The Minimum Road Trips Problem

Samuel Micka and Brendan Mumey
Gianforte School of Computing, Montana State University
Bozeman, Montana, USA
samuel.micka@msu.montana.edu and brendan.mumey@montana.edu

Abstract—Road networks can be represented as partially directed graphs; directed edges are one-way road segments and undirected edges can be traversed in either direction. Vehicle trips are simply paths from some starting vertex to some ending vertex that must agree with the direction of any directed edge taken. A loop detector is a device that counts the number of vehicles that cross an edge during some time period. Loop detectors are typically present on only a subset of the edges in the network. The basic problem we are interested in is to determine the minimum number of trips (simple paths) needed to explain all of the loop detector count measurements. We also consider a dynamic version of the problem in which time is discretized and vehicles move one edge per time step. In this case, the loop detectors provide traffic counts for each time step and the goal is again to determine the fewest number of trips needed to explain the data. Trips are now specified by a path and a starting time.

I. INTRODUCTION

Networks arise everywhere in the modern world, from roadways to the Internet, networks are responsible for delivering us, and our ideas, to everyone else. Restricting our attention to road networks, we can represent the roadways as edges in a graph and intersections as vertices. Furthermore, the flow in this graph is representative of vehicles traveling from one location to another. We refer to individual vehicle paths as trips that describe the vehicular flow. Determining detailed trip information about vehicles is a difficult task without monitoring each individual vehicle. However, due the prevalence of loop detector data, it is not difficult to represent the network as a graph with a vehicle count on various edges. In this work we consider the problem of finding the smallest number of trips that could be responsible for the known edge volumes in the given graph.

We consider a network with a volume associated with some edges in the graph. A volume represents the number of vehicles traversing that edge. We discuss various problem formulations, each of which consider different types of graphs or trip decompositions. We cover complexity results and preliminary ideas for algorithmic solutions for inferring trips from these graphs.

Providing the underlying trips responsible for flow volumes in a network offers insight into the structure of vehicular flows. This structure can help provide input to planning and routing algorithms to make vehicles travel more efficiently, ultimately reducing travel times and congestion in various network settings.

In previous work, the problem of flow decomposition for individual commodities has been considered [1], [2], [3]. In the research done by Vatinlen et al., the authors introduce two greedy heuristics for extracting the smallest number of paths from a s-t multipath flow [1]. The work done by Hartman et al. extend the work done by Vatinlen et al. by comparing their heuristics against new methods [2]. Specifically, the authors introduce a new approximation algorithm that decomposes the flow into no more than \((1/e^2)\) times the optimal number of paths. Two versions of their approximation algorithm are compared against the heuristics introduced by [1] and the width-based decomposition algorithm performs almost as well as the original width-based heuristic. Mumey et al. consider the same problem but offer improvements to the approximation bound [3]. The new algorithms are logarithmically bounded by the length of the longest path in the flow and the largest flow volume on any particular edge. We divert from previous research by considering graphs that do not have pre-defined sources and destinations. This formulation more accurately represents instantaneous edge volume data in a generic network, such as a roadway. Furthermore, we are interested in finding plausible trips in order to extract more precise information about individual agents in the graph. This type of information can provide insight into the current status of road networks, such as a lower bound on the number of vehicles on a road system at a given time.

II. PROBLEM FORMULATION

To better understand graph flows in the context of road networks and vehicles, an accurate model of the roadway must be considered. To represent the roadway, we consider a mixed graph defined as \( G = \langle V, E, A \rangle \) where \( V \) is the set of vertices (intersections), \( E \) is the set of undirected edges, and \( A \) is the set of directed edges, or arcs. We define a partial volume function \( vol : E \cup A \rightarrow \mathbb{Z}_+ \). In other words, edges in some subset of \( E \cup A \) have positive integer volumes associated with them, while some edges may remain unlabeled. Undirected edges are representative of roadways that can be traversed in either direction. Then, with vehicles, we are interested in the problem of identifying individual paths which can be used to identify large trends in movement data that might, otherwise, go unnoticed. Specifically, we want our decomposition to be composed of the fewest number of vehicle trip paths, hereon referred to as trips, that could explain all edge counts in the graph. Formally, \( trips \) are walks through the graph that do not contain repeated edges or vertices, i.e. simple paths. We restrict our attention to trips to avoid single vehicles explaining all of the traffic around cycles in the graph. When a trip traverses an edge, it accounts for a single unit of volume. We refer to this formulation as the Minimum Road Trips (MRT) problem.
Lemma 1. MRT is NP-hard.

