least half of the size of a maximum matching, every deterministic algorithm that matches a request, when it is possible to do so, is $\frac{1}{2}$-competitive. It is easy to see that no deterministic algorithm can be have a competitive ratio better than $\frac{1}{2}$. [19] gives a randomized algorithm RANKING whose competitive ratio approaches $1 - 1/e$ for large n. Initially, RANKING numbers the sites randomly so that each of the $n!$ orderings are equally likely. Then in response to a request, RANKING selects the highest ranked unused site to service that request. [19] also show that RANKING is optimally competitive, up to lower order terms, for randomized algorithms. Kao and Tate [18] give some partial results for the case that points arrive in groups of t at a time.

Kalyanasundaram and Pruhs [17] continue the study in the monotonic model for the case where each capacity c_j is equal to some fixed integer k. [17] give a deterministic algorithm BALANCE whose competitive ratio is $1 - \frac{1}{(1+\frac{1}{k})^k}$, which approaches $1-1/e$ for large n and large k. Among all server sites that can service a request r_j, BALANCE selects an arbitrary site that has a maximal number of unused servers at that site. The competitive ratio of BALANCE is shown to be optimal (including low order terms) among deterministic algorithms.

Open Question: What is the optimal randomized algorithm and the optimal randomized competitive ratio for the case that each $c_j = k$. One obvious candidate algorithm, BALANCE-RANKING, would initially rank the server sites randomly, and then in response to a request r_j select the highest ranked server site among the set of sites with the maximal number of unused servers among the sites that can service r_j.

Motivated by problems in mobile computing, Grove, Kao, Krishnan, and Vitter [10] consider the unweighted matching problem in essentially the fixed quality model, with the quality parameter $\beta = 1$. That is they wish to always maintain a maximum matching. Strictly speaking, [10] assume that a request r_j must be serviced when it arrives, if it is possible to do so without terminating service to another request, but there is no requirement that r_j receive service at some later point of time if it should become possible to provide service. However, this seems to be equivalent to the fixed quality model in the sense that bounds for one model hold for the other. The setting for mobile computation is many roving mobile computers (the requests) that each must stay in contact with a mobile support station, or MSS (the server sites). Each MSS has a limit on the number of mobile computers it can support, and a maximum range of service (so not every MSS can support every mobile computer). The event of a mobile computer leaving the coverage area of its current MSS can be simulated by a request departure, followed by a request arrival. Motivated by this mobile computing application, [10] make the natural assumption that the cost of switching the server site handling a request is approximately the same as the cost of handling a request. So generally speaking, servicing a request requires a sequences of switches along an augmenting path, with the cost being the length of the augmenting path.

All of the results presented in [10] assume that the degree of each request vertex is only two. They argue that this should be the worst case because more paths intuitively help the online algorithm more than the adversary. When each $c_j = 1$ and no vertices leave G, the greedy algorithm (that assigns the request to an arbitrary server site that will minimize the number of switches) is $\Theta(\log n)$ competitive and this is optimal among deterministic algorithms. However, in the case that vertices can leave G, the competitive ratio of greedy is $\Omega(n)$. Define RAND to be the algorithm that switches along an augmenting path with probability inversely proportional to the length of the path. So RAND is essentially a randomized harmonic form of the greedy algorithm. In the case that each $c_j = 1$ and vertices are allowed to leave G, the competitive ratio of RAND is $O(\sqrt{n})$ and is $\Omega(n^{1/3})$. [10] show that the competitive ratio of every deterministic algorithm is $\Omega(\sqrt{n})$, and implicitly show that the competitive ratio of every randomized algorithm is $\Omega(\log n)$ for the general problem in the fixed quality model with $\beta = 1$. Optimal algorithms for some specific metric spaces, and lower bounds for some other natural algorithms can also be found in [10].

Open Question: The obvious open question is to find optimally competitive deterministic and randomized algorithms for the general problem in the fixed cost model. Perhaps allowing β to be slightly larger than 1 might make the online algorithm's task easier. Note that the problem becomes almost trivial for $\beta = 2$.

