Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 24, no. 3, pp. 523–546 (2020)
DOI: 10.7155/jgaa.00544

Problems on One Way Road Networks

Jammigumpula Ajay[1] *Avinandan Das*[2] *Binayak Dutta*[3]
Arindam Karmakar[3] *Sasanka Roy*[1] *Navaneeta Saikia*[3]

[1]Indian Statistical Institute, Kolkata
[2]Chennai Mathematical Institute, Chennai
[3]Tezpur University, Tezpur

Abstract

A One-Way Road Network is an ordered pair $OWRN = (W_x, W_y)$ comprising of a set W_x of m directed horizontal roads along with another set W_y of n directed vertical roads. An $OWRN$ can also be viewed as a directed grid graph $GG = (V, E)$, where V corresponds to intersections between every pair of horizontal and vertical roads, and there is a directed edge between every pair of consecutive vertices in V in the same direction corresponding to that road. A *vehicle* c is defined as a 3-tuple (t, s, P), where c starts moving at time t and moves with a constant speed s from its start vertex to destination vertex along pre-specified directed path P, unless a collision occurs. A collision between a pair of vehicles c_i and $c_j (i \neq j)$ occurs if they reach a vertex $v \in V$ (a junction in $OWRN$) orthogonally at the same time. A traffic configuration on an $OWRN$ is a 2-tuple $TC = (GG, C)$, where C is a set of vehicles, each travelling on a pre-specified path on GG. A collision-free TC is a traffic configuration without any collision. We prove that finding a maximum cardinality subset $C_{max} \subseteq C$, such that $TC = (GG, C_{max})$ is collision-free, is NP-hard. We also show that GG can be preprocessed into a data-structure in $\mathcal{O}(n + m)$ time and space, such that the length of the shortest path between any pair of vertices in GG can be computed in $\mathcal{O}(1)$ time and the shortest path can be computed in $\mathcal{O}(p)$ time, where p is the number of vertices in the path.

Keywords: Directed graphs, Independent sets, Traffic configuration, Shortest path

Submitted: July 2019	Reviewed: October 2019	Revised: April 2020	Reviewed: May 2020
Revised: August 2020	Accepted: October 2020	Final: October 2020	Published: October 2020
Article type: Regular paper		Communicated by: G. Barequet	

Preliminary version appeared in CCCG 2016

E-mail addresses: ak.jammi@gmail.com (Jammigumpula Ajay) adas33745@gmail.com (Avinandan Das) binayak66@gmail.com (Binayak Dutta) arindam@tezu.ernet.in (Arindam Karmakar) sasanka.ro@gmail.com (Sasanka Roy) saikia.navaneeta@gmail.com (Navaneeta Saikia)

1 Introduction

The rapid development in the existing motor vehicle technology has led to an increase in the demand for automated vehicles, which are themselves capable of various decision activities such as motion-controlling, path planning. This has motivated many researchers to address a large number of algorithmic and optimisation problems. Automated transportation is a field with a rich history and great relevance. One shall find the reason to address the problem that we study in this paper, *one-way road networks* or *OWRN*, in the fields that involve automated transportation. Imagine a port with thousands of evenly spaced shipping containers. The job at hand is to fill the containers with their respective goods, which are to be shipped. We have automated trolleys that transport the goods from the warehouse to the containers and the only available paths (one-way) are the space between containers. The objective is to find the maximum number of trolleys that can be deployed in this setting so that all the trolleys reach their designated containers and no two trolleys collide. This is the type of setting where *OWRN* shall manifest in practice.

The field of *automated guided vehicles*, *AGV*, has a rich history and has a close relation to our work. We refer the reader to a paper by Vis [11] for an elaborate review on the field of *AGVs*. For a more recent survey, the reader is referred to the survey by Hyla and Szpytko [4]. A very important topic of interest in the field of *AGVs* has been collision avoidance between the automated vehicles. Kim and Tanchoco [6] propose an algorithm based on Dijkstra's algorithm, for conflict-free routing of *AGVs*. The algorithm runs in $\mathcal{O}(v^4n^2)$ time, where v is the number of vehicles and n is the number of nodes. Arora et al. [1] study the problem of collision avoidance on junctions. They design a game theory based methodology for *AGV* traffic control. Yan et al. [12] propose a digraph based technique for collision-free routing of *AGVs* on both unidirectional and bi-directional paths. We refer the reader to [7, 8, 13] for more studies on the conflict-free routing of *AGVs*. For an elaborate survey on routing and scheduling algorithms for *AGVs*, we refer the reader to [9]. While these studies are focused on the problem of finding efficient and collision-free routing for a given set of *AGVs*, our work is concerned with the problem of computing, for a given set of vehicles C and a one-way road network, the subset of maximum cardinality $C_{max} \subseteq C$ such that all the vehicles $c \in C_{max}$ move through their pre-specified paths without colliding with each other.

The 1939 paper by Robbins [10], which gives the idea of *orientable* graphs and the paper by Masayoshi et al. [5], which considered one-way paths as graphs and the optimal closed-loop is determined on the graph, motivated us to formulate our graph network. Another work from which we got motivated is by Dasler and Mount [2], which basically considers motion coordination of a set of vehicles at a *traffic-crossing* (intersection). However, unlike their work, we consider a much simpler version of a grid graph and mainly concentrate on analysing essential properties and proving the hardness of a collision-free movement of traffic in the given graph network. Furthermore, our work suggests a suitable algorithm and data structure to find the shortest path in a one-way road network.

2 Preliminary definitions and results

A road is a directed line, which is either parallel to (oriented along) X-axis (Y_i) or Y-axis (X_i) and it is uniquely defined by its direction and distance from the corresponding parallel axis. Here direction is the constraint which restricts the movement of a vehicle on the road.

Definition 1 *A road is a 2-tuple, $Y_i = (d_i, x_i)$, $X_j = (d_j, y_j)$. Here x_i is the length of the road Y_i from X-axis such that $x_i < x_{i+1}$, similarly, we define y_j. d_k is the direction of the road i.e., $d_k \in \{-1, 1\}$ (where -1 represents negative direction and 1 represents positive direction of the respective axis).*

Definition 2 *A One Way Road Network (OWRN) is a network with a set of m horizontal and n vertical Roads. Formally an $OWRN$ is a 2-tuple, $OWRN = (W_x, W_y)$, where, $W_x = \{Y_1, Y_2, Y_3 \ldots, Y_m\}$ and $W_y = \{X_1, X_2, X_3 \ldots, X_n\}$.*

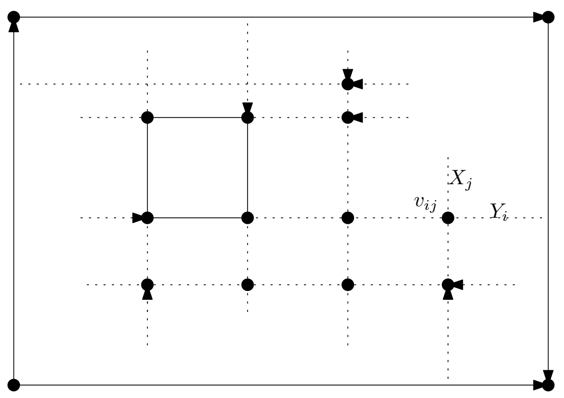

Figure 1: Graphical representation of an $OWRN$

Definition 3 *Let $V = \{v_{ij} | v_{ij}$ is the intersection of the roads Y_i and X_j, for $i \leq m, j \leq n\}$.*

Definition 4 *A directed edge $e = (u, v)$, $u, v \in V$, is a connection between two adjacent intersections on a road. Every edge is a part of a road X_i or Y_j and is in the same direction of the road to which it belongs.*

Definition 5 *The boundary roads of an $OWRN$ are the outermost roads, i.e., X_1, X_n, Y_1 and Y_m. The vertices on the boundary roads are denoted as boundary vertices. The edges joining two boundary vertices are called boundary edges. The $OWRN$ is defined within these four boundary roads. The non-boundary vertices (edges) are the vertices (edges) other than boundary vertices (edges).*

Definition 6 *Let $GG(V, E)$ be a directed grid graph defined on an $OWRN$, where the vertex set V and the edge set E are defined in Definition 3 and Definition 4, respectively.*

Definition 7 *A directed path $\pi(u_{\alpha_i}, u_{\alpha_k})$, from u_{α_i} to u_{α_k} in GG is a sequence of consecutive directed edges between a sequence of consecutive intersections $u_{\alpha_i}, u_{\alpha_{i+1}}, \ldots u_{\alpha_k}$, such that the edges are directed $(u_{\alpha_i}, u_{\alpha_{i+1}})$, $(u_{\alpha_{i+2}}, u_{\alpha_{i+3}}), \ldots,$ $(u_{\alpha_{k-1}}, u_{\alpha_k})$.*

From the above set of definitions, it becomes clear that every road $Y_i = (d_i, x_i)$ is a directed path. But every directed path is not necessarily a road. A road $Y_i = (d_i, x_i)$ is a directed path, parallel to the positive x axis, that does not change direction. Its direction is suggested by the variable d_i, which is set to -1 if its direction is opposite to the direction of positive x axis, and 1 if it is directed along the direction of positive x axis. The same is true for all road $X_j = (d_j, y_j)$ parallel to the positive y axis.