Proof. We show that the problem is NP-hard with a reduction from 3-SAT. The 3-SAT problem asks whether there exists a truth assignment to variables $x_1, \ldots, x_n$ that will satisfy the boolean formula with $m$ clauses which each contain three variables: $(x_1 \lor x_2 \lor x_3) \land \ldots \land (x_{n-1} \lor x_n \lor x_1)$. For an instance of 3-SAT with $n$ variables, we create a corresponding instance of MRT such that there is an solution of the MRT instance with exactly $n$ trips if and only if the 3-SAT instance is satisfiable. The construction is as follows: We begin by creating an undirected edge for each variable $x_i$ for $i \in \{1, \ldots, n\}$, denoted as $e_{x_i}$. Set $vol(e_{x_i}) = 1$ for all variable edges. The left endpoint of a variable edge is associated with $\neg x_i$ and the right endpoint with $x_i$. Next, we create clause gadgets, each clause gadget is a directed cycle consisting of six edges, three of which with unit volume. Arrange the clause gadgets vertically as shown in Fig. 1. Beginning with the top clause, we create an edge to each literal in the clause. This edge originates at either the associated variable edge, if that literal has not appeared before in a clause, or from the last clause it was used in. We also label this newly created edge with the literal. Note that the edges between clauses are directed downwards (towards higher number clauses). Note, that a variable traversing a clause does not result in a cycle because the variable can exit the clause one edge before repeating a vertex while simultaneously covering all unit edges in the clause.

Suppose the 3-SAT instance does have a satisfying assignment. We can create a trip solution for the MRT instance by creating a trip that originates at each variable edge; if $x_i$ is true in the assignment then this trip originates at $\neg x_i$ and leaves the edge from $x_i$. If the trip enters a clause gadget, and the edges of the clause gadget have not yet been traversed, then the trip makes a traverse around the clause gadget prior to continuing if there is an outgoing edge with that literal label. Since each clause is satisfied by the truth assignment, some trip will reach the endpoint with that literal label. Observe that each trip must originate at a unique variable edge and there are $n$ variable edges and $n$ trips. Next, we argue that we can modify the trip solution so that all trips agree with the edge labels created. Observe that if one trip visits a clause gadget, then it must circle the gadget and return to the literal that entered on. If the trip continues it must do so on the correctly-labeled exiting edge. If two trips enter the gadget, then it is possible that the trips exit on the wrong label. However, we can do a path swap to modify these trips so that they continue on the correct labels, as shown in Fig. 2. Similarly, if three trips enter a clause gadget, it is easy to see that they can be path-swapped if needed so that all exiting trips do so on the correct labels. This process of path-swapping is continued until all trips agree with all edge labels. We set the truth value of each variable $x_i$ according to which endpoint the trip leaves $e_{x_i}$. Since each clause gadget is now circled by a single literal (assigned true), this truth assignment satisfies all clauses. \hfill $\Box$

Fig. 1: The MRT instance corresponding to the 3-SAT instance $(x_1 \lor x_2 \lor x_3) \land (\neg x_1 \lor x_2 \lor x_4)$.

Fig. 2: By path swapping, all trips passing through a clause gadget can be routed to agree with their edge labels.

III. A Heuristic MRT Algorithm

In this section, we introduce a heuristic for the MRT Problem. The algorithm works by finding a maximal trip in the network, adding it to the set of trips $\tau$, reducing the remaining volume of the volume-labeled edges along the path by 1 and repeating this process until all the volumes are accounted for. We refer to this algorithm as the Greedy Trips (GT) algorithm. A maximal trip $p$ is constructed by starting with some edge that has positive volume and greedily extending the path in both directions until further expansion is not possible without adding an already-used vertex. The trip $p$ is then trimmed so that it starts and ends with edges with positive volume, and added to the MRT trips solution. Volume labels along $p$ are decremented by 1 and the GT algorithm continues until no edges with positive volume remain. Pseudocode can be found in Algorithm 1.

The trips returned by the GT heuristic will be a valid MRT solution as it accounts for all edge volumes in $G$. However, the solution is not necessarily minimal and we have not yet determined any performance guarantees for this heuristic on the general MRT problem.

