Collision-free Routing Problem with Restricted L-path.

DISSERTATION SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology
in
Computer Science

by

Jammigumpula Ajay Kumar
[Roll No: CS-1606]

Under the Guidance of

Dr. Sasanka Roy
Associate Professor
Advanced Computing and Microelectronics Unit

Indian Statistical Institute
Kolkata-700108, India

May 2018

Dedicated to my family and my supervisor

Declaration

I hereby declare that the dissertation report entitled **"Collision-free Routing Problem with Restricted L-path"** submitted to Indian Statistical Institute, Kolkata, is a bonafide record of work carried out in partial fulfilment for the award of the degree of **Master of Technology in Computer Science**. The work has been carried out under the guidance of **Dr. Sasanka Roy**, Associate Professor, ACMU, Indian Statistical Institute, Kolkata.

I further declare that this work is original, composed by myself. The work contained herein is my own except where stated otherwise by reference or acknowledgement, and that this work has not been submitted to any other institution for award of any other degree or professional qualification.

Place : Kolkata

Date : May 26, 2018

Jammigumpula Ajay Kumar

Roll No: CS-1606

Indian Statistical Institute

Kolkata - 700108 , India.

CERTIFICATE

This is to certify that the dissertation entitled **"Collision-free Routing Problem with Restricted L-path"** submitted by **Jammigumpula Ajay Kumar** to Indian Statistical Institute, Kolkata, in partial fulfilment for the award of the degree of **Master of Technology in Computer Science** is a bonafide record of work carried out by him under my supervision and guidance. The dissertation has fulfilled all the requirements as per the regulations of this institute and, in my opinion, has reached the standard needed for submission.

Sasanka Roy

Associate Professor,

Advanced Computing and Microelectronics Unit,

Indian Statistical Institute,

Kolkata-700108, INDIA.

Acknowledgments

In the accomplishment of this project successfully, many people have bestowed upon me their blessings and their heart pledged support, this time I am utilizing to thank all the people who have been concerned with this project.

I would like to show my highest gratitude to my advisor, *Dr. Sasanka Roy*, Advanced Computing and Microelectronics Unit, Indian Statistical Institute, Kolkata, for his guidance and continuous support and encouragement. He has taught me how to do good research, and motivated me with great insights and innovative ideas.

I would also like to thank *Joydeep Mukherjee*, for his valuable suggestions and discussions.

My deepest thanks to all the teachers of Indian Statistical Institute, for their valuable suggestions which added an important dimension to my research work.

Finally, I am very thankful to my parents and family for their everlasting support.

Last but not the least, I would like to thank all of my friends for their help. I also would like to thank all those, whom I have missed out from the above list.

Jammigumpula Ajay Kumar

Indian Statistical Institute

Kolkata - 700108 , India.

Publication based on this Dissertation

1. Jammigumpula Ajay and Sasanka Roy. Collision-free Routing Problem with Restricted L-path. Accepted In *Proceedings of 29th International Workshop on Combinatorial Algorithms (IWOCA)*, 2018.

Abstract

We consider a variant of collision-free routing problem CRP. In this problem, we are given set C of n vehicles which are moving in a plane along a predefined directed rectilinear path. Our objective (CRP) is to find the maximum number of vehicles that can move without collision. CRP is shown to be NP-Hard by Ajaykumar et al. [1]. It was also shown that the approximation of this problem is as hard as Maximum Independent Set problem (MIS) even if the paths between a pair of vehicles intersects at most once. We study the constrained version $CCRP$ of CRP in which each vehicle c_i is allowed to move in a directed L-Shaped Path.

We prove $CCRP$ is NP-Hard by a reduction from MIS in L-graphs, which was proved to be NP-Hard even for unit L-graph by Lahiri, Mukherjee, and Subramanian [2]. Simultaneously, we show that any $CCRP$ can be partitioned into collection $\mathcal{L}$ of L-graphs such that $CCRP$ reduces to a problem of finding MIS in L-graph for each partition in $\mathcal{L}$. Thus we show that any algorithm, that can produce a β-approximation for L-graph, would produce a β-approximation for $CCRP$. We show that unit L-graphs intersected by an axis-parallel line is Co-comparable. For this problem, we propose an algorithm for finding MIS that runs in $O(n^2)$ time and uses $O(n)$ space. As a corollary, we get a 2-approximation algorithm for finding MIS of unit L-graph that runs in $O(n^2)$ time and uses $O(n)$ space.

Keywords: *Maximum Independent Set, L-Graphs, Approximation Algorithm, Collision-free, Co-comparable Graph.*

Contents

1 Introduction **1**

2 Related Work **3**

2.1 Decision Problem and Reduction 3

2.2 Maximum Independent Set 4

2.3 Traffic Crossing Problem 4

2.4 Traffic Configuration Problem 5

2.5 Intersection of Paths on a Grid 6

2.6 Comparable and Co-comparable Graphs 7

2.7 Our Contribution 8

3 Our Work **9**

3.1 Definitions and Notations 9

3.2 Hardness of CCRP 10

3.3 Approximation for MIS of G_L^t 18

3.4 Unit L Graph Approximation 20

4 Conclusion And Future Work **25**

List of Figures

3.1 Illustration of Lemma 3.1 11

3.2 Illustration of Case 1 13

3.3 Modifications for Case 1 14

3.4 Illustration of Case 2.1 15

3.5 Modification for Case 2.1 15

3.6 Illustration of Case 2.2 16

3.7 Modification for Case 2.2 17

3.8 G_{LU} with four cycle 21

List of Algorithms

1 Assignment of Time in G_L to obtain G_L^t 18

2 Procedure to partition G_L^t 19

3 Computing $R(k)$ for each index in S_i 23

Chapter 1

Introduction

The problem is motivated by the recent development of automated driver-less vehicles, which are capable of various decision activities such as motion-controlling, path planning. If we consider a simple road network like Manhattan (grid network) and restrict it to be one way for the simplicity of driver-less vehicles routes, many interesting problems can be seen in this network.

The paper on Problems on One Way Road Networks [1], gives an idea of One Way Road Network($OWRN$) and Traffic Configuration(TC), where each vehicle moves in a predetermined path in an $OWRN$ and the aim is to find the maximum number of vehicles that can be allowed to move without having any collision for a given TC. They proved that this problem is NP-hard by reducing it to MIS, and also showed that the approximation for this problem is as hard as approximating MIS. It is known that, for every fixed $\epsilon > 0$, MIS cannot be approximated within a multiplicative factor of $n^{1-\epsilon}$ for a general graph, unless $NP = ZPP$ [5].

We can generalize TC to CRP, where each vehicle is allowed to move in a rectilinear path, replacing the vertices of the $OWRN$ by their coordinate points and the path same as in TC. Similar kind of road network has also been studied by Dasler and Mount [3].

If we constrain the vehicles to move in directed straight lines parallel to the axis, then the corresponding graph to CRP will be a Bipartite Graph. MIS of a Bipartite Graph can be computed using Kőnig's Theorem[18] and Network-Flow Algorithm [19] in polynomial-time.