Definition 8 *A vehicle c is a 3-tuple (t, s, P), where c starts moving at time t and moves with a constant speed s from its start vertex to its destination vertex along pre-specified directed path P, unless a collision occurs. In this paper, we assume that all the vehicles move with the same constant velocity.*

Definition 9 *A traffic configuration $TC(GG, C)$ is a set of vehicles over an $OWRN$, where C is the set of vehicles $\{c_1, c_2, \ldots, c_k\}$.*

Definition 10 *A collision is said to occur when two vehicles c_i and c_j $(i \neq j)$ reach the same vertex orthogonally at the same time. So a collision-free traffic configuration is a TC without any collisions.*

2.1 Main results

In this work, we consider the problem of collision-free traffic configuration TC in GG. Our objective is to find a maximum cardinality subset $C_{max} \subseteq C$ such that $TC = (GG, C_{max})$ is collision-free. We prove that finding a maximum cardinality subset $C_{max} \subseteq C$ such that $TC = (GG, C_{max})$ is collision-free, is NP- hard. We reduce the problem of finding the maximum independent set of a graph (MIS) to the problem of Collision-free Traffic Configuration. Our reduction is a gap preserving reduction and thus, it is as hard as MIS. It is proven that no algorithm can give $n^{1-\epsilon}$ factor approximation of MIS, for any $\epsilon > 0$ [3]. We also design a data structure in $\mathcal{O}(n+m)$ preprocessing time and space such that the length of the shortest path between any pair of vertices in GG can be computed in $\mathcal{O}(1)$ time and shortest path can be computed in $\mathcal{O}(p)$ time, where p is the number of nodes in the path. We also characterize the connectedness of an $OWRN$ or GG.

2.2 Strongly-Connectedness in a One Way Road Network

A grid graph $GG(V, E)$ is strongly-connected if and only if every ordered pair of vertices $u, v \in V$ is connected by a directed path. The following lemmas capture the strongly-connectedness property of a grid graph.

Lemma 1 *For every non-boundary vertex v, there always exist two boundary vertices w and x such that there exist directed paths $\pi(v, w)$ and $\pi(v, x)$ and there always exists two boundary vertices w' and x' such that there exist directed paths $\pi(w', v)$ and $\pi(x', v)$.*

Proof: Follows directly from the properties of the roads of the $OWRN$. □

Lemma 2 *If all the boundary vertices of an $OWRN$ form a cycle, then the grid graph GG is strongly-connected.*

Proof: Let u and v be pair of vertices in an $OWRN$. We prove that there always exists a directed path $\pi(u, v)$ by considering the following cases.

- **Case 1:** Both u and v are boundary vertices. Since the boundary vertices form a cycle, therefore $\pi(u, v)$ exists.
- **Case 2:** If at least one of them is boundary vertex. Lemma 1 ensures that there exists a directed path from the boundary to the interior and vice versa.
- **Case 3:** Both are non-boundary vertices. From Lemma 1, We can find two boundary vertices x and w such that there exists $\pi(u, x)$ and $\pi(w, v)$. So there exists a directed path from u to v because we have directed paths $\pi(u, x)$, $\pi(x, w)$ and $\pi(w, v)$. Therefore $\pi(u, v)$ exists.

This concludes the proof. □

Lemma 3 *If $GG(V, E)$ is strongly-connected, then all the boundary vertices of GG form a cycle.*

Proof: We prove the lemma by contradiction. The boundary vertices are either of degree 2 or 3. The degree three vertices have two boundary edges adjacent to them, one is incoming and the other is outgoing. We assume that the boundary vertices do not form a cycle. Therefore, there exists a boundary vertex of degree 2, such that either both the boundary roads are incoming or outgoing. If both roads are incoming, we will not be able to reach any another vertex from that vertex. If both are outgoing edges, it will not be possible to reach that vertex from any other vertex. Therefore the graph is not strongly-connected, which is a contradiction. □

Theorem 1 *A One Way Road Network is strongly-connected if and only if the boundary roads form a cycle.*

Proof: The proof of this theorem follows from Lemma 2 and Lemma 3. □

3 Hardness of Collision-Free Traffic Configuration

In this section, we show that finding a solution to the traffic configuration problem is ***NP-Hard***. We reduce the Maximum Independent Set problem (MIS) of a graph to the Traffic Configuration problem. For this, we have the following theorem.

Theorem 2 *Given an undirected graph $G = (V, E)$, there exists a traffic configuration (GG, C), computable in polynomial-time, such that the cardinality of MIS of G is k if and only if the cardinality of C_{max} is k.*

For the simplicity of reduction, we first show how to construct a TC for a complete graph K_n that satisfies the properties in Lemma 4.

Lemma 4 *For any complete graph K_n, it is possible to construct a TC such that every vertex v of the K_n corresponds to a vehicle c in TC, and for every edge $e = (u, v)$ of K_n the vehicles corresponding to vertex u and v collide.*

Proof: We prove this lemma by constructing a TC for a K_n.

1. We construct an $OWRN$ with $2n$ horizontal roads $Y_i = (d_i, x_i)$ and n vertical roads $X_j = (d_j, y_j)$ as follows:
   (a) For the road Y_i, we have
   $$d_i = \begin{cases} 1 & 1 < i \leq 2n \\ -1 & i = 1 \end{cases}, \text{ and } x_i = \begin{cases} -1 & i = 1 \\ x_{i-1} + \delta & 1 < i \leq 2n \end{cases}$$
   (b) For the road X_j, we have
   $$d_j = \begin{cases} -1 & 1 < j \leq n \\ 1 & j = 1 \end{cases}, \text{ and } y_j = \begin{cases} -1 & j = 1 \\ y_{j-1} + \delta & 1 < j \leq n \end{cases}$$
   where δ is a positive real valued constant.
2. The set C of vehicles is defined as $\{c_1, c_2, c_3 \ldots, c_n\}$, each vehicle $c_i(t_i, s_i, P_i)$ is as follows:
   (a) $t_i = 0$.
   (b) $s_i = \omega$.
   (c) $P_i = \left\{v_{ri}, v_{(r-1)i} \ldots, v_{qi}, v_{q(i+1)} \ldots, v_{qn}\right\}$, where $r = n + i - 1$, $q = n - i + 1$.

Now we can observe that the two paths P_i and P_j corresponding to vehicles c_i and c_j, $(i < j)$, intersect at only one vertex $v_{(n-i+1)(j)}$. The length of the paths P_i and P_j from their respective start vertices to the vertex $v_{(n-i+1)(j)}$ is same which is equal to $(i + j - 2) \times \delta$. Since all the vehicles start at the same time and travel with the same constant speed, c_i and c_j will reach the vertex

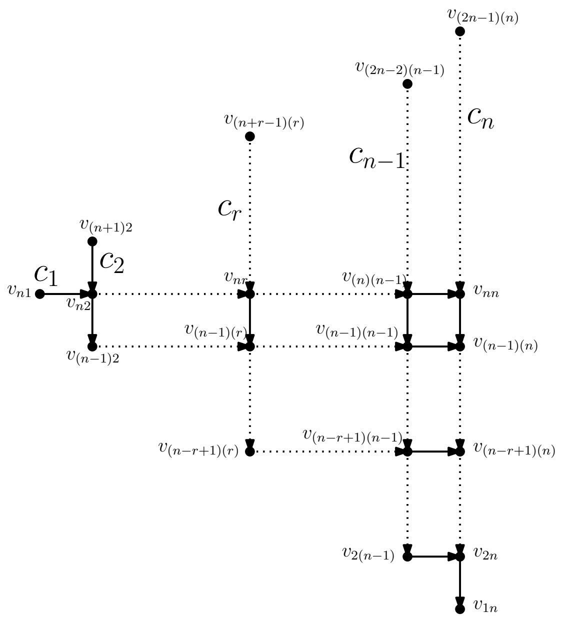

Figure 2: Paths for vehicles $\{c_1, c_2, \ldots, c_n\}$ in the TC obtained from Lemma 4

$v_{(n-i+1)(j)}$ at same time in mutually orthogonal directions. Hence, a collision occurs. Thus, we obtain the corresponding $TC = (GG, C)$ for a K_n. □

Before we reduce the MIS problem for general graphs to the TC problem, for better understanding, we first explain the characterizations of the paths P_i's of TC that is constructed from K_n. Each path P_i has $(n + i - 2)$ edges. P_i is the concatenation of two subpaths $P_i = P_i^x P_i^y$ of lengths $2(i - 1)$ and $(n - i)$, respectively. Each edge in the subpath P_i^x is vertical and directed downwards while the edges in the subpath P_i^y is horizontal and directed towards right.

