Approximation Scheme for Vehicle Routing Problems

by

Zhong Ren

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

In this thesis we consider the point to point orienteering and deadline traveling salesman problems on graphs with bounded treewidth and graphs with consant doubling dimension and present approximation schemes for them. These are extensions of the classic Traveling Salesman Problem (TSP). Suppose we are given a (weighted) graph $G = (V, E)$. In point to point orienteering we are given a length budget B, start and end nodes $s, t \in V$ and the goal is to find a path of length at most B starting at s and ending at t that visits as many vertices as possible. In deadline TSP we are given a start node $s \in V$ and $D(v) \geq 0$ (called deadline) for all $v \in V$ and the goal is to find a path starting at s that visits as many vertices as possible before their deadline (where the visit time of a node is the distance travelled from s to that node). On general metrics, the best approximation for point to point orienteering is $(2+\epsilon)$-approximation [8] and $O(\log n)$-approximation for deadline TSP [4]. On Euclidean space, [10] shows a polynomial time approximation scheme (PTAS) for rooted oritenteering. For graphs with bounded treewidth ω we show point to point orienteering can be solved exactly in polynomial time and a quasi-polynomial time approximation scheme (QPTAS) for deadline TSP when the distances are quasi-poly bounded and integers. For graphs with constant doubling dimension, we show QPTAS for point to point orienteering when the distances are quasi-poly bounded and a QPTAS for deadilne TSP when the distances are quasi-poly bounded and integers.

Preface

This thesis is an original work by Kinter Ren under the supervision of professor Mohammad R. Salavatipour. No part of this thesis has been previously published.

Acknowledgements

First I would like to thank my supervisor, Mohammad Salavatipour, for his support during my the master program. He gives me a good guidance through the whole research process from where I learn a lot. He is also very patient to give many constructive comments and suggestions on the thesis.

I also want to thank to all other people in the group, especially Mohsen Rezapour for the discussions during the early and middle stages of the research, and Zachary Friggstad for explaining the idea in his paper very clear which would be adapted in this thesis.

Finally, I would like to thank the Department of Computing Science for financial supporting during my program.

Contents

Chapter 1

Introduction

Vehicle routing problems have been studied extensively in theoretical computer science and have a large variety of applications. Generally speaking, there are two types of vehicle routing problems. One type is that we have a set of clients that need to be visited, and we seek to find the most $'$effective$'$ route for visiting these clients. We list some well-known problems in this category:

- Travelling Salesman Problem (TSP): find the shortest route that visits all the clients.
- Minimum latency problem: find a single route that visits all the clients minimizing the sum of waiting times for all the clients.
- k-TSP: find the shortest route that visits k clients.
- Capacitated Vehicle Routing Problem (CVRP): find the shortest collection of routes visiting all the clients such that each route satisfies a given capacity constraint.

Another type is that we have limited resources, and we need to select a set of clients to visit and plan the most 'suitable' route for these clients. We list some well-known problems in this category:

- Orienteering problem: find a single route maximizing the number of clients visited such that it satisfies the length budget given.

- Deadline TSP: find a single route maximizing the number of clients visited such that every clients on the route is visited before its given deadline constraint.

- TSP with time window: find a single route maximizing the number of clients visited such that every client on the route is visited within the time window constraint given.

All the problems mentioned above are **NP**-hard in general: it is widely believed that there is no time efficient algorithm to solve these problems exactly for arbitrarily large input. One different strategy is to find a time efficient algorithm that returns a nearly optimal solution. We call such an algorithm an approximation algorithm. In this thesis, we consider efficient algorithms that produce approximate solutions, that is, the solutions they return are at most α times worse than an exact optimal solution.

We focus on orienteering problem and deadline TSP on certain metrics and give approximation algorithms for them. In the rest of this chapter we first begin with an introduction to the terminology and concepts used in our thesis, then formalize the problems we consider, discuss related work in the literature, and state the results we obtain.

1.1 Preliminary

We start formalizing the terminology we will use throughout this thesis. The definitions are mainly adapted from [19], [20].

1.1.1 Graph

We only consider undirected simple graphs in the thesis, such a graph is defined by a vertex set $V(G) = \{v_1, \cdots, v_n\}$, an edge set $E(G)$, where each edge $e \in E(G)$ is an unordered pair of distinct vertices, i.e. $E(G) = \{(u, v) : u, v \in V(G), u \neq v\}$. To simplify notation, we refer to graph G as undirected simple graph, and refer V and E as $V(G)$ and $E(G)$ when G is clear from the context

and denote $G = (V, E)$. The size of G, denoted as $|G|$, is defined to be $|V|$, i.e. the number of the vertices in G.

For an edge $(u, v) \in E$, u and v are adjacent and called the endpoints of e. The neighbours of a vertex v, denoted as $N(v)$, is the set of vertices u such that u are v are adjacent, i.e. $N(v) = \{u \in V : (u, v) \in E\}$. A subgraph of graph a G is a graph G' such that $V(G') \subset V(G)$, $E(G') \subset E(G)$ and if $(v_1, v_2) \in E(G')$ then $v_1, v_2 \in V(G')$.