Asinowski et.al [4], discuss the class of vertex intersection graphs of paths on a grid (VPG), they consider a special subclass where each path is of at most k bends. This subclass is denoted as B_k-VPG graphs, $k \geq 0$. If k is unrestricted then VPG is equivalent to a class of string graphs.

The maximum independent set problem (MIS) of B_1-VPG graphs is studied by Lahiri, Mukherjee, and Subramanian [2], they gave a $\mathcal{O}(\log^2 n)$ approximation for B_1-VPG graphs.

Together CRP and B_k-VPG has motivated us to study the union of both i.e, A CRP where each vehicles is allowed to move in a k bend path on a grid. We consider a subclass of this problem where $k = 1$ and prove its hardness and give a couple of approximation algorithms for this problem and its restricted versions.

Though the problem is very restricted it has few applications in aeroplane scheduling on a runway, automated driving vehicles, and chemical flows in a bio-chip etc.

Chapter 2

Related Work

2.1 Decision Problem and Reduction

Michael and David [10] discussed in details about reduction, decision problems and NP-hardness etc.

A problem is said to be a decision problem if its output is a single boolean value: YES or NO. **P** denotes the set of decision problems that can be solved in polynomial time. **NP** denotes the set of decision problems where we can verify a YES answer in polynomial time if we have the solution.

A problem Π_1 is said to be reducible to another problem Π_2 if there exists a polynomial time algorithm to convert any given instance of Π_1 into an instance of Π_2. Hence if Π_1 is reducible to Π_2, then a solution to Π_2 can be used to solve Π_1.

A problem Π is said to be **NP-hard** if every problem in *NP* can polynomial time reducible to Π. Alternatively if a polynomial time algorithm for Π imply a polynomial time algorithm for all problem in NP, then Π is said to be *NP-hard*. A problem is said to be **NP-complete** if it is both *NP-hard* and *NP*.

To prove that problem Π_1 is *NP-hard*, reduce a known *NP-hard* problem to Π_1. i.e, If there exists a polynomial time algorithm such that given any instance of a known *NP-hard* problem Π_2, the algorithm produces an instance of another problem Π_1, then Π_1 also belong to *NP-hard*.

2.2 Maximum Independent Set

Håstad [5] in his work has discussed in details about the hardness of the clique problem, maximum independent set, and the hardness of approximation.

An independent set in a graph $G = (V, E)$ is defined as a subset of vertices $S \subseteq V$ in G such that no two vertices in S have an edge in E.

Given an undirected graph $G = (V, E)$, the maximum independent set problem (MIS) is to find an independent set in G with maximum cardinality. MIS is a well know problem and is proven to be NP-Hard, and its decision version is to find if there exists an independent set of size k in G. The decision version is known to be NP-Complete.

Its is natural to try to give an approximation algorithm for NP-Hard problems, but the theorem by Håstad proved that MIS is extremely hard to approximate.

The Håstad theorem says the following: there are a class of graphs in which the maximum independent set size is either less than n^{δ} or greater than $n^{1-\delta}$ and it is *NP-Complete* to decide whether a given graph falls into the former category or the latter.

Chordal graphs, perfect graphs, comparable graphs, and co-comparable graphs are few special classes of graph for which MIS can be found in polynomial time.

2.3 Traffic Crossing Problem

The automated vehicles moving through an intersection are bound to have collisions if the motion of vehicles are not coordinated. The traffic crossing problem is how to coordinate the motions of a given set of vehicles in a given network with intersections. The decision version for this problem is, given a traffic crossing C, is there any valid set of speed assignments for C.

The work by Dasler and Mount [3], focuses on the control of a vehicle over a span of interval(seconds to minutes). The traffic network they have considered is a collection of axis-parallel lines, which represent roads. Each vehicles is represented by a line segments. The vehicles are allowed to move monotonically along the roads(axis parallel lines in the plane), they are allowed to change their speeds at any instant, provided it doesn't exceed the speed limit. No vehicle is allowed to make a turn, reverse the direction, or change lanes. The objective is to find speed profiling for these vehicles which are moving from source to destination(No two vehicles have same source and destination), without any collision.

They reduced 3-SAT to traffic crossing problem and thus proved traffic crossing problem is NP-Hard.

A one-sided traffic crossing problem is a restricted version of the traffic crossing problem, in which vehicles moving in one direction have a fixed speed, and the vehicles moving in the other direction will have to adjust their speeds to avoid the collisions. The objective is to find a valid speed profile for the vehicles moving in the direction where their speed should be adjusted.

The One-Sided Traffic Crossing Problem can be solved in $\mathcal{O}(n \log n)$ time. The algorithm involves two applications of plane sweep.

2.4 Traffic Configuration Problem

A One way road network is defined to be a set of axis parallel roads forming a grid network, and each road has a specific direction (each road is a one way).

Given a set of vehicles, each moving in a predefined path with a unit velocity in a one way road network. When two vehicles reach a junction at same time orthogonally they will collide. The traffic configuration problem is to find the maximum subset of vehicles that can be allowed to move such that no collision occurs. The decision version is to find if there exists a subset of vehicles that doesn't have a collision with cardinality k. This problem is also called collision-free routing problem.

The traffic configuration problem is proven to be NP-hard by Ajaykumar et.al [1] , even when the path of no two vehicles over lap more than once. They achieved it by reducing the MIS for general graph to traffic Configuration problem, by using a gadget called delay which modifies the path to avoid/create a collision. This reduction is gap preserving and hence it is as hard as MIS for general graph to approximate.

2.5 Intersection of Paths on a Grid

Asinowski et.al [4] in their work presented the following ideas.

A vertex intersection graphs of paths on a grid (VPG) is a graph with the set of vertices representing the paths and set of edges representing the intersection of the respective path, also note that no two paths have an overlap and the intersection(s) is(are) the common point(s) where segments of the two paths are orthogonal and have a point in common.

When each path in the representation has at most $k \geq 0$ bends this subclass is named as B_k-VPG is defined, if k is unbounded then MIS for B_k-VPG is NP-hard and even hard to approximate.

Lahiri, Mukherjee and Subramanian [2] has proven MIS of unit length equilateral B_1-VPG is NP-hard. i.e, when each path has at most 1 bend and the length of both the segments are unit for all paths. They also proposed a $\mathcal{O}(\log^2 n)$ approximation algorithm for MIS of B_1-VPG.

2.6 Comparable and Co-comparable Graphs

A directed graph $G = (V, E)$ is called transitive oriented graph, if there exists directed edges $(u, v) \in E$ and $(v, w) \in E$, then for all such $u, v, w \in V$ there exists a directed edge $(u, w) \in E$.

An undirected graph is called a comparability graph if it has a transitive orientation, i.e, an assignment of directions to the edges such that the resultant graph is transitive oriented graph.

Alternatively, a simple undirected graph is called the comparability graph of the poset P if the vertices of G are the elements P, and two vertices are adjacent if and only if the corresponding elements of P are comparable.

A co-comparability graph is an undirected graph that connects pairs of elements that are incomparable to each other in a partial order. The co-comparability graphs and comparability graphs are complements to each other.

Mirsky's theorem [14] proves that every comparability graph is a perfect graph, and Dilworth's theorem [13] proves that complement of every comparability graph(co-comparable graph) is a perfect graph. i.e, Both comparability graphs and co-comparability graphs are perfect graphs.