IV. Variations on MRT

A. Directed Acyclic Graphs with No Missing Measurements

Under certain circumstances, we may have a graph that does not contain any undirected edges or cycles and has fully known
edge volumes. When the graph contains no undirected edges, no cycles, and there are no unknown edge volumes, i.e., we have a Directed Acyclic Graph (DAG), the problem becomes solvable in polynomial time with the heuristic described in Section III.

Lemma 2. For the MRT problem where the graph is a DAG with no unknown edge counts, the GT algorithm produces (in polynomial time) an optimal trip solution.

Proof. Let \( \tau \) be the solution returned by the GT algorithm and let \( \tau^* \) be an optimal solution. We will show that \( |\tau| = |\tau^*| \) by demonstrating that every trip in \( \tau \) is either already in \( \tau^* \), or that the trip can exist in \( \tau^* \), without producing any additional trips, through path swapping. Let \( p \) be a trip returned in \( \tau \), if \( p \in \tau^* \), then \( p \) is compatible with the optimal solution and we are done. However, if \( p \) is not in \( \tau^* \), then \( p \) must be covered by a set of trips \( \{t_1, \ldots, t_k\} \in \tau^* \). We order these trips in \( \tau^* \) as they appear when we follow \( p \) from the source to the sink. Note that, since \( p \) is maximal, the first trip \( t_1 \) must start at \( p \)'s source vertex and the final trip \( t_k \) covering \( p \) must end at \( p \)'s sink vertex. Then, we can transform the trips that cover \( p \) in \( \tau^* \) into \( p \) without adding any extra trips, demonstrating that \( p \) is compatible with the optimal solution. We start at \( p \) and \( t_1 \)'s source and follow \( p \) until \( t_1 \) diverges, at this divergence, we note that \( t_2 \) must enter the trail that \( p \) follows in order to cover the volume that \( t_1 \) misses by diverging. Then, we perform a path swap with \( t_1 \) and \( t_2 \) so that \( t_2 \) continues where \( t_1 \) diverged and \( t_1 \) continues to follow \( p \). We continue to perform path swaps on \( t_1 \) and \( t_k \), for \( 1 < k < i \), until we reach trip \( t_i \), swapping \( t_i \) and \( t_i \) allows us to set the sink of \( t_i \) to be equal to the sink of \( p \), making \( p = t_1 \) and proving that \( p \) is compatible with \( \tau^* \) without adding any additional trips. The described procedure can be repeated on all trips in \( \tau \) to show that \( \tau = \tau^* \) through a series of path swaps.

Finally, we must show that the algorithm runs in polynomial time. First, we consider time to find a maximal trail; in the worst case, we have to traverse every edge in the graph, so the time is bounded by \( O(|E|) \). The number of iterations of the loop in the GT algorithm is bounded by \( S = \sum_{e \in E \cup A} vol(e) \), since each maximal trail found decreases the volume remaining on at least one edge. Each loop iteration takes at most \( O(|E|) \) time. Thus, the running time of the GT algorithm is \( O(S|E|) \).

Algorithm 1 Greedy Trips
\[
\begin{align*}
\tau &= \emptyset \\
\text{while } &\exists e \in G \text{ with } vol(e) > 0 \text{ do} \\
&\quad \text{Find a maximal trip } p \in G \text{ as described} \\
&\quad \tau = \tau \cup \{p\} \\
&\quad \text{Decrement volume-labeled edges in } p \text{ by 1} \\
\text{end while} \\
\text{return } \tau
\end{align*}
\]

B. Dynamic Minimum Road Trips Problem

In this section, we consider a mixed dynamic graph with edge volumes known for a subset of the edges. In essence, this version of the problem is a direct extension of the formulation discussed in Section II, but with a dynamic graph. We refer to the problem as the Dynamic Minimum Road Trips (DMRT) Problem and ask for the smallest set of trips that explain the edge volumes in the dynamic, time-varying, road network. The network is represented as a mixed graph \( G = (V, E, A) \) where \( V \) is the vertices and \( E \) and \( A \) represent the undirected and directed edges respectively. We denote the lifetime of \( G \) to be \( T \), where \( T \) is a positive integer value (i.e. \( lt(G) = T \)). Edges \( e \in E \) and \( a \in A \) may have a volume \( vol(e, t) \) or \( vol(a, t) \) at each discrete time \( t \in [1, \ldots, T] \) over the lifetime of \( G \) representative of the number of trips using that edge during that time. The volume of some edge or arc is either a positive integer, or unknown. Specifically, we have a partial volume function \( vol : (E \cup A, t) \rightarrow \mathbb{Z}_+ \). The solution to the DMRT Problem is the smallest set of trips that explain the edge volumes in \( G \). We assume that it takes each flow one time unit to traverse any edge in the graph.