A path is a graph where the vertices can be ordered such that two vertices are adjacent if and only if they appear consecutively in the ordering, i.e. a path P is a sequence of distinct vertices $\langle v_1, v_2, \cdots, v_{m+1} \rangle$ which joins a sequence of edges $\langle e_1, e_2, \cdots, e_m \rangle$ where $e_i = (v_i, v_{i+1})$ for $1 \leq i \leq m$. Note the size of P, denoted as $|P|$, is the number of vertices visited along the path. We call v_1 the start node of P and v_{m+1} the end node of P. A subpath of P is a path with a sub-sequence of vertices of P provided they are consecutively adjacent. A rooted path is a path with specified start node s. A s-t path is a path with specified start-end node pair s and t. A cycle is a graph with an equal number of distinct vertices and edges where the vertices can be placed around a circle such that two vertices are adjacent if and only if they appear consecutively along the circle.

A connected graph is a graph G where for any $u, v \in V$, there is a u-v path in G. An acyclic graph is a graph G that does not contain any cycle. A tree T is an acyclic, connected graph. A rooted tree T is a tree with a specified vertex $r \in V$, for each vertex $v \in T$, let $P(v)$ be the unique r-v path in the tree. The parent of v is its neighbour that is in $P(v)$. The children of v are its other neighbours. The ancestors of v are the vertices in $P(v)$ excluding v. The descendants of v are the vertices u such that $P(u)$ contains v. The leaves of T are the vertices having no children. The height of T is the maximum size of $P(v)$ for $v \in T$. The branching factor of T is maximum number of children of $v \in T$. A binary tree is a tree with branching factor 2.

A tree decomposition of a graph $G = (V, E)$ consists of a rooted tree T, for each vertex $t \in V(T)$ (we call t a bag from now on), it is associated with a subset $V_t \subset V(G)$. T and $\{V_t : t \in T\}$ satisfy the following properties:

- every vertex $v \in V(G)$, it is in some bag V_t, i.e. $\cup_{t \in V(T)} V_t = V(G)$.
- for every edge $e \in E(G)$, there is some bag V_t containing both endpoints of e.
- for every vertex $v \in V(G)$, the set of bags whose corresponding subset in V containing v, denoted as T_v i.e. $T_v = \{t \in T : v \in V_t\}$, is a connected subtree of T,

The width of a tree decomposition T is one less than the maximum size of bags in the decomposition. The treewidth of a graph G is the minimmum ω such that there exists a tree decomposition T with the width ω. Note a tree has treewidth 1.

1.1.2 Metrics

A weighted graph is a graph with a mapping $C : E \to \mathbb{Q}^{\geq 0}$, we call $C(e)$ the weight of the edge e. A metric space is an ordered pair (X, d_X) where X is a set of points and d_X is a mapping $X \times X \to \mathbb{Q}^{\geq 0}$ satisfying the following properties:

- For all $x \in X$, $d_X(x, x) = 0$.
- For any $x, y \in X$ and $x \neq y$, $d_X(x, y) > 0$.
- For any $x, y \in X$, $d_X(x, y) = d_X(y, x)$.
- For any $x, y, z \in X$, $d_X(x, z) \leq d_X(x, y) + d_X(y, z)$, which is referred to as triangle inequality.

A metric space (X, d_X) can be represented by a complete weighted graph G such that $V(G) = X$ and $C(e) = d_X(u, v)$. A metric graph is such complete weighted graph that represents some metric space. Let P be a path in a weighted graph $G = (V, E)$, the length of a path P, denoted as $||P||$, is the sum of the weights of the edges along the path, i.e. $||P|| = \sum_{e \in P} C(e)$. For any $u, v \in V$, the distance between u and v, denoted as $d(u, v)$, is length of the shortest u-v path. Metric completion of G is a complete weighted graph

$G' = (V, E')$ where for any $u, v \in V$, the weight of $(u, v) \in E'$ is $d(u, v)$, i.e. length of the shortest u-v path in G. In this thesis when we consider a weighted graph we implicitly refer it to its metric completion.

Distance between u and v is called poly bounded if $d(u, v) = O(|V|^c)$ for some constant $c > 0$. We allow the distance extend to $|V|^{O(\log^c |V|)}$ for some constant c and call it quasi-poly bounded in this case. The diameter of G, denoted as Δ_G, is the maximum distance between all possible vertices, i.e $\Delta_G = \max_{u,v \in V} d(u, v)$. G is called poly bounded if for any pair of vertices in G, the distance between them is poly bounded.

For a metric space (X, d), $x \in X$ and $r > 0$, a ball of center x with radius r, denoted as $B_x(r)$, is the set of points containing all y such that the distance between x and y is at most r. i.e. $B_x(r) = \{y : d(x, y) \leq r\}$. The doubling dimension of X is the smallest $\kappa \in \mathbb{Z}^+$ such that for any $x \in X$ and $r > 0$, for the ball $B_x(r)$ there exists $Y \subset X$ that $B_x(r) \subset \cup_{y \in Y} B_y(\frac{r}{2})$ and $|Y| \leq 2^\kappa$. Let G be the complete weighted graph that is converted by (X, d). The doubling dimension of G is defined to be κ as well.

