1 Introduction

This chapter surveys results on on-line 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. 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 chapter 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 on-line matching problem. These problems are on-line 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 on-line 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 on-line 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 on-line 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 $\beta$, 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 $\beta$ times the cost of optimal solution for $G_i$. The competitive ratio in this case is then the cost the on-line algorithm incurs for handling $I$ divided by the optimal cost of maintaining a subgraph within a multiplicative factor of $\beta$ 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 $\alpha$ 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 $G$. 