Golumbic, Rotem and Urrutia [8] proved that Interval graphs are chordal graphs and their graph complements are comparability graphs.

Because comparability graphs are perfect, many problems that are hard on more general classes of graphs, including graph coloring and the independent set problem, can be computed in polynomial time for comparability graphs. Same goes for the co-comparability graphs.

2.7 Our Contribution

We considered a special case of CRP, called constrained collision-free routing problem $CCRP$, where each vehicle is restricted to move in an L-shaped path. We prove $CCRP$ is NP-hard by reduction from MIS in L-graphs.

Simultaneously, we show that any $CCRP$ can be partitioned into a collection $\mathcal{L}$ of L-graphs such that $CCRP$ reduces to a problem of finding MIS for each partition in $\mathcal{L}$. Thus we show that any algorithm, that can produce a β-approximation for L-graph, would produce a β-approximation for $CCRP$. Since the best-known algorithm for L-graph by Lahiri, Mukherjee, and Subramanian [2] has $O(\log^2 n)$-approximation, $CCRP$ has $O(\log^2 n)$-approximation.

Further, we extended our work to study the properties of unit L-graph*, denoted as G_{LU}, where all the objects are of unit size. We prove that unit L-graph, denoted as $G_{LU}(\ell)$, where all L's are intersected by a single axis parallel line ℓ is a Co-comparable graph. This characterization gives us an algorithm for finding MIS in $O(n^2)$ time using $O(n^2)$ space using results by Rose, Tarjan and Lueker [20]. We propose a dynamic programming based algorithm for finding MIS of $G_{LU}(\ell)$ that runs in $O(n^2)$ time and uses $O(n)$ space. Also as a corollary, we get a 2-approximation for finding MIS of G_{LU}.

*Both the horizontal and vertical segments of an L are of unit length

Chapter 3

Our Work

3.1 Definitions and Notations

Following are the few definitions we will be using throughout this work and our main problem statement.

Definition 3.1. *An L-shaped path $P_i = (p_i, q_i, r_i)$ is defined by three co-ordinate points, where the path segment p_iq_i of P_i forms a vertical segment (directed downwards) and path segment q_ir_i of P_i forms a horizontal segment (directed rightwards).*

Definition 3.2. *A vehicle c_i is defined as a 3-tuple (t_i, s_i, P_i), where t_i is the start time, s_i is a constant speed with which it will travel till it reaches the destination, P_i is the L-shaped path (with source p_i and destination r_i).*

Definition 3.3. *If two L-shaped paths have a common point, then they are said to be intersecting with each other. This common point is called the intersection point of the two vehicles moving in these L-shaped paths.*

Definition 3.4. *If two vehicles reach an intersection point orthogonally at the same time, then we call it a collision.*

Problem 3.1 (CCRP)**.** *Given a set C of vehicles moving in an L-shaped path on a plane, find the maximum subset $C_{max} \subseteq C$ such that no two vehicles in C_{max} has a collision.*

Problem 3.2 (B_1-CRP)**.** *Given a set C of vehicles moving in a single bend path on a grid network*, find the maximum subset $C_{max} \subseteq C$ such that no two vehicles in C_{max} has a collision.*

Clearly $CCRP$ is a subclass of B_1-CRP where each path is of L shape. Hence as a corollary of $CCRP$ we also prove the hardness and give approximation for B_1-CRP.

3.2 Hardness of CCRP

In this section we prove the hardness of $CCRP$. Throughout this work, we assume that each vehicle is moving with a unit velocity, and the paths intersect at a single point.

Definition 3.5. *We define $x(p)$, $y(p)$ as the X-coordinate and Y-coordinate of the point p.*

Observation 3.1. *If two paths P_i and P_j intersect with each other such that $x(q_i) < x(q_j)$, then $y(q_i) > y(q_j)$.*

Lemma 3.1. *If two vehicles collide with each other, then a third vehicle whose path intersects with both paths would either (i) collide with both the vehicles or (ii) does not collide with both the vehicles.*

Proof. Consider three vehicles c_1, c_2 and c_3 with paths P_1, P_2 and P_3, respectively. Without loss of generality, we can assume $x(q_1) < x(q_2) < x(q_3)$. Thus from

*all the possible paths in B_1-VPG graphs

Observation 3.1 we can claim $y(q_1) > y(q_2) > y(q_3)$. Let P_1, P_2 intersect at point γ, P_3 intersects with both P_1 and P_2 at points α, β respectively. Let the distance from γ to α be a units, and the distance from α to β be b units, refer to Fig 3.1.

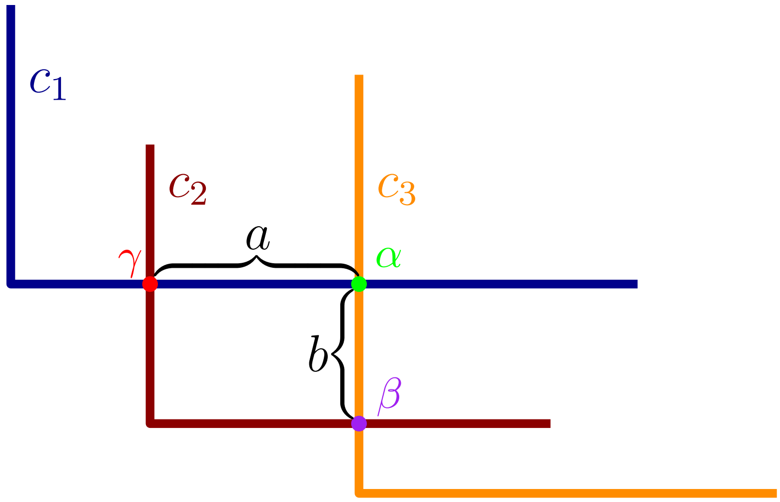

Figure 3.1: Illustration of Lemma 3.1

Let c_1 reaches point γ at time t^1_γ, then the time at which it reaches point α is $t^1_\alpha = t^1_\gamma + a$. Let c_2 reaches point γ at time t^2_γ, then the time at which it will reach point β is $t^2_\beta = t^2_\gamma + a + b$. Let c_3 reaches point α at time t^3_α, then the time at which c_3 reaches point β is $t^3_\beta = t^3_\alpha + b$.

Clearly $(t^1_\alpha - t^1_\gamma) - (t^2_\beta - t^2_\gamma) + (t^3_\beta - t^3_\alpha) = 0$, rearranging the terms we get, $(t^1_\alpha - t^3_\alpha) + (t^2_\gamma - t^1_\gamma) + (t^3_\beta - t^2_\beta) = 0$. If two vehicles collide then one of the three parts in the above equation will become zero. Thus, if one of the remaining two parts becomes zero so does the other, this concludes the proof. □

Definition 3.6. *We define an L-path graph G^t_L as a collision graph of vehicles moving in an L-shaped path, where each vehicle represents a vertex in G^t_L, and there is an edge between two vertices in G^t_L if the respective vehicles collide.*

So our *CCRP* problem reduces to the problem of finding MIS of G^t_L.We may use MIS of G^t_L and *CCRP* interchangeably. We denote $|S|$ as the cardinality of the set S. We also denote $|a - b|$ as the distance between two points a and b on a real line.