A cluster C in the metric (X, d) is a subset of nodes in X. A decomposition of the metric (X, d) is a partitioning of X into clusters. A hierarchical decomposition of X is a sequence of partitions of X, where each partition is a refinement of the previous one. Normally this is represented by a split-tree T, where each node of T corresponds to a cluster. We use C to both refer to a node in T as well as the cluster (set of vertices in X) it corresponds to. The root node of T corresponds to the single set $\{X\}$ and the leaf nodes correspond to singleton sets $\{\{x\}\}_{x \in X}$. The children of each node $C \in T$ correspond to a partition of C where each part has diameter about half of that of C. The union of all subsets corresponding to the vertices at each level in this split-tree constitutes a partition of X.

1.1.3 Optimization Problem and Approximation Algorithm

A decision problem Π is a problem that for any instance I we determine whether it has a feasible solution s of size polynomial in $|I|$ or not, where $|I|$

is the size of I. A decision problem is polynomial time solvable if there exists an algorithm such that for any instance I, the algorithm can find a feasible solution s if there is any, or outputs 'there's no feasible for instance I' in time polynomial in $|I|$, i.e. $O(|I|^c)$ for some constant $c > 0$. The complexity class P is the set of decision problems that are polynomial time solvable. A verifier for a decision problem is an algorithm that given an instance I and a proposed solution s it determines whether s is feasible solution of I or not. The complexity class NP is the set of decision problems that have a verifier running in time polynomial in $|I|$.

The optimization problem Π consists of a set of valid instance D_Π, a set of feasible solution $S_\Pi(I)$ for any instance $I \in D_\Pi$ where for each solution $s \in S_\Pi(I)$ the size of s is polynomial in $|I|$, an objective function obj_Π that assigns each instance-solution pair (I, s) a non-negative value, which can be computed in time polynomial in $|I|$. We also specify whether Π is a minimization problem or a maximization problem. For a minimization (maximization) problem Π and instance $I \in D_\Pi$, an optimal solution is a feasible solution $s \in S_\Pi(I)$ that that minimizes (maximizes) the value obj_Π, i,e. $\arg\min_{s \in S_\Pi} obj_\Pi(I, s)$/ $\arg\max_{s \in S_\Pi} obj_\Pi(I, s)$. We denote such a solution as $OPT_\Pi(I)$ and denote the objective value of the optimal solution as OPT.

For optimization problems we consider in the thesis, the decision versions have been shown to be NP-hard. Unless $P = NP$, there is no polynomial time algorithm that can solve these decision versions and optimization problems exactly. One strategy is to find a time efficient algorithm that returns nearly optimal solution. We call it approximation algorithm. Let Π be a minimization (maximization) problem, and let $\alpha : \mathbb{Z}^+ \to \mathbb{Q}^+$ be a function such that $\alpha \geq 1$ for all inputs. An algorithm $\mathcal{A}$ is an α-approximation for Π if for any instance, $\mathcal{A}$ returns a feasible solution $s \in S_\Pi(I)$ such that $obj_\Pi(I, s) \leq \alpha(|I|)OPT_\Pi(I)$ ($obj_\Pi(I, s) \geq \frac{OPT_\Pi(I)}{\alpha(|I|)}$) and the running time is polynomial in $|I|$. We allow the running time extend to $|I|^{O(\log^c |I|)}$ for some constant $c > 0$ and call it quasi-polynomial in this case. The function α is called the approximation ratio of $\mathcal{A}$.

An approximation scheme for a minimization (maximization) problem Π

is a class of algorithms that takes a valid instance I as well as a parameter $\epsilon > 0$ as input such that for any fixed ϵ, the scheme is a $(1+\epsilon)$-approximation algorithm. We call $\mathcal{A}$ a polynomial time approximation scheme (PTAS) if its running time is polynomial in $|I|$ for each fixed ϵ. We call $\mathcal{A}$ a quasi polynomial time approximation scheme (QPTAS) if its running time is quasi polynomial in $|I|$ for each fixed ϵ.

1.2 Problems Considered

We list the problems we consider in this thesis below:

Point to point k-TSP.

In this problem, we are given a (weighted) graph $G = (V, E)$, an integer $k > 0$, and a start-end node pair s, t. The goal is to find a s-t path P where the length of P i.e. $||P||$ is minimized while the number of vertices visited $|P|$ is at least k.

Point to point orienteering problem.

In this problem, we are given a (weighted) graph $G = (V, E)$, a length budget $B > 0$ and a start-end node pair s, t. The goal is to find a s-t path P where the number of vertices visited i.e. $|P|$ is maximized such that the length of the path $||P||$ is at most B.

Deadline TSP.