Now we show how to reduce the MIS problem on any simple graph G with n vertices to a TC. We first construct the TC for a complete graph K_n. We construct modified TC_m by an incremental method from TC. We will consider the vertices of G in any arbitrary order $U = (u_1, u_2, \ldots, u_n)$ to create the set of vehicles C such that a vehicle $c_i \in C$ corresponds to a vertex $u_i \in G$.

To achieve the above construction of modified TC_m, we construct a modified $OWRN_m$ by inserting few horizontal and vertical roads on $OWRN$ of K_n from Lemma 4 as follow:

1. We insert four equally spaced horizontal roads with directions $\{1, -1, 1, -1\}$ between every two adjacent roads Y_i, Y_{i+1} and name them $Y_i^1, Y_i^2, Y_i^3, Y_i^4$.

2. We insert one vertical road between every two adjacent roads X_j, X_{j+1}, which is the perpendicular bisector of lines containing roads X_j and X_{j+1} and directed downwards, and name it X_j^0.

For convenience in the modified $OWRN$ we name the vertex formed by the intersection of X_j and Y_i^k as v_{ij}^k, where $k = 1, \ldots, 4$, the vertex formed by the intersection of X_j and Y_i as v_{ij}, the vertex formed by intersection of X_j^0 and Y_i as v_{ij}^0, and the vertex formed by intersection of X_j^0 and Y_i^k as v_{ij}^{k0}, refer to Figure 3.(b).

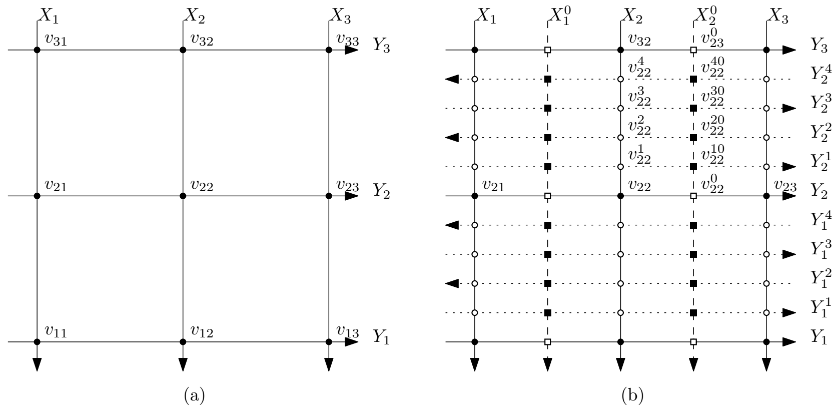

Figure 3: (a) A portion of $OWRN$ and (b) A portion of $OWRN_m$

Note that, now we have a modified $OWRN$ of $(10n - 4) \times (2n - 1)$ roads, with $(10n - 4)$ horizontal roads and $(2n - 1)$ vertical roads.

We redefine the path of vehicles in the $OWRN_m$ such that every vehicle still collides (like $OWRN$ of K_n) with all other vehicles. For each vehicle c_i in TC_m we define its path in $OWRN_m$ as
$P_i = \left\{v_{ri}, v_{(r-1)i}^4, v_{(r-1)i}^3, v_{(r-1)i}^2, v_{(r-1)i}^1, v_{(r-1)i} \ldots, v_{qi}, v_{qi}^0, v_{q(i+1)} \ldots, v_{qn}\right\}$, where $r = n + i - 1$, $q = n - i + 1$.

Note that, $P_i \in TC_m$ is obtained by subdividing $P_i \in TC$ with new vertices because we have inserted horizontal and vertical roads in between. Now onwards $P_i \in TC_m$ would mean same path $P_i \in TC$ with those extra vertices. Thus, we also define $P_i = P_i^x P_i^y$ in TC_m. From Lemma 4, it is evident that every pair of vehicles still collide.

We propose two types of *delays*. A delay is a small detour in a path $P_i \in TC_m$, between any two specified vertices, particularly on the vertical subpath P_i^x. Let $P_i^x = \left\{\ldots, v_{kl}, v_{(k-1)l}^4, v_{(k-1)l}^3, v_{(k-1)l}^2, v_{(k-1)l}^1, v_{(k-1)l}, \ldots\right\}$. We define a single delay between the vertices v_{kl} and $v_{(k-1)l}$ in the path P_i^x, as follows
$\left\{\ldots, v_{kl}, v_{(k-1)l}^4, v_{(k-1)(l-1)}^{40}, v_{(k-1)(l-1)}^{30}, v_{(k-1)l}^3, v_{(k-1)l}^2, v_{(k-1)l}^1, v_{(k-1)l}, \ldots\right\}$.
Instead of reaching to vertex $v_{(k-1)l}$ directly from v_{kl}, it takes a detour of length $\frac{6}{5}\delta$ such that the length of the path is increased by δ. We refer the reader to Figure 4.

Similarly, we define two delays (two single delays) between the vertices v_{kl} and $v_{(k-1)l}$ in the path P_i^x as $\{\ldots, v_{kl}, v_{(k-1)l}^4, v_{(k-1)(l-1)}^{40}, v_{(k-1)(l-1)}^{30}, v_{(k-1)l}^3, v_{(k-1)l}^2,$

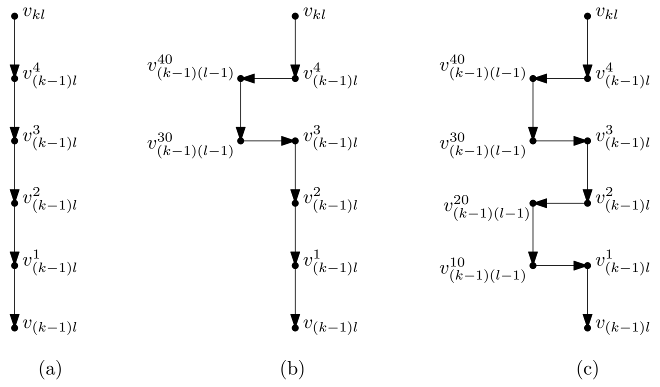

Figure 4: Path P_i^x from v_{kl} to $v_{(k-1)l}$ with (a) No delay (b) Single delay (c) Double delay

$v^{20}_{(k-1)(l-1)}, v^{10}_{(k-1)(l-1)}, v^{1}_{(k-1)l}, v_{(k-1)l}, \dots\}$. Here, the length of the path is increased by 2δ.

We keep modifying the path $P_i \in TC_m$ of a vehicle c_i, that corresponds to vertex u_i in G, to make sure that a collision between vehicles c_i and c_j in TC_m occurs if and only if there is an edge between u_i and u_j in G, $j < i$. While updating the path $P_i \in TC_m$, we shall make further necessary modifications by inserting some delays in the path $P_i \in TC_m$ such that it satisfies the following four properties:

Property 1 *If there is an edge between vertex u_i,u_j $j < i$ in G, then the vehicles c_i and c_j reach vertex $v_{(n-j+1)(i)}$ at the same time (having collision)*

Property 2 *If there is no edge between vertex u_i,u_j $j < i$, then the vehicles c_i and c_j should reach vertex $v_{(n-j+1)(i)}$ at different times (no collision)*

Property 3 *The number of delays introduced in the path P_i of the vehicle c_i before reaching vertex $v_{(n-i+1)(i)}$ is $i-1$*

Property 4 *For any two vehicles c_i, c_j, $(j < i)$, the number of delays introduced in path P_i before vertex $v_{(n-j+1)(i)}$ is $j-2$ if there is no edge between vertices u_i, u_j in G. Else, the delay is $j-1$.*

It is easy to see if the TC_m satisfies properties 1 and 2 then, G has an independent set of size k if and only if there are k vehicles that are collision-free in TC_m.

Furthermore, since all the horizontal roads are equidistant $(\frac{1}{2}\delta)$ according to our construction, and if a delay is introduced only in vertical part of the path, then the distance of the path is increased by δ, since after reaching vertex $v^4_{(k-1)l}$ [or $v^2_{(k-1)l}$], in the modified path, the vehicle c_i must take a horizontal road till it reaches the vertex $v^{40}_{(k-1)(l-1)}$ [or $v^{20}_{(k-1)(l-1)}$], then again take a vertical road till it reaches vertex $v^{30}_{(k-1)(l-1)}$ [or $v^{10}_{(k-1)(l-1)}$], and then take a horizontal road back to $v^{3}_{(k-1)l}$ [or $v^{1}_{(k-1)l}$]. From our construction, the distance from $v^{4}_{(k-1)l}$ [or $v^{2}_{(k-1)l}$] to $v^{40}_{(k-1)(l-1)}$ [or $v^{20}_{(k-1)(l-1)}$] is $\frac{1}{2}\delta$, the distance from $v^{40}_{(k-1)(l-1)}$ [or $v^{20}_{(k-1)(l-1)}$] to $v^{30}_{(k-1)(l-1)}$ [or $v^{10}_{(k-1)(l-1)}$] is same as the distance from $v^{4}_{(k-1)l}$ [or $v^{2}_{(k-1)l}$] to $v^{3}_{(k-1)l}$ [or $v^{1}_{(k-1)l}$], and the distance from $v^{30}_{(k-1)(l-1)}$ [or $v^{10}_{(k-1)(l-1)}$] to $v^{3}_{(k-1)l}$ [or $v^{1}_{(k-1)l}$] is $\frac{1}{2}\delta$.