Definition 3.7. *Any induced sub-graph H^t of G_L^t is called a connected component in L-path graph, if for every vertex pair u, v in H^t there exists a path from u to v in H^t.*

Theorem 3.2. *If the path of a vehicle c_i intersect with paths of two or more vehicles in a connected component and it collides with one of them, then it collides with all the vehicles whose path it intersects.*

Proof. We prove this theorem using strong induction. As the base case, if the connected component has two vehicles and the path of a third vehicle intersects the path of both vehicles, and it collides with one of them, then from Lemma 3.1 the statement holds for the base case of three vehicles.

We assume that any connected component of size less than k follows this property, and we prove the claim holds for any connected component of size k.

Given any connected component H^t of size k, select any vehicle c_3, if its path intersects with only one vehicle (which is a collision since c_3 belongs to the connected component), then the claim is true. If the path of c_3 intersects with the path of more than one vehicle, then it must collide with at least one of the vehicles since it belongs to the connected component. So we choose one intersection and one collision to prove that the intersection will be a collision, thus inductively prove that all intersections will be collisions.

Let c_1 and c_2 be vehicles such that either c_1 or c_2 has a collision with c_3, while the other has an intersection with the path of c_3. Without loss of generality we can assume $y(q_1) > y(q_2)$.

Delete c_3 from H^t and find the path in H^t with minimum number of nodes from corresponding vertex of c_1 to respective vertex of c_2. Consider all the corresponding vehicles of the vertices in this path and remove the rest of the vehicles. If c_1 and

c_2 intersect with each other then by our inductive assumption c_1 and c_2 belong to a connected component of size less than k. Thus c_1 and c_2 collide with each other. By Lemma 3.1 c_3 collides with both c_1 and c_2.

Thus we only need to show for the case where P_1 and P_2 doesn't intersect with each other. Since it is the shortest path in H^t no vehicle's path will intersect more than two vehicles. P_1 and P_2 intersect with only one path each. Note all these vehicles together form a single connected component. If we insert c_3 it will still collide with c_1(or c_2) while its path intersect with the path of c_2 (or c_1).

Here we have following two cases, where in each case we replace c_1 with another vehicle c_1' and c_2 with another vehicle c_2'. Such that (i) P_1' and P_2' will intersect and (ii) the set of vehicles in the plane after replacing c_1 and c_2 will still be a connected component.

Case 1: $x(q_1) < x(q_2)$. Since we assumed $y(q_1) > y(q_2)$ and P_1 and P_2 doesn't intersect, we can have following three configurations as shown in Fig 3.2.

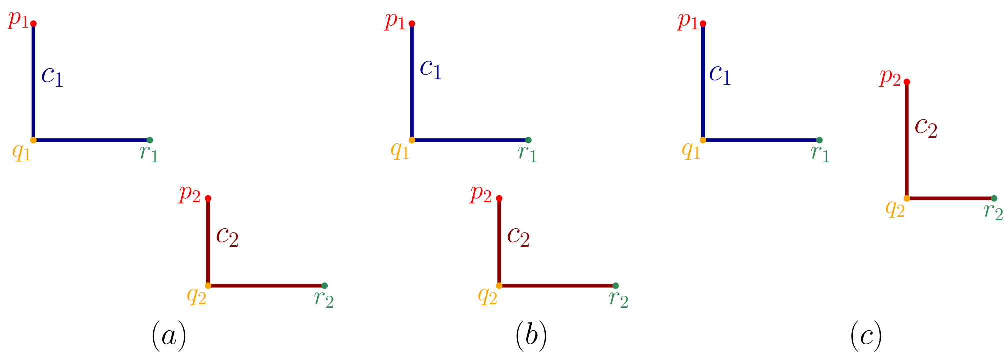

Figure 3.2: Illustration of Case 1

For configuration in Fig 3.2 (a) If we extend q_1r_1 in rightward direction and p_2q_2 in upward direction they intersect as shown in Fig 3.3.(a) where c_1' and c_2' represent this modification.

For configuration in Fig 3.2 (b) If we extend p_2q_2 in upward direction they intersect as shown in Fig 3.3.(b) where c_1' and c_2' represent this modification.

For configuration in Fig 3.2 (c) If we extend q_1r_1 in rightward direction they intersect as shown in Fig 3.3.(c) where c_1' and c_2' represent this modification.

We replace c_1 with c_1' and c_2 with c_2', such that $t_1' = t_1$, $p_1' = p_1$, $q_1' = q_1$, $y(r_1') = y(r_1)$, $x(r_1') = max(x(r_1), x(q_2) + \epsilon)$, and $r_2' = r_2$, $q_2' = q_2$,$x(p_2') = x(p_2)$, $y(p_2') = max(y(p_2), y(q_1) + \epsilon)$, $t_2' = t_2 - (y(p_2') - y(p_2))$, for some $\epsilon > 0$.

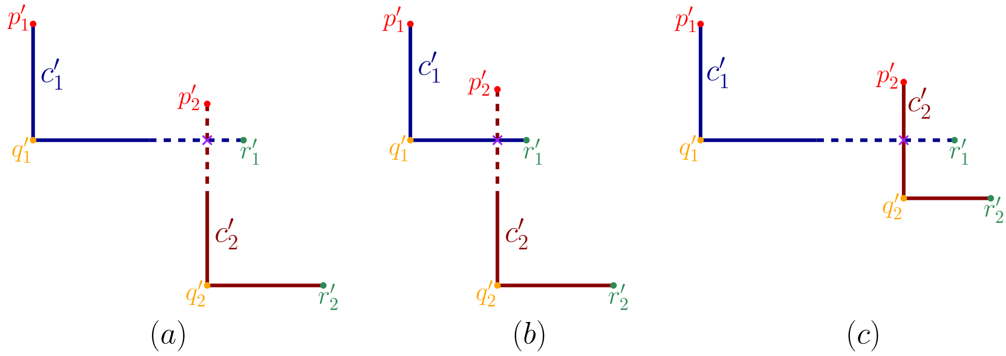

Figure 3.3: Modifications for Case 1

The above modification doesn't change the time at which c_1' (or c_2') reaches the collision point of c_1 (or c_2). Hence, after the replacement, c_1' and c_2' belongs to the same connected component. In our construction we also made sure that P_1' and P_2' intersect. Now we have a connected component of size less than k. Hence c_1' and c_2' must also collide.

Now consider c_3, if it collides with c_1(or c_2) then it must also collide with c_1'(or c_2') according to our construction. From Lemma 3.1 it is evident that it collides with both c_1' and c_2'. Hence the intersection must also be a collision.

Case 2: $x(q_1) > x(q_2)$. In the previous case we only extended one of the line segments for c_1 and c_2 to get c_1' and c_2' respectively, but in this case we are moving the segment i.e both points p_1, q_1 are moved by some distance leftwards or both q_1,r_1 are moved by some distance downwards. In order to keep the connectivity we check the immediate neighbour c_4 of c_1 and the segment say p_1q_1 (or q_1r_1) of P_1 with which the path P_4 intersects. Then modify the other segment q_1r_1 (or p_1q_1) of P_1 to get c_1'. c_2' can be generated just by extending one of the segments.