In this problem, we are given a (weighted) graph $G = (V, E)$, a start node $s \in V$ and $D(v) \geq 0$ (called deadline of v) for all $v \in V$. The goal is to find a path P rooted at s with the number of vertices visited, i.e. $|P|$ is maximized such that for every $v \in P$, $||P_{sv}|| \leq D(v)$, where P_{sv} is the subpath of P from s to v.

1.3 Related Work

For TSP, Arora [2] and Mitchell [16] give the first polynomial time approximation schemes on Euclidean space. They extend the result to some variants of it such as k-TSP. Arora et al. [3] present a polynomial time approximation

schemes on planar graph. Talwar [17] present a quasi polynomial time approximation schemes on doubling metrics. Bartal et al. [5] presented a polynomial time approximation schemes on doubling metrics building upon the work of [17].

For orienteering, Blum et al. [6] give the first constant-factor approximation algorithm on general metrics. Chekuri et al. [8] improve the approximation factor to $(2+\epsilon)$. Arkin et al. [1] presented a $(2+\epsilon)$-approximation on Euclidean space. Chen and Har-Peled [10] give the first polynomial time approximation schemes on Euclidean space. Rrecently Gottlieb et al. [14] presented a more efficient polynomial time approximation schemes for on Euclidean space.

For deadline TSP, Bansal et al. [4] give polynomial time $O(\log n)$ approximation algorithm on general metric. They also provides a $(O(\log \frac{1}{\epsilon}), 1+\epsilon)$-bicriteria approximation. Assuming distances (and deadlines) that are integers, this implies an $O(\log D_{\max})$-approximation where $D_{\max}$ is the maximum deadline. Chekuri and Kumar [9] give a polynomial time 3-approximation algorithm on general metric assuming there are only a constant number of distinct deadlines. Levin and Farbstein [12] give a polynomial time $(3(1+\epsilon), 1+\epsilon)$-bicriteria approximation algorithm on general metric. They also give a polynomial time $(1+\epsilon, 1+\epsilon)$-bicriteria approximation for deadline TSP on weighted trees assuming distances are poly bounded. Friggstad and Swamy [13] give a quasi-polynomial time $(7.63+\epsilon)$-approximation algorithm on general metrics assuming distance are quasi-poly bounded and integers.

For window-TSP, Bansal et al. [4] present an $O(\log^2 n)$-approximation for general metrics. Chekuri et al. [8] show that any α-approximation for point to point implies an $O(\alpha \max\{\log opt, \log \frac{L_{\max}}{L_{\min}}\})$-approximation for the time-window version where opt is the optimal value and $L_{\max}$ and $L_{\min}$ are the sizes of the largest and smallest windows.

1.4 Our Result

We list the main results of this thesis:

Theorem 1 *Let $G = (V, E)$ be a graph with bounded treewidth ω, given a budget $B > 0$ and start-end node pair $s, t \in V$ as an instance of point to point orienteering on G. Let $n = |V|$, we can find an optimal solution in polynomial time in n.*

Theorem 2 *Let $G = (V, E)$ be a graph with constant doubling dimension κ, given a budget $B > 0$ and start-end node pair $s, t \in V$ as an instance of point to point orienteering on G. Let $\delta = \log \Delta_G$ and $n = |V|$, with probability at least $1 - \frac{1}{n}$ we can find a $(1 + \epsilon)$-approximation for this instance in time $n^{O((\frac{\delta}{\epsilon})^{4\kappa+1})}$.*

Theorem 3 *Let $G = (V, E)$ be a graph with bounded treewidth ω, given a start node $s \in V$ and $D(v)$ for all $v \in V$ as an instance of deadline TSP on G. Let $\delta = \log \Delta_G$ and $n = |V|$, we can find a $(1 + \epsilon)$-approximation for this instance in time $n^{O((\frac{\omega\delta}{\epsilon})^2)}$ assuming all distance are integers.*

Theorem 4 *Let $G = (V, E)$ be a graph with constant doubling dimension κ, given a start node $s \in V$ and $D(v)$ for all $v \in V$ as an instance of deadline TSP. Let $\delta = \log \Delta_G$ and $n = |V|$, with probability at least $1 - \frac{1}{n}$ we can find a $(1+\epsilon)$-approximation for this instance in time $n^{O((\frac{\delta}{\epsilon})^{4\kappa+2})}$ assuming all distance are integers.*

A (α, β)-approximation for deadline TSP is for any given instance, let OPT be its optimal value, then it returns a path P such that $|P| \leq \alpha \cdot OPT$ and for each $v \in P$, $||P_{sv}|| \leq (1+\beta)D(v)$, in other words the visiting time of v can be violated its deadline by at most β factor. Without the assumption that all distances are integers (fractional value instead) in Theorem 3 and Theorem 4, we can get a bicritiria approximation respectively:

Theorem 5 *Let $G = (V, E)$ be a graph with bounded treewidth ω, given a start node $s \in V$ and $D(v)$ for all $v \in V$ as an instance of deadline TSP on G. Let $\delta = \log \Delta_G$ and $n = |V|$, we can find a $(1 + \epsilon, 1 + \epsilon)$-bicritiria approximation for this instance in time $n^{O((\frac{\omega\delta}{\epsilon})^2)}$.*

Theorem 6 *Let $G = (V, E)$ be a graph with constant doubling dimension κ, given a start node $s \in V$ and $D(v)$ for all $v \in V$ as an instance of deadline TSP. Let $\delta = \log \Delta_G$ and $n = |V|$, with probability at least $1 - \frac{1}{n}$ we can find a $(1+\epsilon, 1+\epsilon)$-bicritiria approximation for this instance in time $n^{O((\frac{\delta}{\epsilon})^{4\kappa+2})}$.*

In the rest of thesis, we prove Theorem 1 and 2 in Chapter 2. Then we prove Theorem 3, 4, 5 and 6 in Chapter 3. We discuss some possible future directions in Chapter 4.

Chapter 2

Point to Point Orienteering

In this chapter, we consider point to point orienteering on different metrics. In Section 2.1 we consider graph with bounded treewidth and prove Theorem 1. In Section 2.2 we consider grpah with constant doubling dimension and prove Theorem 2.

2.1 Point to Point Orienteering on Graphs with Bounded Treewidth

Let $G = (V, E)$ be a graph with bounded treewidth ω and $n = |V|$. Given a length budget B, start and end node pair $s, t \in V$ as an instance of point to point orienteering on G. We show how to solve the point to point orienteering instance exactly in polynomial time in n. Let P^* be the optimal for the point to point orienteering instance and $k = |P^*|$. Note P^* is a feasible solution for the instance of point to point k-TSP on G with specified k and the same start-end node pair $s, t \in V$ as the given instance of point to point orienteering. Let P' be the optimal for the point to point k-TSP instance. Note $||P'|| \leq ||P^*|| \leq B$

Let T be a tree decomposition of G. Bodlaender and Hagerup [7] show that one can build a tree decomposition T of G with the following properties:

- T is binary.
- The height of T is $\rho \log n$ for some constant $\rho > 0$.
- The width of T is at most $\omega' = 3\omega + 2 = O(\omega)$.

We assume that T possesses these additional properties and use ω (instead of ω') to refer to its width. Note the the size of T is then at most $2^{\rho \log n} = n^{\rho}$. For a bag $b \in T$, recall V_b is the set containing vertices associated with bag b. Let C_b denote the union of the associated vertices in bags below and including b and G_b denote the corresponding subgraph in G over the vertices in C_b, i.e. $G(C_b)$. Note from [7] for any bag b, it forms a boundary of G_b and $G - G_b$, i.e. for any path from a vertex in G_b to a vertex in $G - G_b$ must pass through V_b. Let b_1 and b_2 be children bags of b in T, we define R_b to be the set of edges in G_b crossing b_1 and b_2, i.e. $R_b = \{(u, v) : u \in G_{b_1}, v \in G_{b_2} \text{ or } u \in G_{b_2}, v \in G_{b_1}\}$. From the construction of R_b, $|R_b| \leq |G_{b_1}||G_{b_2}| \leq (\omega + 1)^2 = O(\omega^2)$.

We show a dynamic programming built based on T that can find P'. Recall T_v is set of the bags in T containing v which is a connected subtree of T. In order to avoid overcounting, for every vertex $v \in V(G)$, we put a token on v at the root of T_v, i.e. the bag containing v that is closest to the root bag in T. Note $'$a token in T is picked up$'$ means that the corresponding vertex of the token in G is visited. For a path P, we adapt the notation $|P|$ to refer to the number of tokens picked by P in T (instead of the number of vertices visited by P in G).

Note that for a bag $b \in T$, P' restricted in G_b may be a collection of disjoint paths where they all enter and exit G_b via V_b. Since $|b| \leq \omega + 1$ the number of such subpaths is at most $O(\omega^2)$. We introduce the notion of multiple point to point k-TSP in order to precisely define subproblems in the dynamic programming:

Definition 1 *Let $G = (V, E)$ be a graph, given an integer k and σ start-end node pairs $(s_1, t_1), \cdots, (s_\sigma, t_\sigma)$ (where σ can be at most poly logarithmic in n) as an instance of multi-path k-TSP is to find a collection of paths $\{P_1, \cdots, P_\sigma\}$ such that P_i is a s_i-t_i path in G and $|P_1 \cup \cdots \cup P_\sigma| = k$, with the total length $\sum_{i=1}^{\sigma} ||P_i||$ minimized.*

Note point to point k-TSP is a special case of multi-path k-TSP where there is only one start-end node pair instead of multiple start-end node pairs and the goal is to find a single path instead of a collection of paths. We

define a subproblem in the dynamic programming as an instance of multi-paths k-TSP with specified bag $b \in T$, integer k_b and σ_b start-end node pairs $(s_i, t_i), 1 \leq i \leq \sigma_b$, the goal is to find a collection of paths $P_i, 1 \leq i \leq \sigma_b$ such that P_i is a s_i-t_i path in $G(b)$ and $|\cup_{i=1}^{\sigma_b} P_i| = k_b$ with the total length $\sum_{i=1}^{\sigma_b} ||P_i||$ minimized. We use $A[b, k_b, (s_i, t_i)_{i=1}^{\sigma_b}]$ to denote the subproblem defined above and the entry of table stores the optimal value of the subproblem.