If after performing certain modifications in the new TC_m, there exist two vehicles c_i and c_j ($j < i$), such that they reach the vertex $v_{(n-j+1)(i)}$ at same time, then the collision can be avoided by introducing a delay of δ in the path of c_i before vertex $v_{(n-j+1)(i)}$. Also, if a pair of vehicles c_i and $c_j (j < i)$ do not collide in TC_m, i.e, they reach the vertex $v_{(n-j+1)(i)}$ at different times, then by introducing delays such that both the paths before $v_{(n-j+1)(i)}$ have an equal number of delays, a collision can be created.

To enforce the Properties 3 and 4, we introduce no delays for c_1, and for c_2, if (u_1, u_2) are connected by an edge in graph G, then introducing a single delay in its path P_2 after the vertex v_{n2} shall cause a collision between c_1 and c_2. Otherwise, introducing a delay before the vertex v_{n2} does not cause a collision between c_1 and c_2. In both the cases, c_2 will have 1 delay introduced in the vertical component of its path. This serves as our base case. Now, we prove this by double induction.

We assume that for all $j < i$, in the modified TC_m so far all the four properties are satisfied. Consider the vehicles c_i, c_j and c_{j-1} in the path of c_i. All the four properties (Property 1 to Property 4) are valid till vertex $v_{(n-j+2)(i)}$, the common vertex in path of P_i and P_{j-1}. For our incremental construction it is sufficient to show that all these four properties (Property 1 to Property 4) can be maintained for vertex $v_{(n-j+1)(i)}$, which is common vertex for paths P_i and P_j.

We have the following four cases, and for all cases we show how to satisfy all the properties (Property 1 to Property 4) at vertex $v_{(n-j+1)(i)}$ as follows:

- **Case 1:** u_i and u_j are connected by an edge in G, u_i and u_{j-1} are connected by an edge in G, then introduce a single delay in between vertices $v_{(n-j+2)(i)}$ and $v_{(n-j+1)(i)}$ in path P_i, since P_i already has $j-2$ delays before vertex $v_{(n-j+2)(i)}$.

- **Case 2:** u_i, u_j are connected by an edge in G and u_i, u_{j-1} are not connected by an edge in G, then introduce two delays in between vertices $v_{(n-j+2)(i)}$ and $v_{(n-j+1)(i)}$, since P_i has $j-3$ delays in its path before vertex $v_{(n-j+1)(i)}$.

- **Case 3:** u_i, u_j are not connected by an edge in G and u_i, u_{j-1} are connected by an edge in G, then no delays is introduced between vertices $v_{(n-j+2)(i)}$ and $v_{(n-j+1)(i)}$ in path P_i.

- **Case 4:** u_i, u_j are not connected by an edge in G and u_i, u_{j-1} are not connected by an edge in G, then introduce a single delay in between vertices $v_{(n-j+2)(i)}$ and $v_{(n-j+1)(i)}$ in the path P_i, since P_i has $j-3$ delays before vertex $v_{(n-j+1)(i)}$.

In all the aforementioned four cases, we made sure the four properties are satisfied, and successfully modified T_m till c_i and c_j. In this way we can increment j till $i-1$ and modify the path of c_i while maintaining all the four properties. Note that, if u_i, u_{i-1} are connected by an edge in G then we add a delay in between vertices $v_{(n-i+2)(i)}$ and $v_{(n-i+1)(i)}$. Else, we introduce two delays in between the vertices $v_{(n-i+2)(i)}$ and $v_{(n-i+1)(i)}$ in the path P_i, to maintain Property 3. Hence, in the incremental method, we can construct a TC_m satisfying all the four properties (Property 1 to Property 4) for any given graph G.

Now according to our construction, it is easy to see that no two vehicles have more than one common vertex in their paths, and each delay increases the length of the path by the same distance. Let GG_m be the graph of $OWRN_m$.

Lemma 5 *Let C_{sub} be any subset of C in TC_m, such that $TC_{new} = (GG_m, C_{sub})$ is collision-free. Then C_{sub} corresponds to Independent Set of G and vice-versa.*

Proof: Since TC_{new} is collision-free, therefore no two nodes in the graph G, which correspond to the respective vehicles in C_{sub}, are connected by an edge. Thus, we can claim that C_{sub} corresponds to an independent set in G. On the other hand, we know from Property 2 that if there is no edge between the vertices u_i and u_j $(j < i)$ in G then in the corresponding TC_m there is no collision between vehicles. We know from Property 1 that if there is an edge between vertices u_i and u_j $(j < i)$ in G then there is a collision between the vehicles in the corresponding TC_m. Hence, an independent set in G corresponds to a C_{sub} in TC_m. $\square$

From Lemma 5, we can say that maximum C_{sub}, i.e C_{max}, corresponds to *Maximum Independent Set* in G and vice-versa. Now, from our construction and Lemma 5 we can prove that the traffic configuration problem is *NP-Hard*. Thus, we have Theorem 2 and we have the following corollary of Theorem 2.

Corollary 1 *The collision-free traffic configuration problem cannot have a better approximation than maximum independent set.*

4 Properties of Shortest Path

The length of the shortest path between a pair of vertices in an $OWRN$ may not be the Manhattan distance. There may be a pair of neighbouring vertices which are the farthest pair of vertices in the $OWRN$ metric because of one-way direction.

Definition 11 *A turn in a path P is defined as two consecutive edges belonging to two orthogonal roads.*

We redefine a directed path as traversal which will help us in proving the following lemmas, theorems and the resulting shortest path algorithm. The traversal is essentially the roads taken to reach from the source to the destination, i.e instead of writing all the edges we combine all the consecutive edges belonging to the same road and replace it with this road. Since a path is directed, therefore a traversal is also directed. Note that, since in a path we cannot visit a vertex twice, no vertex will be visited twice in the corresponding traversal.

Let $r_{(i)(j)}$ represent the common vertex to two orthogonal roads R_i and R_j $(i < j)$ in an $OWRN$. Now, formally we define a traversal as follows.

Definition 12 *A traversal $T(s,d) = (R_1, R_2, \ldots, R_k)$ of $k-1$ turns from a source s to a destination d consists of k road segments, where*

(i) R_{i-1}, R_i are orthogonal to each other for all $1 < i \leq k$.

(ii) There is a directed path from s to $r_{(1)(2)}$, such that all the edges in this path belong to road R_1.

(iii) There is a directed path from $r_{(i-1)(i)}$ to $r_{(i)(i+1)}$, such that all the edges in this path belong to road R_i, for all $1 < i < k$.

(iv) There is a directed path from $r_{(k-1)(k)}$ to d, such that all the edges in this path belong to road R_k.

(v) The length $L(T)$ of $T(s,d)$ is sum of the lengths of all the directed segments in $T(s,d)$.

Definition 13 *Let $T(s,d) = (R_1, R_2, \ldots, R_k)$ be a traversal of length D in an $OWRN$. $T(s,d)$ is called a valid traversal, if we can modify the direction of roads passing through the source s and the destination d without modifying the directions of roads $R_i \in T(s,d), \forall 1 \leq i \leq k$ in the $OWRN$, such that no traversal of less than $k-1$ turns has length at most D.*

Definition 14 *A traversal is said to be invalid when no such modification of direction, as mentioned in Definition 13, is possible, i.e, there is another traversal which uses less turns (or roads) to reach from the source to the destination without increasing the total length.*

Definition 15 *We define $T_{sub}(p,q) = (R_i, R_{i+1}, \ldots R_j)$ as a sub-traversal of $T(s,d) = (R_1, R_2, \ldots, R_k)$, for all $1 \leq i < j \leq k$, where*

$$p = \begin{cases} r_{(i-1)(i)} & 1 < i \leq k \\ s & i = 1 \end{cases}, \text{ and } q = \begin{cases} r_{(j)(j+1)} & i \leq j < k \\ d & j = k \end{cases}$$

and hence, $T(s,d)$ is also a sub-traversal of itself.

Lemma 6 *A traversal $T(s,d) = (R_1, R_2, \ldots, R_k)$ is valid if and only if every sub-traversal of $T(s,d)$ is valid.*

Proof: First we will prove the if part by contradiction, suppose the traversal $T(s,d)$ is valid, and there is a sub-traversal $T_{sub}(p,q) = (R_i, R_{i+1}, \ldots R_j)$ which is invalid, since $T_{sub}(p,q)$ is invalid for any modification of $OWRN$ there exists a traversal $T'_{sub}(p,q) = (R_\alpha, \ldots, R_\beta)$ with fewer turns (or roads) than $T_{sub}(p,q)$ and with length at most that of $T_{sub}(p,q)$.