The vehicle c_4 that collides with c_1 could have $y(q_4) > y(q_1)$ or $y(q_4) < y(q_1)$.

1. If $y(q_4) < y(q_1)$ (i.e, P_4 intersects segment q_1r_1 of P_1) then we can have following three configurations as shown in Fig 3.4.

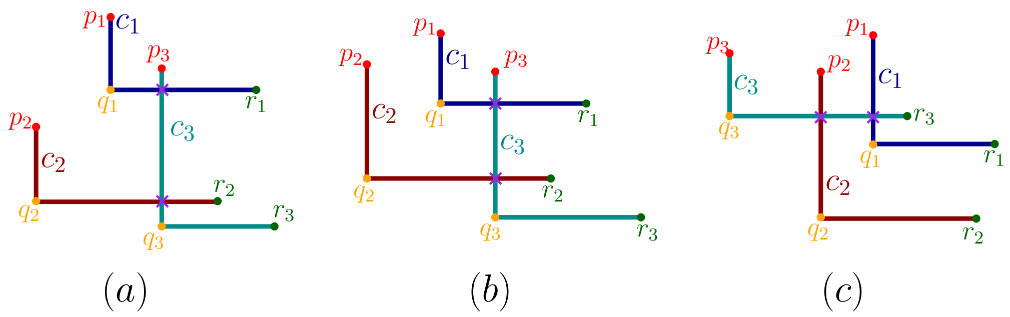

Figure 3.4: Illustration of Case 2.1

For configuration in Fig 3.4 (a) If we shift p_1, q_1 to the left direction and extend p_2q_2 in upward direction and P_3 intersects segments q_1r_1 and q_2r_2 then they intersect as shown in Fig 3.5 (a) where c_1' and c_2' represent this modification.

For configuration in Fig 3.4 (b) If we shift p_1, q_1 to the left direction and P_3 intersects segments q_1r_1 and q_2r_2 then they intersect as shown in Fig 3.5.(b) where c_1' and c_2' represent this modification.

For configuration in Fig 3.4 (c) If we shift p_1, q_1 to the left direction and P_3 intersects segments p_1q_1 and p_2q_2 then they intersect as shown in Fig 3.5.(c) where c_1' and c_2' represent this modification.

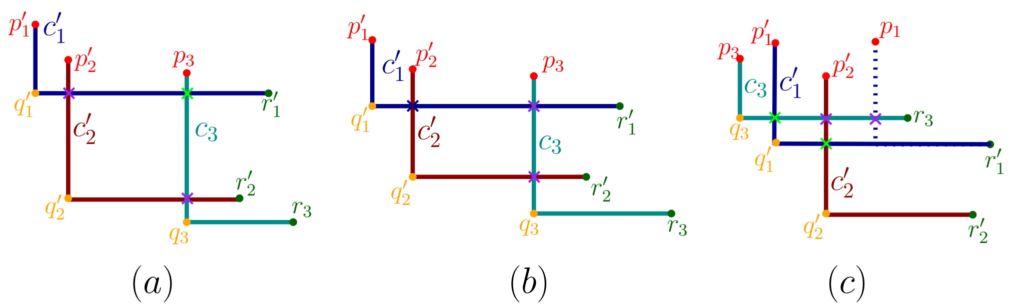

Figure 3.5: Modification for Case 2.1

We replace c_1 with c'_1, c_2 with c'_2 such that, $y(p'_1) = y(p_1)$, $x(p'_1) = x(p_2) - \epsilon$, $y(q'_1) = y(q_1)$, $x(q'_1) = x(p'_1)$, $t'_1 = t_1 - (x(q'_1) - x(q_1))$, $r'_1 = r_1$ and $y(p'_2) = max(y(p_2), y(p_1) + \epsilon)$, $r'_2 = r_2$, $q'_2 = q_2$, $t'_2 = t_2 - (y(p'_2) - y(p_2))$, for some $\epsilon > 0$.

By our construction P'_1 and P'_2 intersect with each other, c'_1 collides with c_4, and c'_2 reaches the collision points at the same time as c_2. Hence even after replacing c_1 by c'_1 and c_2 with c'_2, the whole component remains connected with size less than k. By inductive hypothesis c'_1 collides with c'_2.

Now we have the following three scenarios, (a), (b), (c) as shown in Fig 3.4 for scenario (a) and (b), from Lemma 3.1, it is evident that c_3 collides with both c'_1 and c'_2 as shown in Fig 3.5 (a) and (b). Hence it collides with both c_1 and c_2 as well.

In Fig 3.4.(c) if c_3 collides with c_1, then it must also collide with c'_1 which can be proved in a way similar to Lemma 3.1 by considering c_1, c'_1 and c_3. Since c'_1 and c'_2 collide with each other c_3 must also collide with c'_2. Hence c_3 collides with both c_1 and c_2. Else, if c_3 collides with c_2 then it trivially collides with c'_2. Hence c_3 collides with c'_1. Thus c_3 collides c_1 which can be proved in a way similar to Lemma 3.1 by considering c_1, c'_1 and c_3.

2. If $y(q_4) > y(q_1)$ (i.e, P_4 intersects segment p_1q_1 of P_1) then similar arguments can be made but instead of shifting the vertical segment p_1r_1 by some distance, we shift the horizontal segment q_1r_1. This makes sure that replacing c_1 and c_2 with c'_1 and c'_2 respectively doesn't disturb the connectedness. We can have the following three configurations as shown in Fig 3.6.

   For configuration in Fig 3.6 (a) If we shift q_1, r_1 to downward direction and P_3 intersects segments p_1q_1 and p_2q_2 then they intersect as shown in Fig 3.7.(a) where c'_1 and c'_2 represent this modification.

   For configuration in Fig 3.6 (b) If we shift q_1, r_1 to downward direction and

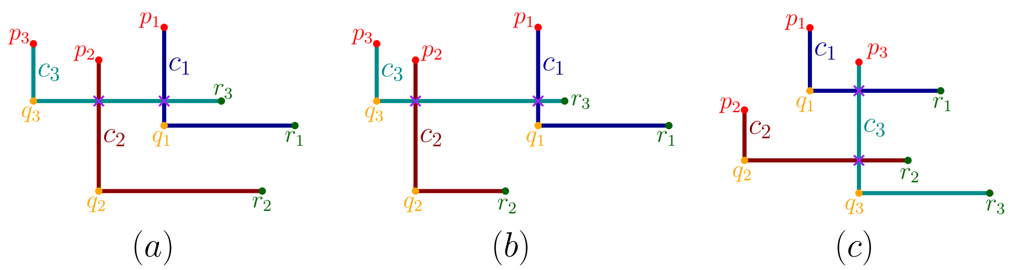

Figure 3.6: Illustration of Case 2.2

extend q_2r_2 in rightward direction and P_3 intersects segments p_1q_1 and p_2q_2 then they intersect as shown in Fig 3.7.(b) where c'_1 and c'_2 represent this modification.

For configuration in Fig 3.6 (c) If we shift q_1, r_1 to downward direction and P_3 intersects segments q_1r_1 and q_2r_2 then they intersect as shown in Fig 3.7.(c) where c'_1 and c'_2 represent this modification.