We compute the entries of this dynamic programming from bottom to up on T. The base cases are when b is a leaf node of T. For such instances, we will show G_b has constant size therefore each such subproblem can be solved by exhaustive search in $O(1)$ time. In the recursion, for a non-leaf bag b, let b_1 and b_2 be the children bags of b in T and recall R_b be the set of edges crossing b_1 and b_2. We guess k_{b_1} and k_{b_2} for b_1 and b_2 such that $k_{b_1} + k_{b_2} = k_b$. We show how to guess start-end node pairs $\{(s_i, t_i)\}_{i=1}^{\sigma_{b_1}}$ for b_1 and $\{(s_i, t_i)\}_{i=1}^{\sigma_{b_2}}$ and b_2 that they are consistent with $\{(s_i, t_i)\}_{i=1}^{\sigma_b}$ for b : first we guess the set of edges of $P_i, 1 \leq i \leq \sigma_b$ crossing b_1 and b_2, which is a subset of R_b, denoted as E_b. For each edge in E_b we guess it is in which one of the σ_b path with start-end node pair $(s_i, t_i)_{i=1}^{\sigma_b}$ and for each path with start-end node pair (s_i, t_i) we further guess the order of the guessed edges appearing. Specifically speaking, let $e_1, e_2, \cdots, e_l$ be the edges guessed in order appearing in the path with start-end pair node (s_i, t_i). Without loss of generality, say $s_i \in V_{b_1}$ and $t_i \in V_{b_2}$. Then we set s_i and the endpoint of e_1 in V_{b_1} to be a start-end node pair in b_1, the endpoint of e_1 in V_{b_2} and the endpoint of e_2 in V_{b_2} to be a start-end node pair in b_2, $\cdots$, the endpoint of e_l in V_{b_2} and t_i to be a start-end node pair in b_2. By doing so we generate start-end node pairs for b_1 and b_2 and we sort them based on their appearing in s_i-t_i path. This defines σ_{b_1} and σ_{b_2} start-end node pairs for b_1 and b_2.

We formalize the recursion:

- for any bag $b \in T$ let b_1 and b_2 be the children bags of b.
- guess k_{b_1} and k_{b_2} for b_1 and b_2 such that $k_{b_1} + k_{b_2} = k_b$.
- let R_b be the set of edges crossing V_{b_1} and V_{b_2} and guess a subset $E_b \subset R_b$.

- for each edge in E_b, we guess it is in which of the σ_b path with start-end node pair $(s_i, t_i)_{i=1}^{\sigma_b}$ and for each path with start-end node pair (s_i, t_i) we guess the order of the edges appearing as described above. We generate $\{(s_i, t_i)\}_{i=1}^{\sigma_{b_1}}$ for b_1 and $\{(s_i, t_i)\}_{i=1}^{\sigma_{b_2}}$ for b_2.

- $A[b, k_b, (s_i, t_i)_{i=1}^{\sigma_b}] =$
  $\min_{k_{b_1}, k_{b_2}, (s_i,t_i)_{i=1}^{\sigma_{b_1}}, (s_i,t_i)_{i=1}^{\sigma_{b_2}}} A[b_1, k_{b_1}, (s_i, t_i)_{i=1}^{\sigma_{b_1}}] + A[b_2, k_{b_2}, (s_i, t_i)_{i=1}^{\sigma_{b_2}}] + \sum_{(u,v)\in E_b} d(u, v)$

The dynamic programming starts with $A[r, k, (s, t)]$ where r is the root bag in T, k and (s, t) are specified in the point to point k-TSP instance. The base case is when b is a leaf bag in T, i.e. C_b is exactly V_b, thus $|G_b| \leq \omega+1$. Note σ_b is $O(\omega^2)$ in this case because there are at most $(\omega + 1)^2$ pairs of vertices in G_b. We can enumerate all possible collections of $P_1 \cdots, P_{\sigma_b}$ such that P_i is a s_i-t_i paths. Specifically speaking, we guess all possible subset of V_b, which are $2^{\omega+1}$ many. Then for a specific subset, denoted as U, for each vertex in U we guess it is in which one of the σ_b path with source-sink pair $(s_i, t_i)_{i=1}^{\sigma_b}$ and for each path with source-sink pair (s_i, t_i) we further guess the order of guessed vertices appearing on the path, which is at most $|U|!|U|^{\sigma_b} = (\omega + 1)!(\omega + 1)^{(\omega+1)^2}$ guessings. Among these enumeration of $\{P_i\}_{i=1}^{\sigma_b}$, which is at most $O(\omega^{\omega^2})$ many, we consider the one such that $|P_1 \cup \cdots \cup P_{\sigma_b}| = k_b$ and $\sum_{i=1}^{\sigma_b} ||P_i||$ is minimized.