Consider $T(s,d)$, we can divide it into three sub-traversals as follows $T(s,d) = T_{sub}(s,p) + T_{sub}(p,q) + T_{sub}(q,d)$, now we can write a new traversal $T'(s,d) = T_{sub}(s,p) + T'_{sub}(p,q) + T_{sub}(q,d)$, hence, $T'(s,d)$ has fewer turns than $T(s,d)$ and also the length is at most that of $T(s,d)$.

Hence, we can write $T'(s,d) = (R_1, \ldots, R_{i-1}, R_\alpha, \ldots, R_\beta, R_{j+1}, \ldots, R_k)$. Note that, if there are two consecutive roads which are same then remove one from the list of roads, the final list of remaining roads will be the roads used by traversal, which is a contradiction.

Proving the only if part is easy since every sub-traversal is valid and $T(s,d)$ is a sub-traversal to itself, hence, the only if part is true. □

Definition 16 *We define $\Delta(OWRN) = (\delta_1, \delta_2, \ldots, \delta_r)$ to be a sequence of operations, i.e, each δ_l is applied on the $OWRN$ sequentially, where δ_l is either rotation of plane of $OWRN$ by 90° or taking the mirror image of the $OWRN$ along the y-axis.*

Definition 17 *Two traversals $T(s,d) = (R_1, R_2, \ldots, R_k)$ in an $OWRN$, and $T'(s',d') = (R'_1, R'_2, \ldots, R'_k)$ in an $OWRN'$, are said to be struct-alike if there exists a $\Delta(OWRN')$ such that $R'_i \in T'(s',d')$ is parallel to $R_i \in T(s,d)$ for all $1 \leq i \leq k$ and d' is in the same quadrant with respect to s' as d with respect to s.*

The Quadrant of d with respect to s is one of the four quadrants in Euclidean co-ordinate geometry in which d exists when the origin of Euclidean plane is translated to s.

Lemma 7 *If a traversal $T(s,d) = (R_1, R_2, \ldots, R_k)$ is valid in an $OWRN$ then any traversal $T'(s',d') = (R'_1, R'_2, \ldots, R'_k)$ in an $OWRN'$ which is struct-alike to T is also valid.*

Proof: Given $T(s,d)$ and $T'(s',d')$ are struct-alike. Let the operations to make roads R'_i parallel to R_i for all $1 \leq i \leq k$ be Δ, then we know that applying Δ^{-1} (inverse of Δ) on T we can make R_i parallel to R'_i. Since T is valid there exists an $OWRN_1$ in which the number of turns in the traversal cannot be reduced without increasing the length. For every road passing through vertex $r_{(i)(i+1)'}$ there exists a road passing through $r_{(i)(i+1)}$ in $OWRN_1$. Δ^{-1} can be applied on this road, and then directions can be given accordingly in $OWRN'$. Hence, the traversal $T'(s',d')$ is valid. □

By incremental construction, we shall show that any five-turn traversal from any source vertex to a destination vertex in an $OWRN$ can always be reduced to a four or fewer turn traversal without increasing the traversal length, and hence, inductively any traversal of more than five turns can be reduced to four or fewer turn traversals.

From Lemma 6 and Lemma 7, we can see that a valid k turn traversal can only be formed by adding a perpendicular road at the end (or beginning) of a valid $(k-1)$-turn traversal. Note that, if we add a perpendicular road at the end (or the beginning) of two struct-alike traversals of $k-1$ turns then they will result in the same set of struct-alike k-turn traversals. Hence, we will consider only one valid traversal among all struct-alike valid traversal set, and extend only on it.

While adding a perpendicular road to a traversal we have to take care of source and destinations relative position as well since our struct-alike definition also depends on their relative position. Hence, we add a shorter and a longer road at the end (or beginning) to make sure the relative position (quadrant with respect to the source) of destination changes (if that can be changed).

Definition 18 *A shortcut is a modification in a traversal, such that either the modified traversal has a length strictly less than the original traversal or, if the length of the modified traversal is equal to the length of the original traversal then the modified traversal has fewer turns than the original traversal.*

Instead of generating all possible five-turn traversals, we use Lemma 6 to generate them incrementally. We start with zero-turn traversal and then generate a valid one-turn traversal and then proceed to the valid two-turn traversals and so on till we reach five-turn traversal.

From here on, we show the newly added road as a dotted line, the source is represented as a disk and the destination as a squared box.

- **Case 1:** The traversal with zero turns is a straight line, i.e the source and the destination are on the same road. All the zero-turn traversals are struct-alike which is trivial, hence, we take a vertical line to construct further.

- **Case 2:** The traversal with one turn can be formed by adding a perpendicular road at the end (or beginning) of the zero-turn traversal. Four different traversals of one turn can be formed, as shown in Figure 5, since increasing the length of an additional road doesn't change the quadrant of destination, and all the four are struct- alike to an L-shaped path, i.e Figure 5 (i).

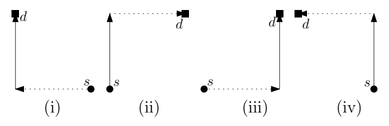

Figure 5: Struct-alike to an L-shaped path

- **Case 3:** The traversal with two turns can be formed by adding a perpendicular road at the end (or beginning) of an L-shaped path, which is a single-turn traversal. Six paths with two turns can be formed as shown in Figure 6. Note that, increasing the length of the newly added road doesn't change the relative position of destination with respect to source

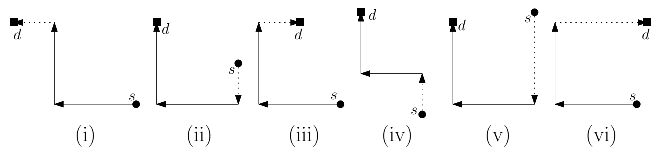

Figure 6: Two turn paths

in Figure 6 (i) and Figure 6 (iv). In Figure 6 we can see (i) and (iv) are struct-alike, (ii) and (vi) are struct-alike, and (iii) and (v) are struct-alike. Hence, we have three different possible two-turn traversals, clearly the three traversals are valid (direct the road passing through source but not in traversal away from destination, and the road passing through destination but not in traversal toward the source).

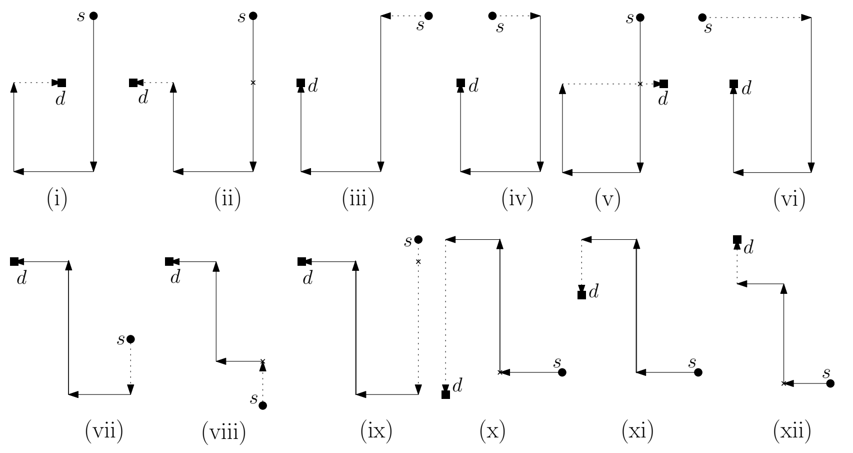

Figure 7: Possible Three turn paths

- **Case 4:** The traversal $T(s, d) = (R_1, R_2, R_3, R_4)$ with three turns can be formed by adding a perpendicular road at the end (or beginning) of the three valid traversals shown in Figure 6. Twelve different traversal of three turns are possible as shown in Figure 7. The traversals shown in Figure 7 (ii),$(viii)$, (ix), (x) and (xii) are invalid since there is a traversal $T'(s, d) = (R_1, R_4)$ with one turn and also the traversal length is at most the length of traversal shown in each case (observe Figure 8, where shortcuts are shown in blue), (v) is not a valid traversal since a vertex is being visited twice, from here on we will only extend a newly added road till it doesn't intersect with the current road. Note that, (i) and (vi) are struct-alike, (iii) and (xi) are struct-alike. Hence, we have four different valid possible traversals, Figure 7 (i), (iii), (iv) and (vii).
- **Case 5:** The traversal $T(s, d) = (R_1, R_2, \ldots, R_5)$ of four turns can be