2.2 Weighted Matching

Kalyanasundaram and Pruhs [14] and Khuller, Mitchell, and Vazirani [20] considered the online assignment problem, i.e. the online transportation problem where each capacity $c_j = 1$ (so $n \leq m$). If the costs are arbitrary then [14, 20] show that no algorithm (deterministic or randomized) can have a competitive ratio independent of the edge costs. So [14, 20] assume the costs are indeed distances, and satisfy the triangle inequality. [14, 20] give a deterministic online algorithm, called PERMUTATION in [14], with competitive ratio $2m - 1$. Let M_i be the minimum weight matching of the first i requests. In response to request r_i, PERMUTATION answers r_i with the unique unused server site in the same connected component as r_i in $M_{i-1} \cup M_i$. [14, 20] show that $2m - 1$ is the lower bound on the competitive ratio of any deterministic algorithm in a "star" metric space with the servers initially on the leaves. A star metric space has one center point of distance 1 from the leaves and each leaf is distance 2 from every other leaf.

The natural greedy algorithm that matches a request with the closest unused server has a competitive ratio of $2^m - 1$ [14]. Assuming that the points are uniformly scattered on the unit disk in the Euclidean plane, Tsai, Tang and Chen [22] showed that the expected cost for greedy is at most $O(\sqrt{m})$ times the cost of the minimum matching. Other variants of online assignment have also been considered. For the online bottleneck matching problem, where you want to minimize the maximum weight of an edge in the matching, PERMUTATION is still $2m - 1$ competitive and Idury and Schäffer [11] showed a lower bound on the competitive ratio of any deterministic algorithm that approaches $m/\ln 2 \approx 1.44\, m$ for large m. For the problem of maximizing the weight of the matching, [14] showed that the greedy algorithm, which matches each server to the farthest available server, is 3-competitive and that this is optimal for deterministic algorithms.

Kalyanasundaram and Pruhs [16] consider the online transportation problem under the assumption that the capacity of each site c_j is equal to some integer

k. The competitive ratio for greedy is still $2^m - 1$, but the competitive ratio for PERMUTATION rises to $\Theta(km) = \Theta(n)$. Notice that greedy can be asymptotically better than PERMUTATION if $2^m = o(n)$.

Open Question: What is the optimal competitive ratio in terms of m, the number of sites, for the online transportation problem? It seems that there should be an $2m - 1$ competitive algorithm.

In [16] the weak adversary model is assumed. In this model, the online algorithm A that has c_i servers at s_i is compared to an offline line adversary that has only $a_i < c_i$ servers at s_i. The weak competitive ratio of an online algorithm A is the supremum over all instances I, with $n \leq \sum_{i=1}^{m} a_i$ requests, of the cost incurred when the assignment is made by A on I divided by the minimum possible cost of any assignment for the instance I given only a_i servers at s_i. The weak adversary model is motivated by the desire to exclude adversarial inputs, and the desire to determine the additional resources required by the online algorithm to counteract the adverse effect of nonclairvoyance. Further motivation can be found in [16]. In [16] only the case $a_i = c_i/2$ is considered. [16] show that the weak-competitive ratio of the greedy algorithm is $\Theta(\min(m, \log C))$, where $C = \sum_{i=1}^{m} c_i$ is the sum of the capacities. If the server capacity of each site is constant, then the weak competitive ratio is logarithmic in m, a significant improvement over the exponential traditional competitive ratio for greedy. Furthermore, [16] give an online algorithm, BALANCE, that is a simple modification of the greedy algorithm, and that has a constant weak competitive ratio. Define the pseudo-distance $pd(s_i, r_j)$ to handle a request r_j by s_i to be $d(s_i, r_j)$ if less than half the servers at s_i have been used, and $\gamma \cdot d(s_i, r_j)$ otherwise. Here γ is some sufficiently large constant. BALANCE then services each request r_j with the available server site s_i that minimizes $pd(s_i, r_j)$.

Open Question: What is the weak competitive ratio for a general a_i? We know of no nonconstant lower bound on the weak competitive ratio even in the case that each $a_i = c_i - 1$.

Open Question: The optimal competitive ratio for the online transportation problem depends critically on the underlying metric space N. It would be interesting to determine the optimal competitive algorithms for common metric spaces such as the line and the Euclidean plane. The online matching problem on a line can also be formulated in the following way. A ski rental shop must match skis to skiers as the skiers walk through the door, with each skier ideally getting skis of height equal to their height. The problem is to minimize the average discrepancy between skier's heights and the heights of their assigned skis.