We show the running time of computing one entry of the dynamic programming table is at most $O(\omega^{\omega^2} n)$.

In the recursion, for bag b and its children b_1 and b_2, there are at most $k_b \leq n$ guesses for k_{b_1} and k_{b_2} such that $k_{b_1} + k_{b_2} = k_b$. For E_b: because $E_b \subset R_b$ and $|R_b| \leq (\omega + 1)^2$ there are at most $2^{(\omega+1)^2}$ guesses. To generate $(s_i, t_i), 1 \leq i \leq \sigma_{b_1}$ for b_1 and $(s_i, t_i), 1 \leq i \leq \sigma_{b_2}$ for b_2: for a certain E_b and for each edge in E_b we guess it is in which one of σ_b path with start-end node pair $\{(s_i, t_i)_{i=1}^{\sigma_b}\}$ and for each path with start-end node pair (s_i, t_i) we guess the order of the edges appearing on it, which are at most $|E_b|!|E_b|^{\sigma_b}$ guesses. Note a start-end node pair in σ_b is a pair of vertices in $V_b \cup (\{s, t\} \cap C_b)$, thus $\sigma_b \leq (\omega + 3)^2$ and therefore $|E_b|!|E_b|^{\sigma_b}$ is at most $(\omega + 1)^2!(\omega + 1)^{2(\omega+3)^2} = O(\omega^{\omega^2})$.

We bound the size of the dynamic programming table by $O(\omega^{\omega^2} n^{\rho+1})$: Recall the entry of the table is $A[b, k_b, (s_i, t_i)_{i=1}^{\sigma_b}]$. For b, there are at most $O(n^\rho)$ bags in T. For k_b, there are at most n possible value to consider. For $(s_i, t_i)_{i=1}^{\sigma_b}$, there are at most $(\omega+3)^{2(\omega+3)^2}$ start-end node pairs to consider.

Therefore, computing the dynamic programming table and finding P' a takes at most $O(\omega^{\omega^2} n^{\rho+2})$ time.

Theorem 7 *Let $G = (V, E)$ be a graph with bounded tree width ω, an integer k and start-end node pair $s, t \in V$ as an instance of point to point k-TSP on G. We can find an optimal solution in time $O(\omega^{\omega^2} n^{\rho+2})$.*

For the point to point orienteering instance on G with specified budget B and start-end node pair $s, t \in V$. We guess all possible k (from 1 to n) for the size of optimal solution P^* and for each k we compute the optimal for the point to point k-TSP instance on G with corresponding k and the same start-end node pair $s, t \in V$. We return the maximum k such that the length of the optimal for corresponding point to point k-TSP instance on G is at most B. This compelets the proof of Theorem 1.

2.2 Point to Point Orienteering on Graphs with Constant Doubling Dimension

2.2.1 Overview of the Technique

In this section we prove Theorem 2. Our starting point is to get a QPTAS for point to point k-TSP on graphs with constant doubling dimension when distance are quasi-poly bounded. The idea is built upon [10], where they present a polynomial time approximation scheme (PTAS) for rooted k-TSP on $\mathbb{R}^d$ for any fixed constant d. They show how to do in the $\mathbb{R}^2$ and extend it to the higher dimension Euclidean space. Given a graph $G = (V, E)$ on $\mathbb{R}^2$ with $n = |V|$, they introduce the notion of window: a window is a minimum bounding box on the plane containing a subset of the vertices of V. They consider a vertical or horizontal line l on the plane that cuts a windows w and introduce the notion of sparse cuts: if the number of edges of optimum path P that cross

l is at most $O(\frac{1}{\epsilon})$ then they say l is a sparse cut. If for any possible cut l for w, the number of edges of P that are crossing l is larger than $O(\frac{1}{\epsilon})$ then they call P is dense with respect to w. They show in this case the length of P in this window is at least $\Omega(\frac{1}{\epsilon}\Delta_w)$, where Δ_w is the diameter of this window. They use the idea of bridge in [16] (they use the idea of portal respecting in [2] when extending to the high dimension space $\mathbb{R}^d, d > 2$) to modify P to reduce the number of edges of P crossing the cut with a small increase in the length of P which is at most $O(\epsilon)$ portion of the length of P in w before modification by the analysis in [16]. They present a dynamic programming to find such a nearly optimal path. They define a subproblem based on a window and show the subproblem is a general version of k-TSP: a path in a window may be a set of subpaths as it may enter and exit the window many time. They call it window-TSP: the goal is to find a set of paths with specified source-sink pairs in the window with minimum total length such that they visit k vertices in total. They show the size of the dynamic programming table is polynomial in n. In the recursion, for any window w they consider all its possible cuts and for each cut l they guess the edges of P crossing l and the order of them. They generate the source-sink pairs for the subproblems defined by the windows obtained from cutting w by l. They guess the number of vertices visited in these windows. They show the total guessing in one recursion is polynomial in n.