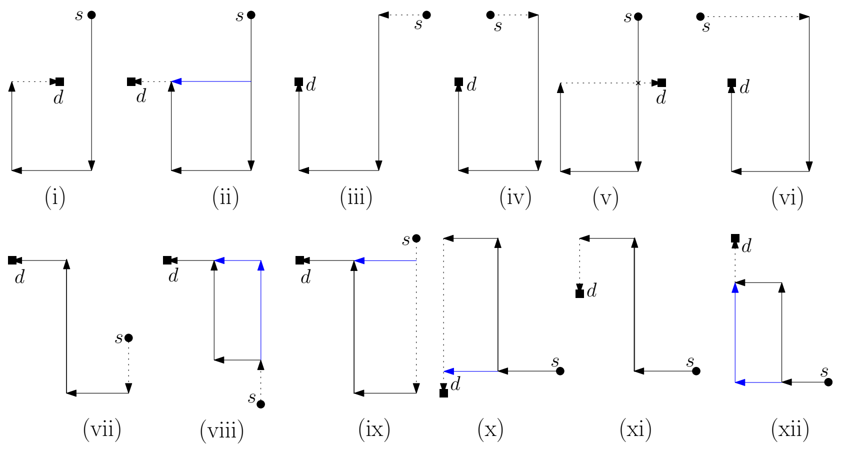

Figure 8: Three turn path modifications

formed by adding perpendicular roads at the end (or beginning) of the four valid traversals in Figure 7 (i), (iii), (iv) and (vii). The resulting possible four-turn traversals are shown in Figure 9. Observe that Figure 9 (ii) and Figure 9 (xv) are both struct-alike and valid, Figure 9 (iii) and Figure 9 (xvi) are both struct-alike and valid, while Figure 9 (x) and Figure 9 $(xiii)$ are not struct-alike to each other, but both are valid traversals. The rest of the traversals are invalid. Note that, the roads in $T(s,d) = (R_1, R_2, \ldots, R_5)$ are ordered from s to d and only shown in Figure 9 (i). We have not shown them in the rest of the figure because the figures will be congested. Let $R_s^{\perp}$ be the road passing through source and not R_1, $R_d^{\perp}$ be the road passing through destination and not R_5. In each of the remaining cases we show another traversal with fewer turns without increasing the total length.

1. For (i), (ix), (xix) the traversal $T'(s,d) = (R_1, R_4, R_5)$ has fewer turn traversals and has a shorter length than $T(s,d)$. For example, in (i) $T'(s,d) = (R_1, R_4, R_5)$ is the shortcut (shown in blue in Figure 10 (i)). A similar strategy has been applied for (ix), (xix).
2. For (iv), $(viii)$, (xiv) either one of the two traversal $T'(s,d) = (R_1, R_d^{\perp})$ or $T'(s,d) = (R_1, R_2, R_3, R_d^{\perp})$ is possible depending on the direction of $R_d^{\perp}$ and has fewer turns and has a shorter length than $T(s,d)$. For example, in (iv) if $R_d^{\perp}$ is directed downwards (Observe Figure 10 (iv), shown in red in this case) then $T'(s,d) = (R_1, R_d^{\perp})$ and shortcuts are shown in red. Otherwise $T'(s,d) = (R_1, R_2, R_3, R_d^{\perp})$ (here in Figure 10 (iv) $R_d^{\perp}$ is shown in blue, directed upward) and shortcuts are shown in blue. A similar strategy has been applied for $(viii)$, (xiv).
3. For (v), (vi), (vii), (xii), $(xvii)$ the traversal $T'(s,d) = (R_1, R_2, R_5)$

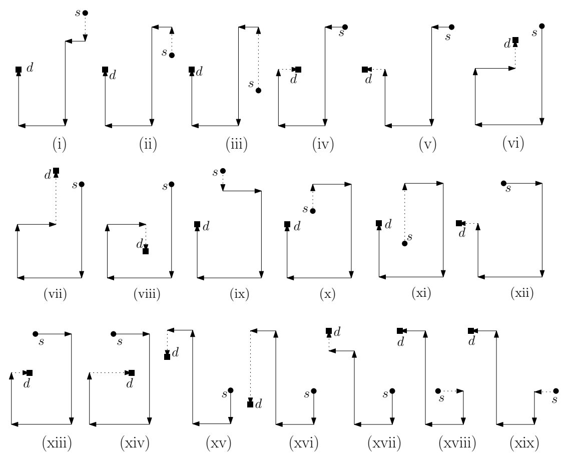

Figure 9: Possible Four turn paths

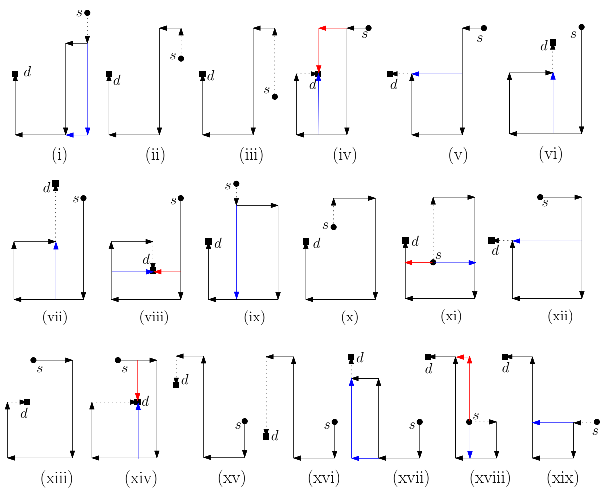

Figure 10: Four turn modifications

has fewer turns and has a shorter length than $T(s,d)$. For example, in (v) $T'(s,d) = (R_1, R_2, R_5)$ and has a shortcut (In Figure 10 (v), this is shown in blue). A similar strategy has been applied for (vi), (vii), (xii), $(xvii)$.

4. For (xi), $(xviii)$ one of the two traversal $T'(s,d) = (R_s^{\perp}, R_5)$ or $T'(s,d) = (R_s^{\perp}, R_3, R_4, R_5)$ is possible depending on the direction of $R_s^{\perp}$ and has fewer turns and has a shorter length than $T(s,d)$. For example, if in (xi) $R_s^{\perp}$ is directed leftwards (in Figure 10 (xi) this is shown in red) then $T'(s,d) = (R_s^{\perp}, R_5)$ and shortcuts are shown in red in Figure 10 (xi). Otherwise $T'(s,d) = (R_s^{\perp}, R_3, R_4, R_5)$ (in Figure 10 (xi) $R_s^{\perp}$ is shown in blue, directed rightward) and shortcuts are shown in blue in Figure 10. In $(xviii)$ if $R_s^{\perp}$ is directed upwards (in Figure 10 $(xviii)$ this is shown in red) then $T'(s,d) = (R_s^{\perp}, R_5)$ and shortcuts are shown in red in Figure 10 $(xviii)$. Otherwise $T'(s,d) = (R_s^{\perp}, R_3, R_4, R_5)$ (in Figure 10 $(xviii)$ $R_s^{\perp}$ is shown in blue, directed rightward) and shortcuts are shown in blue in Figure 10 $(xviii)$.

- **Case 6:** The traversal $T(s,d) = (R_1, R_2, \ldots, R_6)$ of five turns can be formed by adding a perpendicular roads at the end (or beginning) of the four valid traversals Figure 9 (ii), (iii), (x) and $(xiii)$. The resulting possible four turn travels are shown in Figure 11. To prove the rest are invalid traversals let $R_s^{\perp}$ be the road passing through source and not R_1, $R_d^{\perp}$ be the road passing through destination and not R_6. Each of these five-turn traversals can be modified without increasing the total length, as follows.