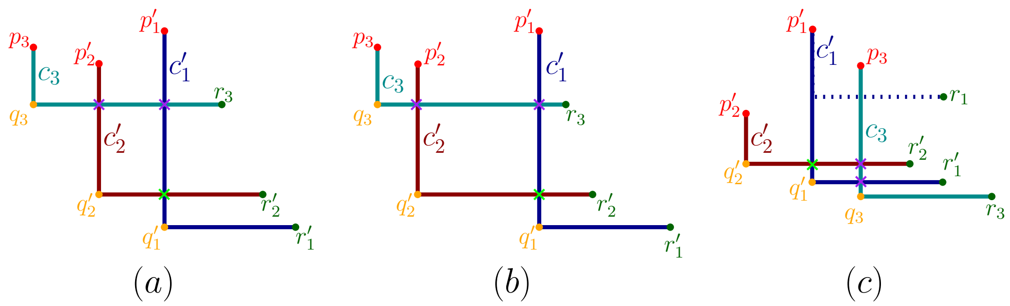

Figure 3.7: Modification for Case 2.2

Replace c_1 with c'_1, c_2 with c'_2 such that $p'_1 = p_1$, $x(q'_1) = x(q_1)$, $y(q'_1) = y(q_2) - \epsilon$, $x(r'_1) = x(r_1)$, $y(r'_1) = y(q'_1)$, $t'_1 = t_1$ and $p'_2 = p_2$, $q'_2 = q_2$, $x(r'_2) = max(x(r_2), x(q_1) + \epsilon)$, $y(r'_2) = y(r_2)$, $t'_2 = t_2$, for some $\epsilon > 0$. The proof can be argued in a similar manner to the above sub-case.

$\square$

Definition 3.8. *We define an L-graph G_L as an intersection graph of L-shaped paths, where each L-shaped path represents a vertex in G_L, and there is an edge between two vertices in G_L if the respective L-shaped paths intersect.*

Now we propose an algorithm to reduce any given instance of G_L to an instance of G_L^t, as follows: For every object $\ell \in G_L$ there exists a vehicle $c \in G_L^t$, such that if and only if $l_i, l_j \in G_L$ has an edge then their corresponding vehicles c_i and c_j collides in G_L^t.

Algorithm 1 Assignment of Time in G_L to obtain G_L^t

procedure ASSIGNTIME(C, S, i)
    insert i into S
    for $\forall j \in C$ **do**
        if $i = 0$ and $j \notin S$ **then**
            set $t_j = 0$
        else
            if $j \notin S$ and intersects with i **then**
                SETTIME(C, i, j)
                ASSIGNTIME(C, S, j)
            end if
        end if
    end for
end procedure

Theorem 3.3. *Given an L-graph, there exists a G_L^t Computable in polynomial time, such that the cardinality of MIS of G_L is k if and only if the cardinality of MIS of G_L^t is k.*

Proof. For each object l_i in L-graph, assign a vehicle c_i with path as l_i and a unit velocity. Insert all vehicles into set C. Let S be an empty set. Now call the procedure ASSIGNTIME$(C, S, 0)$. This will give a time assignment to each and every vehicle. The procedure SETTIME(C, i, j) assigns time t_j such that c_j will collide with c_i (i.e if l_i, l_j intersect at point g then $t_j = t_i + |x(p_i) - x(g)| + |y(p_i) - y(g)| - |x(p_j) - x(g)| - |y(p_j) - y(g)|$).

In the above assignment for each connected component, the time of one of the vehicle is set to zero and every other vehicle is set to collide with at least one of the vehicles in the connected component. Hence from Theorem 3.2 we have, every intersection in G_L as a collision in G_L^t.

This assignment might assign negative time to some vehicles. To ensure that the start time to be non-negative for each vehicle, find the minimum time assignment out of all vehicles and subtract that value from the time of each vehicle. □

Hence as a corollary for the above theorem we can prove that B_1-CRP is NP-Hard.

3.3 Approximation for MIS of G_L^t

We propose an algorithm to partition the G_L^t to collections of G_L's.

Algorithm 2 Procedure to partition G_L^t

```
procedure SEPERATESET(i)
    U = UniversalSet, S = φ
    insert i into S
    for ∀j ∈ U do
        if j ∉ S and collides with i then
            insert j into S
            insert SEPERATESET(j) into S
        end if
    end for
    returnS
end procedure
```

Lemma 3.4. *Any set S generated by the procedure* SEPERATESET *is independent of the set $U \setminus S$, i.e* MIS*(U)* = MIS*(S)* + MIS*($U \setminus S$).*

Proof. Let us assume S is not independent of $U \setminus S$, that implies $\exists i \in U \setminus S$ and $\exists j \in S$ such that i and j are not independent i.e, i and j collides but then by our method SEPERATESET, $i \in S$ which is a contradiction. Hence S is independent of $U \setminus S$. □

Lemma 3.5. *Any set S generated by above algorithm is an L-Graph(G_L).*

Proof. From Lemma 3.4 and Thoerem 3.2 it is evident that S is a connected component and if the path of any two l's intersects then they must collide with each other. Hence we can ignore the time function and say if the paths intersect they collide. This results in nothing but an L-graph. □

Theorem 3.6. *For any L-path graph, there exists an approximation factor equivalent to L-graph. i.e there exists a $O(\log^2 n)$ approximation algorithm.*

Proof. From Lemma 3.4 and Lemma 3.5, it is evident that given any L-path graph, we can separate the L-path graph into subsets $S_1, S_2, \ldots$, and all of them are pair wise independent (i.e no collision between objects from two different sets) and from Theorem 3.2 each set S_i can be treated as an L-graph i.e, each intersection of objects belonging to same set S_i is nothing but a collision in S_i.

Now apply the known approximation algorithm of L-graphs [2] for each S_i, and return the union. Let $\mathcal{O}pt(S_i)$ denote the optimal solution for S_i and $\mathcal{S}ol(S_i)$ denote the solution generated by the algorithm [2]. Since we know $\mathcal{O}pt(S_i) \leq (k \log^2 n)\mathcal{S}ol(S_i)$, summing over all the sets on both sides will result in the desired inequality, $\sum_{i=1} \mathcal{O}pt(S_i) \leq \sum_{i=1} (k \log^2 n)\mathcal{S}ol(S_i)$. This concludes the proof. □

Corollary 3.7. *For a B_1-CRP, there exists a $O(\log^2 n)$ approximation algorithm.*

Proof. If the path of any vehicle in B_1-CRP is a straight line, then append a orthogonal line of length $\delta \approx 0$. Now every path is a single bend path, four different single bends are possible in a grid (└, ┘, ┐, ┌). For each single bend the vehicle can travel in two different ways i.e, the source and destination can be interchanged, hence we have eight different ways a vehicle can move.

Now divided the set of vehicles into 8 disjoint subsets $U_1, U_2, \ldots, U_8$, where each subset has vehicles moving in similar path and direction, due to symmetry each subset hold all the above properties. Solve for each subset U_i as mentioned in Theorem 3.6, let the output for the set U_i be IS_i. Return maximum set among $IS_1, IS_2, \ldots, IS_8$, call it IS_j.