DMRT with No Undirected Edges

In this section we describe a reduction from the special case of DMRT where \( E = \emptyset \), i.e. all edges in the graph are directed, to the original MRT problem. Given a DMRT instance \( G = (V, E, A) \), recall that the lifetime of the network \( lt(G) = T \), then we have \( T \) discrete time intervals from \([1, T]\). For each \( t \in [1, k] \) we create a new copy of \( G \) at time \( t \) and refer to it as \( G_t = (V, E_t, A_t) \). We duplicate each \( v_{i,t} \in V_t \) into \( v_{i,t}^i \) and \( v_{i,t}^o \). We set any existing arc \((v_i, v_j)\) to \((v_{i,t}^i, v_{j,t}^o)\). This modification to the edges and vertices ensures that a trip can not travel more than one arc during a single time epoch. Then, for each modified arc \((v_i, v_j)\) set \( vol(a_t) = vol(a, t) \) where \( vol(a, t) \) is specified in the original network \( G \) at time \( t \). The construction of these graphs gives us \( T \) new copies of \( G \), each with edge weights equal to those of \( G \) at a particular time interval.

Next, we connect these graphs using directed edges to make one, large, graph. Let \( t \) be a time value in the discrete interval \([1, T - 1]\), then we connect \( G_t \) to \( G_{t+1} \) by adding an arc \( a \) from each \( v_{i,t}^o \in V_t \) to each corresponding \( v_{i,t+1}^o \in V_{t+1} \). We specify the \( vol(a) \) to be undefined so that any number of trips can travel on these edges. However, a minimal trip solution will not generate extra trips that are not necessary to explain all labeled volumes in the graph. This ensures that extra trips will not be introduced on these directed edges connecting the graphs.

See Figure 3 for an example of the reduction, shown in the gray boxes. The original dynamic graph, shown at the bottom of the figure and labeled \( G \), has only four vertices and a lifetime of three. The edge volumes at each time interval are clarified in the reduction, but not shown the original topology. For each time unit, a copy of the graph is stacked vertically above the last and, for clarity, each copy is encased by a gray rectangle. The dotted edges, with unspecified volume, connect...
Fig. 3: Full reduction of the DMRT problem to the MRT problem. The graph at the bottom of the figure is the original topology of $G$. The reduction shows each graph at some discrete time enclosed by a gray box the dotted, directed, edges connect the different copies of the graph.

V. DISCUSSION AND FUTURE WORK

In this research, we have discussed various formulations related to the determining trips from traffic volume information in road networks. Solutions from different formulations can be used to answer different types of questions. For example, a solution to the MRT problem can provide a lower bound for the number of drivers on a road system at a given time. Alternatively, a solution to the DMRT problem can provide a set of potential vehicle routes over a specific time interval. In some preliminary simulations we have highlighted the proportion of correctly discovered trips using the heuristic described in the paper on three different types of synthetically generated 10x10 grid graphs and trips (i.e. simple random walks). Specifically, for each number of trips 50 dag known graphs (DAG with all edge weights known), 50 dag unknown graphs (DAG with some unknown edge weights), and 50 mixed graphs (undirected and directed edges with some missing edge weights) were generated. Then, the heuristic was used to discover trips in each graph, Figure 4 shows the average proportion of correctly identified trips on each type of graph for different numbers of synthetically generated trips. We can see that fewer walks leads to less ambiguity, presumably because there will be less overlap between the trips and therefore, less of an opportunity to merge them or swap their paths. The figure demonstrates that accuracy decreases as edge volumes are removed and, then again, as undirected edges are introduced.

Future work includes the development of additional heuristics and approximation algorithms for the these formulations as well as generalizing the DMRT reduction to work for undirected edges. We would like to introduce other techniques to help improve the trip extraction process to more accurately represent real paths taken by drivers. Future simulations will use real loop detector data sets from actual road networks.

REFERENCES