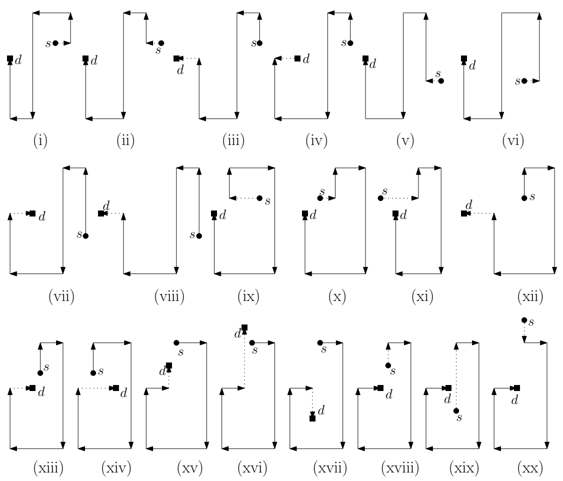

Figure 11: Possible Five turn paths

1. For (i),(vi),(ix),(xix) either $T'(s,d) = (R_s^{\perp}, R_3, R_4, R_5, R_6)$ or $T'(s,d) =$

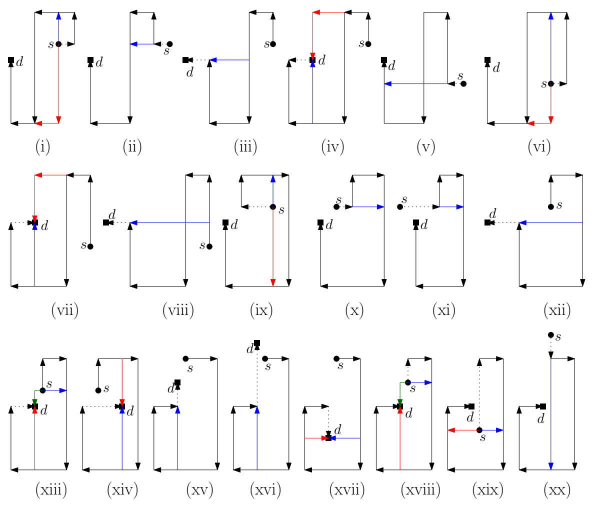

Figure 12: Five turn modifications

$(R_s^{\perp}, R_5, R_6)$ is possible depending on the direction of $R_s^{\perp}$ and has fewer turns and has a shorter length than $T(s,d)$. For example, in (i) if $R_s^{\perp}$ is directed downwards (in Figure 12 (i) this is shown in red) then $T'(s,d) = (R_s^{\perp}, R_5, R_6)$ and shortcuts are shown in red in Figure 12. Otherwise $T'(s,d) = (R_s^{\perp}, R_3, R_4, R_5, R_6)$ (in Figure 12 (i) $R_s^{\perp}$ is shown in blue color, directed upward) and shortcuts are shown in blue in Figure 12 (i). A similar strategy has been applied for (vi), (ix). In figure (xix) if $R_s^{\perp}$ is directed leftwards (in Figure 12 (xix) this is shown in red) then $T'(s,d) = (R_s^{\perp}, R_5, R_6)$ and shortcuts are shown in red in Figure 12 (xix). Otherwise $T'(s,d) = (R_s^{\perp}, R_3, R_4, R_5, R_6)$ (in Figure 12 (xix) $R_s^{\perp}$ is shown in blue color, directed rightward) and shortcuts are shown in blue in Figure 12 (xix).

2. For (ii), (x), (xi), (xx) the traversal $T'(s,d) = (R_1, R_4, R_5, R_6)$ has fewer turns and has a shorter length than $T(s,d)$. For example, in (ii) $T'(s,d) = (R_1, R_4, R_5, R_6)$ and the shortcut is shown in blue in Figure 12 (ii). A similar strategy has been applied for (x), (xi), (xx).

3. For (iii), (xii), (xv), (xvi) the traversal $T'(s,d) = (R_1, R_2, R_3, R_6)$ has fewer turns and has a shorter length than $T(s,d)$. For example, in (iii) $T'(s,d) = (R_1, R_2, R_3, R_6)$ and the shortcut is shown in blue in Figure 12 (iii). A similar strategy has been applied for (xii), (xv), (xvi).

4. For (iv), (vii), (xiv), $(xvii)$ either $T'(s,d) = (R_1, R_2, R_d^{\perp})$ or $T'(s,d) = (R_1, R_2, , R_3, R_4, R_d^{\perp})$ is possible depending on the direction of $R_d^{\perp}$ and has fewer turns and has a shorter length than $T(s,d)$. For example, in figure (iv) if $R_d^{\perp}$ is directed downwards (in Figure 12 (iv) this is shown in red) then $T'(s,d) = (R_1, R_2, R_d^{\perp})$ and shortcuts are shown in red in

Figure 12 (iv). Otherwise $T'(s,d) = (R_1, R_2, , R_3, R_4, R_d^{\perp})$ (in Figure 12 (iv) $R_d^{\perp}$ is shown in blue color, directed upward) and shortcuts are shown in blue in Figure 12 (iv). A similar strategy has been applied for (vii), (xiv). In $(xvii)$ if $R_d^{\perp}$ is directed rightwards (Figure 12 $(xvii)$ this is shown in red color) then $T'(s,d) = (R_1, R_2, , R_3, R_4, R_d^{\perp})$ and shortcuts are shown in red in Figure 12 $(xvii)$. Otherwise $T'(s,d) = (R_1, R_2, R_d^{\perp})$ (Figure 12 $(xvii)$ $R_d^{\perp}$ is shown in blue, directed leftward) and shortcuts are shown in blue in Figure 12 $(xvii)$.

5. For (v), $(viii)$ the traversal $T'(s,d) = (R_1, R_6)$ has fewer turns and has a shorter length than $T(s,d)$. For example, in (v) $T'(s,d) = (R_1, R_6)$ is the shortcut, which is shown in blue in Figure 12 (v). A similar strategy has been applied for $(viii)$.

6. For $(xiii)$, $(xviii)$ either one of the four traversals $T'(s,d) = (R_s^{\perp}, R_d^{\perp})$, $T'(s,d) = (R_s^{\perp}, R_3, R_4, R_d^{\perp})$, $T'(s,d) = (R_s^{\perp}, R_3, R_4, R_5, R_6)$, $T'(s,d) = (R_1, R_2, R_3, R_4, R_d^{\perp})$ is possible depending on the direction of $R_d^{\perp}$,$R_s^{\perp}$ and has fewer turns and has a shorter length than $T(s,d)$. For example, in $(xiii)$ if $R_d^{\perp}$ is directed downwards and $R_s^{\perp}$ directed leftwards (Figure 12 $(xiii)$ this is shown in green) then $T'(s,d) = (R_s^{\perp}, R_d^{\perp})$. If $R_d^{\perp}$ is directed upwards and $R_s^{\perp}$ is directed rightwards (in Figure 12 $(xiii)$ they are shown in red and blue respectively) then $T'(s,d) = (R_s^{\perp}, R_3, R_4, R_d^{\perp})$ and shortcuts are shown with red and blue in Figure 12 $(xiii)$. If $R_d^{\perp}$ is directed downwards and $R_s^{\perp}$ is directed rightwards (Figure 12 $(xiii)$ they are shown with green and blue respectively) then $T'(s,d) = (R_s^{\perp}, R_3, R_4, R_5, R_6)$ and shortcuts are shown in blue in Figure 12 $(xiii)$. Otherwise $T'(s,d) = (R_1, R_2, R_3, R_4, R_d^{\perp})$ (in Figure 12 $(xiii)$ $R_d^{\perp}$ is shown in red, directed upward) and shortcuts are shown in red in Figure 12 $(xiii)$. A similar strategy has been applied for $(xviii)$.

Hence, we can see that no five-turn traversal is valid and hence, from Lemma 6, we claim the following theorem.

Theorem 3 *Any traversal of more than five turns can always be reduced to a four (or less) turn traversal, without increasing the length of the traversal.*

And thus, we arrive at the following corollary.

Corollary 2 *There exists a shortest path between any pair of vertices in a strongly-connected OWRN which requires at most 4 turns or uses at most 5 roads to reach from source to destination.*

Proof: Any path can be modified as a traversal by our definition, suppose we have a shortest path between a pair of vertices then we can write it as a traversal, and from Theorem 3, we can modify this traversal to use at most 5 roads without increasing the length, now from the modified traversal we can get a new path, whose length is at most the shortest path (exactly equal since it is the shortest path). □

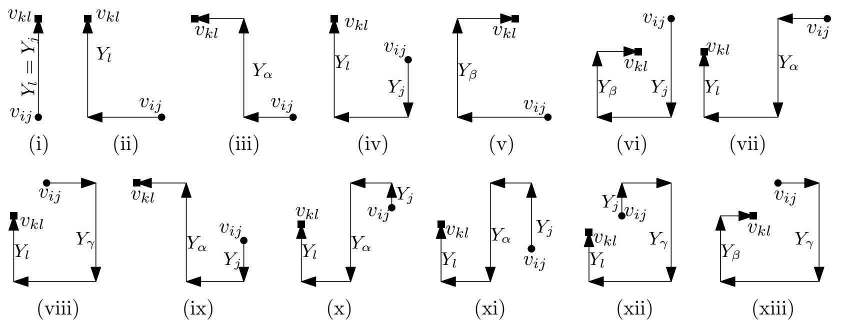

Figure 13: Valid shortest path configurations

Let v_{ij}(source) and v_{kl}(destination) be a pair of vertices for which we have to find the shortest path. Without loss of generality, let us assume v_{ij} is on the left side of v_{kl} (since, if v_{ij} is on the right we can rotate the graph by 180^o). In the following property, we talk about vertically oriented roads. However, the same can be argued in the case of horizontal roads.

Observe the valid configurations listed in Figure 13 and let Y_j and Y_l be the vertical roads passing through v_{ij} (source represented as disk) and v_{kl} (destination represented with squared box), respectively. We have already proved that any valid configuration will be struct-alike to one of the above mentioned configurations. We will now characterize all the valid configurations shown in Figure 13 as follows.

Property 5 *For any v_{ij},v_{kl}, in an OWRN there exists a shortest path that has at most three types of vertical roads, apart from Y_j, Y_l, which are defined as follows:*

(i) One road in between Y_j, Y_l, say Y_α.

(ii) One road on the right of both Y_j, Y_l, say Y_γ .

(iii) One road on the left of both Y_j, Y_l, say Y_β.