The underlying combinatorics for the online assignment problem seems to be captured by an online search game, that we call hide and seek. To begin, the hider and the seeker are at some origin point. The hider, unbeknownst to the seeker, then moves to some new point x in the metric space N. The seeker then traverses N until it arrives at x. The seeker's goal is minimize the ratio of the distance that it travels divided by the distance between the origin and x. This corresponds to a lazy adversary that only moves one server. There is an $O(\sqrt{n})$ competitive algorithm for hide and seek in the plane [13] and a constant competitive algorithm for the line. We believe these are also the optimal competitive ratios for online assignment in these metric spaces. One possible candidate algorithm for the line is the generalized work function algorithm. The generalized work function algorithm picks a collection S_i of servers, to handle the collection R_i of the first i requests, that minimizes $\gamma\ MM(S_i, R_i) + MM(S_i, S_{i-1} \cup \{r_i\})$, where $MM(X, Y)$ denotes the minimum matching between X and Y, and $\gamma > 1$. This algorithm is a linear combination of PERMUTATION (the first term) and greedy (the second term). As circumstantial evidence that the generalized work function algorithm is a reasonable candidate for the line, we note that it is essentially optimal against a lazy adversary.

Ultimately, it would be nice to have a "universal" metric space dependent algorithm for online weighted matching. A universal algorithm would have a competitive ratio that is at most a constant times optimal for every metric space. Since there seems to be no natural description for the optimal online algorithm for online assignment with 3 server sites on a line [14], it seems unlikely that there is a simple algorithm that would assure absolute optimality on every metric space. Note that it is unlikely that the generalized work function algorithm will be universal since it uses nearest neighbor search to search configurations that have the same optimal cost. Nearest neighbor is known to be only $\Theta(\log n)$ competitive for online TSP [21]. Although, a proof that the generalized work function algorithm is within $O(\log n)$ of being universal would be a good first step toward finding a universal algorithm.

Open Question: What is the optimal randomized competitive algorithm for the online assignment problem (and online transportation problem)? Suggestive evidence that an $O(\log n)$-competitive randomized algorithm might exist can be found from the fact that the optimal competitive ratio for the star metric space (the worst case for the deterministic algorithm) is $2H_k - 1$ against an oblivious adversary.

3 Generalized Steiner Trees

Consider the design of a reliable communication/transportation network of low cost. Usually, reliability means that we would like to maintain connections between given pairs of sites even if some number of sites/links are down. Krarup (see [23]) introduced the generalized Steiner problem GSP to study many variants of such design problems.

Given an undirected graph G with positive weights on the edges, the goal in GSP is to construct a minimum weight subgraph of G that satisfies a given connectivity requirement. More precisely, in the node-GSP (edge-GSP) problem the connectivity requirement is specified by assigning a nonnegative integer $r(u, v)$ for every pair of distinct nodes u and v in G, and the goal is to construct a minimum weight subgraph of G with at least $r(u, v)$ node-disjoint (edge-disjoint) paths between u and v.

In the online version of GSP, the online algorithm sees a sequence of requests of the form $(u_i, v_i, r(u_i, v_i))$ where u_i, v_i are vertices in the underlying network

N, which is assumed to be known a priori. The acceptable H subgraphs are those where there are at least $r(u_j, v_j)$ disjoint paths between u_j and v_j for all $j \leq i$. Note that in general the subgraph H need not be a single connected component. With the exception of the paper of Imase and Waxman [12], all the papers assume the monotonic model. We use n to denote the number of nodes in the underlying network N, s to denote the number of requests, and k to denote the number of distinct vertices that appeared in at least one of the s requests.

Westbrook and Yan [23] showed that there is no randomized algorithm that achieves a competitive ratio better than $(k+2)/4$ for online node-GSP, and that there is no randomized algorithm that achieves a competitive ratio better than $(s+1)/4$ for online edge-GSP, even if $r(u,v) = 2$. On the other hand, the greedy algorithm, which adds the minimum cost collection of edges that will maintain the connectivity requirement, is trivially $O(s)$ competitive. Westbrook and Yan showed that the greedy algorithm is $O(k)$-competitive for node-GSP if each $r(u,v) = 2$. For edge-GSP, the greedy algorithm is $O(k)$-competitive if the connectivity requirements form a nonincreasing sequence.

Open Question: Can the competitive ratios be improved if the underlying graph is a metric space, i.e. it is complete graph and the edge weights satisfy the triangle inequality?

In the case that each $r(u,v) = 1$, Westbrook and Yan [24] showed that the greedy algorithm has a competitive ratio of $O(\sqrt{k}\log k)$. In contrast, the competitive ratio of every deterministic online algorithm is $\Omega(\log k)$ [12, 24]. Awerbuch, Azar and Bartal [4] proved that the greedy algorithm is $O(\log^2 k)$ competitive. Subsequently [7] gave a $O(\log k)$ competitive online algorithm.