Since $\mathcal{O}pt(U_i) \leq (k \log^2 n)|IS_i|$, therefore $\sum_{i=1}^{8} \mathcal{O}pt(U_i) \leq \sum_{i=1}^{8}(k \log^2 n)|IS_i| \implies \sum_{i=1}^{8} \mathcal{O}pt(U_i) \leq (8k \log^2 n)|IS_j|$. Thus concludes the proof. □

3.4 Unit L Graph Approximation

Definition 3.9. *A unit L-graph G_{LU} is a special graph of G_L where each L-shaped path is of the unit size, i.e, both the horizontal and vertical segments are of unit length each.*

In this section, we design a 2-approximation algorithm for the maximum independent set in a unit L-graph problem. Let $S = \{P_1, P_2, \ldots, P_n\}$ be a set of n unit L-shaped paths in a plane. We first place vertical lines from leftmost to rightmost with a unit distance between each consecutive pair of lines. Assume that there are k such vertical lines $\{L_1, L_2, \ldots, L_k\}$. Let $S_i \subseteq S$ be the set of L-shaped paths intersected by the line L_i. The idea is to find MIS for each S_i and then combine them to produce an approximate solution. This method is well known for finding an approximate solution for MIS of fixed height rectangle by Agarwal et al. [6] and for the unit disk by Nandy et al. [7]. But our problem is different in a sense that the intersection graph $I(S_i)$ of a S_i may not be a triangulated graph. We can construct an $I(S_i)$ that contains a four-cycle as shown in Fig 3.8. So we show that $I(S_i)$ is a co-comparable graph. Then we give a dynamic programming based algorithm that solves MIS of $I(S_i)$ in $O(n^2)$ time using $O(n)$ space.

Observation 3.2. *Any two L-shaped paths, $P_a \in S_i$ and $P_b \in S_i$ are independent if $|y(q_a) - y(q_b)| > 1$, for $1 \leq i \leq k$.*

Observation 3.3. *Any two L-shaped paths, $P_a \in S_i$ and $P_b \in S_i$ with $y(q_a) < y(q_b)$ are independent if $x(q_a) < x(q_b)$.*

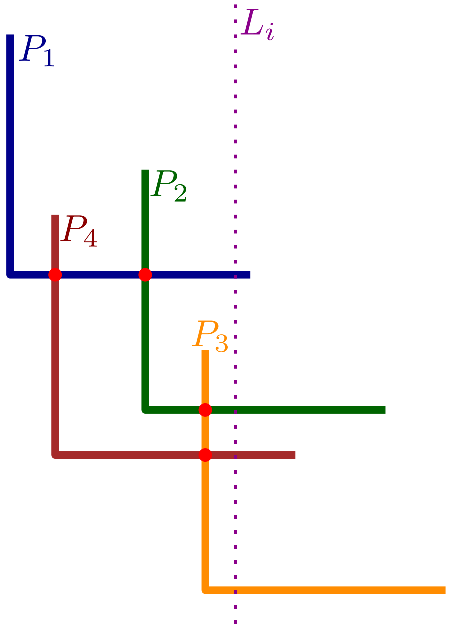

Figure 3.8: G_{LU} with four cycle

Observation 3.4. *Any two L-shaped paths, $P_a \in S_i$ and $P_b \in S_j$ are independent if $|i-j| > 1$, for $1 \leq i, j \leq k$.*

Lemma 3.8. *If P_1, P_2, P_3 are three unit L-shaped paths that intersect a vertical line L_i such that (i) $y(q_1) > y(q_2) > y(q_3)$, (ii) P_1, P_2 doesn't intersect and (iii) P_2, P_3 doesn't intersect, then P_1, P_3 doesn't intersect.*

Proof. Since all the three unit L-shaped paths intersect with the vertical line L_i, we have the following three cases.

Case 1**:** $y(q_1) - y(q_2) \geq 1$. Since P_3 is a unit L and $y(q_3) < y(q_2)$, therefore $y(q_1) - y(q_3) > 1$. Hence P_1 and P_3 cannot intersect.

Case 2**:** $y(q_2) - y(q_3) \geq 1$. Since P_3 is a unit L and $y(q_1) > y(q_2)$, therefore $y(q_1) - y(q_3) > 1$. Hence P_1 and P_3 cannot intersect.

Case 3**:** $y(q_1) - y(q_2) < 1$ and $y(q_2) - y(q_3) < 1$. Since P_1 and P_2 doesn't intersect with each other, thus $x(q_1) > x(q_2)$. Similarly $x(q_2) > x(q_3)$, therefore $x(q_1) > x(q_3)$. Hence P_1 and P_3 cannot intersect.

□

Definition 3.10. *We denote* $\widetilde{G} = (V, \widetilde{E})$ *as the complimentary graph of* $G = (V, E)$*, such that* $(u, v) \in \widetilde{E}$ *if and only if* $(u, v) \notin E$*, for all* $u, v \in V$*.*

Lemma 3.9. *The graph* $\widetilde{G}_{LU}$ *of the unit L-shaped path intersecting a vertical line* L_i *is a Comparable graph.*

Proof. We show that $\widetilde{G}_{LU}$ is orientable, such that if there is a directed edge from vertex a to vertex b and there is a directed edge from vertex b to vertex c, then there is a directed edge from vertex a to vertex c, for all vertices $a \neq b \neq c$ in the $\widetilde{G}_{LU}$.

The ordering of vertices is as follows: A vertex a precedes a vertex b if the Y-coordinates of the respective L-shaped paths P_a and P_b follow the inequality $y(q_a) > y(q_b)$. Now if there is an edge between any two vertices a and b in the graph $\widetilde{G}_{LU}$ and a precedes b then direct the edge from a to b.

In the above mentioned ordering, we can conclude that $\widetilde{G}_{LU}$ is a comparable graph because if and only if there is an edge between a, b, and b,c in $\widetilde{G}_{LU}$ then a,b and b,c are independent in G_{LU}. Since they are in increasing order, by Lemma 3.8 a,c is also independent in G_{LU}.Thus there is an edge between a and c in $\widetilde{G}_{LU}$. This proves the lemma. □

Corollary 3.10. *The graph* $G_{LU}(L_i)$ *formed by unit L-shaped paths which are intersecting with a vertical line* L_i *is a Co-comparable graph i.e the graph* $G_{LU}(L_i)$ *formed by* S_i *is Co-comparable.*

Given any S_i, we sort the elements based on their Y-coordinates. i.e, a path P_a will have an index less than P_b if $y(q_a) < y(q_b)$. For the sake of simplicity we refer to the path at index k as P_k.

For any index k, let $R(k)$ be the maximum possible independent set till k that includes the path P_k and let $J_k = \{P_{j_1}, P_{j_2}, \ldots, P_{j_l}\}$ be the set of all paths that doesn't intersect with P_k and have index less than k.