For example, the road Y_α is used in Figure 13 (*iii*), (*vii*), (*ix*), (*xi*) and (*x*). The road Y_β is used in Figure 13 (*v*), (*vi*) and (*xiii*). The road Y_γ is used in Figure 13 (*viii*), (*xii*) and (*xiii*).

Lemma 8 *If any of Y_α, Y_β or Y_γ is in the shortest path, it has the following properties:*

(i) Y_β is the closest left neighbour of Y_j of opposite direction.

(ii) Y_γ is the closest right neighbour of Y_l of opposite direction.

(iii) Y_α has the direction opposite to that of the directions of both Y_j and Y_l.

Proof: We prove the above statements one by one.

(i) Let us assume the shortest path requires us to travel through a left vertical road of Y_j i.e Y_β. Since v_{kl} is on the right of v_{ij}, the path has to cross Y_j again at some other vertex u to reach v_{kl}. Suppose Y_j and Y_β have the same direction, then we can directly reach u from v_{ij} and then from u to v_{kl} which instead would be the shortest path, contradictory to our assumption. Hence, Y_β and Y_j have opposite directions. Suppose there exists Y'_β which lies between Y_j and Y_β, and has the same direction as Y_β, then instead of going to Y_β and turning, we can directly take the turn at Y'_β which would be shortest path. However, this is a contradiction. Hence, Y_β is the closest left neighbour to Y_j.

(ii) Can be proved using similar arguments.

(iii) Suppose Y_α is in the same direction as that of Y_j then the shortest path should travel till Y_α, then take a turn and travel on road Y_α, again take a turn at some other vertex u on the road Y_α, we can travel in Y_j till we reach the vertex which lies in the same horizontal road as u, then take turn and travel to reach u, then from u to v_{kl}. Similarly, for the other case, where Y_l and Y_α have the same directions.

□

Similarly, we can prove that the above lemma is also true for horizontal direction as well. Now, we have all the pieces together to design an algorithm with a suitable data structure to find the shortest path in an $OWRN$.

Definition 19 *For a given road, the nearest road to it with the opposite direction is called the nearest reverse road.*

We suggest a simple data structure which stores the information of the nearest reverse road on both sides for each road. The following algorithm suggests an approach to construct the data structure in linear time and space in terms of the number of roads.

We use two variables *up* and *dn* to store the previous road with up direction and down direction respectively. We initialize *up* and *dn* with -1, we start scanning each road Y_i in W_x, in increasing order (i.e, $i = 1$ to $i = m$). If the direction of Y_i is *up* then set $pred(i) = dn$ and $up = i$, otherwise set $pred(i) = up$ and $dn = i$. Note that, if $pred(i) = -1$ implies there is no road which is of opposite direction to Y_i, otherwise if $pred(i) = j$ implies Y_j precedes Y_i and are of opposite directions. Similarly, we can do a reverse scan from $i = m$ to $i = 1$ and store the successor information in $succ(i)$. A similar approach can be used for the roads in W_y.

Theorem 4 *The Shortest path in an $OWRN$ can be computed in $\mathcal{O}(p)$ time and the length can be computed in $\mathcal{O}(1)$ time, where p is the number of vertices in the shortest path, with the help of a pre-computed data structure of $\mathcal{O}(|W_x| + |W_y|)$ time, and $\mathcal{O}(|W_x| + |W_y|)$ space.*

Proof: Given any two vertices v_{ij} and v_{kl}, we need to find the shortest path between these two vertices. From Lemma 5 and Lemma 8, we know that there exists a shortest path that uses roads $W'_y = \{Y_\gamma, Y_j, Y_\alpha, Y_l, Y_\beta\}$ among the vertical set of roads. Similarly, there exists a shortest path that uses roads $W'_x = \{X_\omega, X_i, X_\eta, X_k, X_\theta\}$ among the set of horizontal roads.

If we have already computed the nearest neighbours using the algorithm Nearest Reverse Road, then looking at $X(i)$, $X(k)$, $Y(j)$ and $Y(l)$ we can compute the sets W'_x and W'_y in constant time. Now generate an $OWRN' = (W'_x, W'_y)$ and compute the shortest path from the respective vertex of v_{ij} to the respective vertex of v_{kl} in $OWRN'$ using any shortest path algorithm.

To compute the shortest path in $OWRN$, start from v_{ij} and travel along the same set of roads used in the shortest path found in $OWRN'$. Since the size of the $OWRN'$ is of $\mathcal{O}(1)$, computing shortest path length takes $\mathcal{O}(1)$ time. It takes $\mathcal{O}(1)$ time to determine the vertices in the shortest path, since we are only travelling along the vertices which are in the shortest path. Hence, it takes $\mathcal{O}(p)$ time to report the shortest path with p vertices.

The space required to store the information of nearest reverse roads is $\mathcal{O}(|W_x| + |W_y|)$, since for each road we store at most two other roads. The time complexity is also $\mathcal{O}(|W_x| + |W_y|)$ because each road is visited twice. □

5 Conclusions and open problems

We studied the problem of collision-free traffic configuration TC in a directed grid graph GG. We proved that finding a maximum cardinality subset $C_{max} \subseteq C$, where C is a set of vehicles, such that $TC = (GG, C_{max})$ is collision-free, is NP-Hard. We have shown all the possible configurations of the path, that connect two vertices in an $OWRN$ and designed an efficient data structure for dynamic maintenance of shortest path. We show that GG can be preprocessed into a data-structure in $\mathcal{O}(n + m)$ time and space, such that the length of the shortest path between any pair of vertices in GG can be computed in $\mathcal{O}(1)$ time and shortest path can be computed in $\mathcal{O}(p)$ time, where p is the number of vertices in the path. In the future, we will extend this work to compute various kinds of facility location problems on an $OWRN$. It will be interesting to investigate the time complexity of one-centre or k-centre problems in an $OWRN$. A k-Centre in an $OWRN$ is the problem of finding k nodes (centres) in the graph such that the maximum distance from any node to the closest such centre is minimized. Another interesting problem is, designing an efficient algorithm to find the shortest path in an $OWRN$ without any data structure.

References

[1] S. Arora, A. Raina, and A. Mittal. Collision avoidance among agvs at junctions. In *Intelligent Vehicles Symposium, 2000. IV 2000. Proceedings of the IEEE*, pages 585–589. IEEE, 2000. doi:10.1109/IVS.2000.898411.

[2] P. Dasler and D. M. Mount. On the complexity of an unregulated traffic crossing. In *Algorithms and Data Structures - 14th International Symposium, WADS*, volume 9214 of *Lecture Notes in Computer Science*, pages 224–235. Springer, 2015. `doi:10.1007/978-3-319-21840-3\_19`.

[3] J. Håstad. Clique is hard to approximate within $n^{1-\varepsilon}$. *Acta Mathematica*, 182(1):105–142, 1999. `doi:10.1007/BF02392825`.

[4] P. Hyla and J. Szpytko. Automated guided vehicles: the survey. *Journal of KONES*, 24, 2017. `doi:10.5604/01.3001.0010.3055`.

[5] M. Kakikura, J. Takeno, and M. Mukaidono. A tour optimization problem in a road network with one-way paths. *Electrical Engineering in Japan*, 98(4):141–147, 1978. `doi:10.11526/ieejeiss1972.98.257`.

[6] C. W. Kim and J. M. Tanchoco. Conflict-free shortest-time bidirectional agv routeing. *The International Journal of Production Research*, 29(12):2377–2391, 1991. `doi:10.1080/00207549108948090`.

[7] G. A. Koff. Automatic guided vehicle systems: applications, controls and planning. *Material flow*, 4(1-2):3–16, 1987.

[8] C. J. Malmborg. A model for the design of zone control automated guided vehicle systems. *the International Journal of Production Research*, 28(10):1741–1758, 1990. `doi:10.1080/00207549008942830`.

[9] L. Qiu, W.-J. Hsu, S.-Y. Huang, and H. Wang. Scheduling and routing algorithms for agvs: a survey. *International Journal of Production Research*, 40(3):745–760, 2002. `doi:10.1080/00207540110091712`.

[10] H. E. Robbins. A theorem on graphs, with an application to a problem of traffic control. *The American Mathematical Monthly*, 46(5):281–283, 1939. `doi:10.2307/2303897`.

[11] I. F. Vis. Survey of research in the design and control of automated guided vehicle systems. *European Journal of Operational Research*, 170(3):677–709, 2006. `doi:10.1016/j.ejor.2004.09.020`.

[12] X. Yan, C. Zhang, and M. Qi. Multi-agvs collision-avoidance and deadlock-control for item-to-human automated warehouse. In *Industrial Engineering, Management Science and Application (ICIMSA), 2017 International Conference on*, pages 1–5. IEEE, 2017. `doi:10.1109/ICIMSA.2017.7985596`.

[13] L. Zeng, H.-P. Wang, and S. Jin. Conflict detection of automated guided vehicles: a petri net approach. *The International Journal of Production Research*, 29(5):866–879, 1991. `doi:10.1080/00207549108930107`.