Open Question: What is the competitive ratio of the greedy algorithm?

Imase and Waxman [12] considered the dynamic Steiner tree problem, which is a restricted case of GSP, where nodes in G are revealed online and the goal is to maintain a Steiner tree that contains all of the revealed nodes. They considered both the monotonic model and the fixed cost model. They showed a lower bound of $\Omega(\log k)$ on the deterministic competitive factor for the monotonic case. They showed that the greedy algorithm is $O(\log k)$-competitive.

Chandra and Viswanathan [8] extended the results of Imase and Waxman by considering the problem of constructing a d-connected subgraph of the revealed nodes in the monotone model. They showed that the greedy algorithm is $8d^2 \log k$ competitive. They gave a lower bound of $(d/4)\log(k/d-1)$ on the competitive ratio of every deterministic algorithm. They showed that any online algorithm A, with competitive factor c, can be modified to yield another online algorithm B with competitive factor at most $2c$, and B maintains an H subgraph with maximum degree at most $3d$. Notice that a minimum degree bound of d is required to maintain a d-connected graph. Unfortunately, the diameter of this subgraph is not small. They showed that the degree bound of $2d$ is necessary for any deterministic online algorithm to maintain a d-connected subgraph. Analogous to reducing the degree bound, they also showed how to augment the selection of edges by the greedy algorithm so that the diameter is at most 3 times that of the offline optimal diameter. It appears that the diameter is reduced in this construction at the cost of increasing the degrees of the nodes.

Open Question: Is there an online algorithm which is close to optimal in size, diameter, and degree?

Alon and Azar [1] considered the online Steiner tree problem where the underlying graph is the Euclidean plane. Based on a construction by Bentley and Saxe [6], they proved a lower bound of $\Omega(\log k / \log\log k)$ on the competitive ratio of any randomized online algorithm.

The one result that does not assume the monotone model is by Imase and Waxman [12], who considered the fixed cost model. If k points have been added [12] allow $O(k^{3/2})$ edge deletions/additions to the H subgraphs, or in other words, $O(k^{1/2})$ amortized changes per revealed vertex. They showed that this many changes is sufficient to maintain a Steiner tree that has cost at most a constant times optimum.

Open Question: Is it possible to reduce the number of edge deletions to constant per change to G?

Finally we mention some results that are closely related to GSP. Bartal, Fiat and Rabani [5] used competitive analysis to measure the efficiency of algorithms for the file allocation problem which is closely related to GSP. Awerbuch, Azar and Bartal [4] generalized GSP to the *Network Connectivity Leasing Problem*, where the online algorithm can either buy or lease an edge while constructing the graph. Obviously the cost of buying is more than that of leasing. They gave $O(\log^2 n)$-competitive randomized algorithm for this problem.

4 Traveling Salesman Problems

A traveling salesman path (TS path) is a walk that visits each vertex in G, and a tour is a TS path that returns to the point of origin. The online versions of TSP in the literature involve a salesman physically moving through G attempting to visit all the vertices of G. It might help to imagine the edges as streets and the vertices as intersections. Since these problems fall more within the paradigm of search problems (see the survey on search problems) they do not fit neatly within in the models proposed in the introduction. Ausiello, Feuerstein, Leonardi, Stougie, and Talamo [2, 3] consider a variant of the TSP problem with release times. Let t_i be the time that the ith point p_i is added to G. It is required that the salesman must visit p_i after time t_i. One can imagine the salesman as occasionally receiving additional assignments via a cellular phone. When a new point arrives the salesman can, at no cost, recompute a new path to visit the remaining points. The competitive ratio is then just the length of the path traveled by the online traveling salesman divided by the optimal path that satisfies the release time constraints.

[2, 3] give the following results. The graph G is always assumed to be a metric space. In [2] they first considered the TS path problem, i.e. the salesman is not constrained to return to the origin. For an arbitrary metric space, the greedy algorithm has a competitive ratio of $5/2$. If a new point p arrives, greedy recomputes, as its future planned route, the shortest route T that visits the

remaining unvisited points. Note that greedy requires superpolynomial time per query unless $P = NP$. By using the minimum spanning tree approximation for T one can obtain a 3-competitive algorithm that requires only polynomial time computation. The competitive ratio for every deterministic algorithm is at least 2, even if the underlying metric space N is the real line. If N is the real line, the algorithm that always attempts to visit the extreme point that is nearest the origin is 7/3 competitive.