Observation 3.5. $R(k) = \begin{cases} 1 & \textit{if } J_k = \phi \\ 1 + max(R(j_1), R(j_2), \ldots, R(j_l)) & \textit{otherwise} \end{cases}$

Algorithm 3 Computing $R(k)$ for each index in S_i

```
procedure lineIntersectMIS(S_i)
    R, B are arrays of size |S_i|
    for k = 1 to |S_i| do
        Set R(k) = 0, B(k) = −1
    end for
    R(1) = 1
    for k = 2 to |S_i| do
        for j = 1 to k − 1 do
            if P_k and P_j doesn't intersect then
                if R(j) > R(k) then
                    R(k) = R(j)
                    B(k) = j
                end if
            end if
        end for
        R(k) = R(k) + 1
    end for
    return R, B
end procedure
```

Lemma 3.11. *The recurrence to compute the maximum independent set in S_i till index k is $MIS(k) = max(MIS(k-1), R(k))$.*

Proof. Consider the optimal solution $MIS(k)$. There are two cases: Either P_k is in the maximum independent set or it is not.

Case 1: If P_k is not in the maximum independent set then the maximum independent set must have been from 1 to $k-1$. By definition this is $MIS(k-1)$.

Case 2: If P_k is in the maximum independent set then by Observation 3.5 this is $R(k)$.

□

If we compute $R(k)$ for all index k, then in a single run i.e $O(|S_i|)$ we can compute the maximum independent set for S_i.

Note that, the procedure $\text{LINEINTERSECTMIS}(S_i)$ can be modified to solves MIS for S_i optimally. Run LINEINTERSECTMIS on each S_i, for $1 \leq i \leq k$ and let E_i be the maximum independent in S_i. We define two sets $Even_{OPT} = \bigcup\limits_{\substack{1\leq i\leq k \\ i \text{ is even}}} E_i$ and $Odd_{OPT} = \bigcup\limits_{\substack{1\leq i\leq k \\ i \text{ is odd}}} E_i$. We report the set with the maximum cardinality among $Even_{OPT}$ and Odd_{OPT} as the result of our algorithm. Thus we have the following theorem.

Theorem 3.12. *Our algorithm produces a 2-approximation for MIS in G_{LU}, with a time complexity of $O(n^2)$ and a space complexity of $O(n)$.*

Proof. We know $S = \bigcup\limits_{i=1}^{k} S_i$. Hence $\sum\limits_{i=1}^{k} |S_i| = |S| = n$. Therefore $\sum\limits_{i=1}^{k} |S_i|^2 \leq n^2$. Thus the running time is $O(n^2)$. Since for each S_i we used an $O(|S_i|)$ space therefore the total space complexity if $O(n)$.

Let OPT be the optimal solution for S. From observation 3.4, we can conclude that the L-shaped paths in $Even_{OPT}$ are independent and so are Odd_{OPT}. We have $Even_{OPT} + Odd_{OPT} \geq OPT$. Thus $\max\{|Even_{OPT}|, |Odd_{OPT}|\} \geq \frac{|OPT|}{2}$. □

Similarly for a Unit restricted B_1-VPG we can get an 8 approximation using similar approach as stated in Corollary 3.7 since we have four different bends possible and we solve for each set and return the maximum among them.

Chapter 4

Conclusion And Future Work

In this project, we obtained hardness results and approximation algorithms for $CCRP$ and B_1-CRP. We showed that $G_{LU}(\ell)$ is a Co-comparable graph. We proposed a dynamic programming based algorithm for finding MIS of $G_{LU}(\ell)$ in $O(n^2)$ time using linear space. Which produces 2-approximation for finding MIS of G_{LU} with $O(n^2)$ time and $O(n)$ space complexity. Finally we pose the following open problems:

1. Can a 2-approximation for MIS of G_{LU} be obtained in sub-quadratic time?
2. Does there exist a polynomial time sub-linear approximation algorithm for CRP when the vehicles are moving only along XY-monotone paths?
3. Does there exists a better approximation for B_k-CRP than MIS for general graph?

Bibliography

[1] Ajaykumar, J., Das, A., Saikia, N., & Karmakar, A. (2016, August). Problems on One Way Road Networks. In *Canadian Conference on Computational Geometry* (p. 303).

[2] Lahiri, A., Mukherjee, J., & Subramanian, C. R. (2015). Maximum Independent Set on B_1-VPG Graphs. In *Combinatorial Optimization and Applications* (pp. 633-646). Springer, Cham.

[3] Dasler, P., & Mount, D. M. (2015, August). On the complexity of an unregulated traffic crossing. In *Workshop on Algorithms and Data Structures* (pp. 224-235). Springer, Cham.

[4] Asinowski, A., Cohen, E., Golumbic, M.C., Limouzy, V., Lipshteyn, M., & Stern, M. (2012). Vertex Intersection Graphs of Paths on a Grid. *J. Graph Algorithms Appl.*, 16, 129-150.

[5] Håstad, J. (1997). Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica.*, 182(1), 105-142.

[6] Agarwal, P. K., van Kreveld, M., & Suri, S. (1998). Label placement by maximum independent set in rectangles. *Computational Geometry: Theory and Applications*, 3(11), 209-218.

[7] Nandy, S. C., Pandit, S., & Roy, S. (2017). Faster approximation for maximum independent set on unit disk graph. *Information Processing Letters*, 127, 58-61.

[8] Golumbic, M. C., Rotem, D., & Urrutia, J. (1983). Comparability graphs and intersection graphs. *Discrete Mathematics*, 43(1), 37-46.

[9] Crescenzi, P., Fiorini, C., & Silvestri, R. (1991). A note on the approximation of the MAX CLIQUE problem. *Information Processing Letters*, 40(1), 1-5.

[10] Michael, R. G., & David, S. J. (1979). Computers and intractability: a guide to the theory of NP-completeness. *WH Free. Co., San Fr*, 90-91.

[11] Kashiwabara, T., & Fujisawa, T. (1979). NP-Completeness of the problem of finding a minimum clique number interval graph containing a given graph as a subgraph. In *Proceedings International Conference on Circuits and Systems*, 657-660.

[12] Gavril, F. (1972). Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph. *SIAM Journal on Computing*, 1(2), 180-187.

[13] Dilworth, R. P. (1950). A decomposition theorem for partially ordered sets. *Annals of Mathematics*, 161-166.

[14] Mirsky, L. (1971). A dual of Dilworth's decomposition theorem. *The American Mathematical Monthly*, 78(8), 876-877.

[15] Lovász, L. (1972). Normal hypergraphs and the perfect graph conjecture. *Discrete Mathematics*, 2(3), 253-267.

[16] Hochbaum, D. S. (1996). Approximation algorithms for NP-hard problems.*PWS Publishing Co.*.

[17] Lovász, L. (2009). A characterization of perfect graphs. *In Classic Papers in Combinatorics* (pp. 447-450). Birkhäuser Boston.

[18] Konig, D. (1931). Graphok es matrixok (Hungarian)[Graphs and matrices]. *Matematikai és Fizikai Lapok*, 38, 116-119.

[19] Malhotra, V. M., Kumar, M. P., & Maheshwari, S. N. (1978). An $O(|V|^3)$ algorithm for finding maximum flows in networks. *Information Processing Letters*, 7(6), 277-278.

[20] Rose, D. J., Tarjan, R. E., & Lueker, G. S. (1976). Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on computing*, 5(2), 266-283.