Vertex-Ant-Walk – A robust method for efficient exploration of faulty graphs

Vladimir Yanovski a, Israel A. Wagner a,b and Alfred M. Bruckstein a

a Computer Science Department, Technion IIT, Haifa 32000, Israel
b IBM Haifa Research Laboratory, MATAM, Haifa 31905, Israel

We consider a problem of decentralized exploration of a faulty network by several simple, memoryless agents. The model we adopt for a network is a directed graph. We design an asynchronous algorithm that can cope with failures of network edges and nodes. The algorithm is self-stabilizing in the sense that it can be started with arbitrary initializations and scalable – new agents can be added while other agents are already running.

Keywords: dynamic graph search, edge failure model, vertex ant walk, edge ant walk, cover time, self stabilization

1. Introduction

1.1. Background

The inspiration for the algorithms discussed in this chapter is the behavior of ants. Studies on ants (Adler et al. [1], Gordon [4]) suggest that the strategies that ants use in foraging for food in unknown terrains tend to be very efficient. It is believed that ants build a network of information with nodes represented by points of encounter between ants and the information is either passed between ants at a node or via pheromone traces that are left on the ground. In our, clearly over-simplified, model we assume that the information network is a graph and the role of a pheromone is taken by labels at each node, labels that our search agents or “ants” can read and modify.

In particular, we consider a problem of continuously patrolling a network by a group of simple, identical, memoryless agents. The network is represented by a strongly connected graph \( G(V, E) \). Each ant can write a label at the node at which it is positioned and is able to read the labels of the neighboring nodes directly connected with its present location.

Graph exploration has been a subject of interest since ancient times. The ancient Greek myth about Theseus finding his way out of a labyrinth with help of Ariadne’s thread is, probably, the first description of systematic backtracking in graph exploration.

In his paper Tarry [9] gave a linear algorithm for graph exploration which was extended by Tarjan [8] to the algorithm that is now commonly called Depth First Search (DFS). In Tarry’s algorithm a DFS agent labels the nodes that were already visited and, when it enters such a node again, it backtracks from the node using a concept of an
Ariadne-style “thread” it deployed. While exploring, the thread is unfolded; it is folded during the backtracking. In this way the algorithm is guaranteed to visit all vertices of the graph in \(2|E|\) time and return to the origin. Now consider a situation when there exists some faulty edge in the graph; this edge was visible during the exploration but, when an agent tries to backtrack through it, the edge fails. Hence, during the backtracking process “Ariadne’s thread” is cut, and the DFS agent is lost. In addition to not being a robust algorithm, DFS is not really suited for a distributed scenario with several agents required to cooperate indirectly through the labels they leave at the graph’s vertices.

The algorithm we present in this chapter, called Vertex-Ant-Walk (VAW), traverses all vertices of the graph \(G(V, E)\) in \(|V|\text{Diam}(G)\) time. Furthermore, we can provide the agent with a halting condition so that it knows when the graph is covered. While it is possible that by the time all vertices are visited once, some vertices were visited up to \(\text{Diam}(G)\) times, the agent learns the topology of the graph and, as we shall prove later, by time \(|V|(\text{Diam}(G) + 2m)\) each node is visited at least \(m\) times. We show that VAW is a self-stabilizing algorithm: if the network changes as result of edge failures or adding new nodes or edges, the information network built by the agent gradually adapts to the modified graph. We also prove that if our network has a stable backbone – a strongly connected sub-graph \(G'(V, E')\) with failure-free edges – the agent is guaranteed to visit all vertices of the graph in time \(|V|\text{Diam}(G')\). It is not difficult to see that in the absence of a stable backbone, an adaptive adversary can cause any algorithm to fail to cover a graph by dropping or adding edges while preserving the strong connectivity of the graph at all times.

We also design a memory efficient version of VAW that uses \(O(1)\) memory for storing the labels per node. This memory reduction is, however, achieved at the cost of robustness to failures and self-stabilization.

One important property of VAW is its extensibility: new agents can be added on the fly and they will start to cooperate with the already working agents through the labels left in the vertices. Nevertheless, there are extreme cases when there is no speed-up from adding new agents; for instance, if the graph is a “caterpillar” (as in figure 2), experiments show that when the number of agents is small relative to the size of the graph, the practical speed-up is almost linear.

1.2. Related work

The VAW algorithm, as a parallel multi-agent algorithm for graph exploration, was defined by Wagner et al. [11] and has as its precursor the \(LRTA^*\) algorithm of Korf [7]. It was proved by Wagner et al. [11] that in the case of a simple undirected graph \(G(V, E)\) all vertices of a graph are visited in time not exceeding \(|V|\text{diam}(G)\). Koenig et al. [6] show that in a strongly-connected directed graph the cover time by \(LRTA^*\) is bounded from above by \(2|V|\text{diam}(G)\) for any number of agents.

Wagner et al. [12] discuss the behavior of an algorithm similar to that discussed in this chapter, in the case of a network represented by an undirected graph, for different kinds of network failures.