In the tour problem, where the salesman is required to return to the origin, [3] shows that life is slightly easier for the online algorithm. For an arbitrary metric space, a modification to the greedy algorithm is 2-competitive. If the salesman is at the origin, it computes a the optimal tour T that visits the remaining unvisited points in G and then returns to the origin, and then the salesman begins traversing T. If a new point x arrives in G that is farther from the origin than the current position of the salesman, then the salesman returns on the shortest path to the origin, ignoring all point arrivals on the way. New points p that are closer to the origin than the salesman's position when p arrives are ignored until the salesman returns to the origin. There is an algorithm CHR that is 3-competitive and requires only polynomial computation time. If a new point arrives, CHR uses Christofides approximation algorithm to find a path T through the remaining unvisited points and returns to the origin. The competitive ratio for any deterministic algorithm on a general metric space is at least 2, and on a line is at least $(9+\sqrt{17})/8 \approx 1.64$. There is a rather complicated algorithm PQR that is shown to have a competitive ratio of 7/4 on a line.

Motivated by the problem of a salesman attempting to visit all the towns in some rural state for which he does not have a map, Kalyanasundaram and Pruhs [15] define the following version of online TSP. When the salesman visits a vertex v, it learns of each vertex w adjacent to v in N, as well as the length $|vw|$ of the edge vw. The salesman's goal is to visit all of the vertices in N, with his/her path being as short as possible. Note that the salesman controls to some extent how N is revealed. [15] gives a deterministic algorithm ShortCut that is at most 16-competitive when N is planar. Note that N need not even be a metric space, all that is required is that the unweighted version of N can be drawn in the plane so that no edges cross.

ShortCut is a modification of depth first search that occasionally shifts the search from one portion of G to another via a known path in G. At any particular time, let $d(v, w)$ denote the length of the shortest path known between vertices v and w using only the known part of N. An edge xy is called a boundary edge if x is explored but y is not. A boundary edge xy blocks a boundary edge vw if $|xy| < |vw|$ and $d(v, x) + |xy| < (1 + \delta)|vw|$. A boundary edge vw is a shortcut if no other boundary edge blocks vw. In ShortCut the searcher begins as if it were performing a standard depth first search on G. Assume that the searcher is at a vertex v and is considering whether to traverse a boundary edge vw. The edge vw will be traversed at this time if and only if vw is a shortcut. Assume that the searcher just traversed a boundary edge xy, causing y to become visited. It may then be the case that some other boundary edge vw, whose traversal was delayed at some previous point in time, now becomes a shortcut. In this case, a so-called jump edge is added from y to w. If at some time, ShortCut directs the searcher to traverse this jump edge, then the searcher will actually traverse the shortest path that it is aware of from y to w.

Open Question: We do not know of any example showing that ShortCut is not constant competitive for an arbitrary N. Note that in [15] the planarity of N is only required in the proof that ShortCut is constant competitive. The obvious question is to determine if ShortCut, or some other online algorithm, is constant competitive for an arbitrary N.

5 Conclusion

There has been a reasonable amount of work on online network optimization in the monotone model. Usually one first analyzes the greedy algorithm that lets H_{i+1} be the cheapest extension of H_i. Some reasonable fraction of the time the greedy algorithm turns out to be close to optimal. Even if it isn't, the analysis usually provides good intuition for developing a better algorithm. Often this better algorithm is relatively a minor modification to greedy.

There has been much less work in the fixed quality and fixed cost models. This is probably the area where there is the greatest need for further investigation. The obvious greedy algorithm in the fixed quality model is have H_{i+1} be the cheapest modification of H_i that satisfies the quality constant. Analogously, the obvious greedy algorithm in the fixed cost model is to have H_{i+1} be the solution closest to optimal that can be obtained from H_i within the cost bound. As a first step, one should probably try to analyze these greedy algorithms. The scant evidence available to date [10, 12] suggests that perhaps the greedy algorithms in the fixed cost and fixed quality models will be less likely to be close to optimally competitive than in the monotone model.

Acknowledgments: We thank Eddie Grove and Stefano Leonardi for helpful suggestions.

References

1. N. Alon and Y. Azar. On-line steiner trees in the euclidean plane. *Discrete and Computational Geometry*, 10:113–121, 1993. Also in *Proc. 8th ACM Symposium on Computational Geometry*, 1992, pp. 337-343.
2. G. Ausiello, E. Feuerstein, S. Leonardi, L. Stougie, and M. Talamo. Serving requests with online routing. In *Proceedngs of Scandinavian Workshop on Algorithm Theory*, volume 824 of *Lecture Notes in Computer Science*, 1994.
3. G. Ausiello, E. Feuerstein, S. Leonardi, L. Stougie, and M. Talamo. Competitive algorithms for the on-line traveling salesman problem. In *Proceedngs of International Workshop on Algorithms and Data Structures*, volume 955 of *Lecture Notes in Computer Science*, 1995.
4. B. Awerbuch, Y. Azar, and Y. Bartal. On-line generalized steiner problem. In *Proc. of 7th ACM-SIAM Symposium on Discrete Algorithms*, pages 68–74, 1996.

5. Y. Bartal, A. Fiat, and Y. Rabani. Competitive algorithms for distributed data management. In *Proc. of the 24th Symposium on Theory of Computation*, pages 39–48, 1992.
6. J. Bentley and J. Saxe. An analysis of two heuristics for the euclidean traveling salesman problem. In *Proceedings of the Allerton Conference on Communication, Control, and Computing*, pages 41–49, 1980.
7. P. Berman and C. Coulston. Online algorithms for steiner tree problems. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, 1997.
8. B. Chandra and S. Viswanathan. Constructing reliable communication networks of small weight. *Journal of Algorithms*, 18(1):159–175, 1995.
9. R. Graham and P. Hell. On the history of the minimum spanning tree problem. *Annals of the History of Computing*, 7:43–57, 1985.
10. E. Grove, M. Kao, P. Krishnan, and J. Vitter. Online perfect matching and mobile computing. In *Proceedings of International Workshop on Algorithms and Data Structures*, 1995.
11. R. Idury and A. Schaffer. A better lower bound for on-line bottleneck matching. manuscript, 1992.
12. M. Imase and B. M. Waxman. Dynamic steiner tree problem. *SIAM J. Discrete Math.*, 4:369–384, 1991.
13. B. Kalyanasundaram and K. Pruhs. A competitive analysis of algorithms for searching unknown scenes. *Computational Geometry: Theory and Applications*, 3:139–155, 1993. Preliminary version appeared in STACS '92.
14. B. Kalyanasundaram and K. Pruhs. Online weighted matching. *Journal of Algorithms*, 14:478–488, 1993. Preliminary version appeared in SODA '91.
15. B. Kalyanasundaram and K. Pruhs. Constructing competitive tours from local information. *Theoretical Computer Science*, 130:125–138, 1994. Preliminary version appeared in ICALP '93.
16. B. Kalyanasundaram and K. Pruhs. The online transportation problem. In *Proceedings of European Symposium on Algorithms*, volume 979 of *Lecture Notes in Computer Science*, pages 484–493, 1995.
17. B. Kalyanasundaram and K. Pruhs. An optimal determinstic algorithm for online b-matching. manuscript, April 1996.
18. M.-Y. Kao and S. R. Tate. Online matching with blocked input. *Inform. Process. Lett.*, 38:113–116, 1991.
19. R. M. Karp, U. V. Vazirani, and V. V. Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 352–358, Baltimore, Maryland, May 1990.
20. S. Khuller, S. G. Mitchell, and V. V. Vazirani. On-line algorithms for weighted bipartite matching and stable marriages. In *Proc. 18th Int. Colloquium on Automata, Languages and Programming*, pages 728–738. Lecture Notes in Computer Science 510, Springer-Verlag, Berlin, 1991.
21. D. Rosenkrantz, R. Stearns, and P. Lewis. An analysis of several heuristics for the traveling salesman problem. *SIAM Journal of Computing*, 6:563–581, 1977.
22. Y. Tsai, C. Tang, and Y. Chen. Average performance of a greedy algorithm for the on-line minimum matching problem on euclidean space. *Information Processing Letters*, 51:275–282, 1994.
23. J. Westbrook and D. Yan. Linear bounds for on-line steiner problems. *Information Processing Letters*, 55:59–63, 1995.

24. Jeffery Westbrook and Dicky C.K. Yan. Lazy and greedy: On-line algorithms for steiner problems. In *Proc. 1993 Workshop on Algorithms and Data Structures*, 1993. To appear in *Mathematical Systems Theory*.