We present a QPTAS for point to point k-TSP on graphs with constant doubling dimension by extending the result on Euclidean space to doubling metrics. To do so we need to work based on hierarchical decomposition of doubling metrics. Let $G = (V, E)$ be a graph with constant doubling dimension κ, given an integer k and start-end node pair $s, t \in V$ as an instance of point to point k-TSP on G. Let $\delta = \log \Delta_G$ and $n = |V|$. We assume the optimal P^* is known first. We introduce the notion of sparse or dense refer to considering whether the length of P^* in G is at most $\eta = \frac{\log n}{\epsilon}$ times the diameter of G. The idea of partitioning sub-instances into sparse and dense regions has been used in previous works, like in [5] to obtain a PTAS for TSP

on doubling metrics. If the length of P^* in G is at most $\eta\Delta_G$ we say P^* is sparse with respect to G. We use the idea of hierarchical decomposition in [17] to break G into $2^{O(\kappa)}$ many of subgraphs of diameter $\frac{\Delta_G}{2}$. We call this process a random partition. We show in this case the expected number of edges of P^* crossing between different subgraphs is at most $O(\frac{\kappa \log n}{\epsilon})$. If the length of P^* in G is larger than $\eta\Delta_G$ we say P^* is dense with respect to G. We use the idea of hierarchical decomposition again and use the idea of portal respecting in [17]. We break G into $2^{O(\kappa)}$ many of subgraphs of diameter $\frac{\Delta_G}{2}$. We generate a set of portals for each subgraph. We call a random partition with the portals generated in this way a random decomposition. We make P^* portal respecting, that is, we modify P^* such that it crosses between different subgraphs only through portals to reduce the number of times it crosses between different subgraphs to $O((\frac{\kappa\delta}{\epsilon})^{2\kappa})$. We show the expected increase of length of P^* in G is at most $O(\frac{\epsilon}{\delta})$ portion of the length of P^* in G before making it portal respecting in expectation. In order to get rid of the expectation, we use the notion of non-deterministic split tree. The idea of non-deterministic split tree has been used in earlier works, most recently by [11] to present approximation scheme for k-MST on minor free graphs. It is a rooted tree with alternating levels of cluster nodes and split nodes. A cluster node corresponds to a subgraph of G. A child split node of a cluster node corresponds to a decomposition of the subgraph that the cluster node corresponds to. The root of the non-deterministic split tree is the cluster node corresponding to G and each leaf of the non-deterministic split tree is a cluster node where the number of vertices of the subgraph it corresponds to is $O(1)$. For a cluster node C, its children split nodes are generated in the following way: consider $O(\log n)$ random decompositions and for each decomposition create a split node of C. For a child split node s of C, its children cluster nodes are corresponding to the subgraphs produced by applying s (the decomposition it corresponds) to C (the subpgrah it corresponds) . We show if P^* is sparse with respect to a cluster node C, then with high probability in at least one of its children split nodes, say s, the number of edges of P^* crossing different children cluster nodes of s is indeed bounded by $O(\frac{\log n}{\epsilon})$; if P is dense with respect to a cluster

node C, then with high probability in at least one of its children split nodes, say s, the increase of length of P^* in C is indeed bounded by $O(\frac{\epsilon}{\delta})$ portion of the length of P^* in C before making it portal respecting. We show the size of the non-deterministic split tree computed is quasi polynomial in n and present a dynamic programming based on the non-deterministic split tree to find a nearly optimal solution P'. We define a subproblem on a cluster node and show the subproblem is a general version of point to point k-TSP: a path in a cluster node may be a set of subpaths as it may enter and exit the cluster node many time. We call it multi-paths point to point k-TSP: the goal is to find a set of paths in a cluster node with their specified start-end node pairs with minimum total length such that they visit k vertices in total in the cluster node. In the recursion, for any cluster node C we consider all its children split nodes and for each child split node s we guess the number of vertices visited in the subproblems defined by the children cluster nodes of s and the edges of P' crossing different children cluster nodes of s. For each of edges guessed we further guess it is in which one of source-sink pair in the subproblem defined by c and for each source-sink pair we guess the ordering of the edges appearing on it. We generate start-end node pairs for the subproblems defined by children cluster nodes of s. We show the total guessing in one recursion is quasi polynomial in n.

Our next step is to show the above approximation for point to point k-TSP on graphs with constant doubling dimension actually provides a stronger bound than $(1+\epsilon)$ approximation. In [10]. They consider their approximation for rooted k-TSP on Euclidean space and introduce the notion of μ-skeleton: a μ-skeleton of a path P is a subpath of P using only μ vertices, and the notion of μ-excess based on μ-skeleton: a μ-excess of a path P is the difference between the length of P and its μ-skeleton with maximum length. They also introduce the notion of (ϵ, μ)-approximation of rooted k-TSP: a path P' such that its length is at most the length of the optimal path plus ϵ portion of μ-excess of the optimal. Note that the μ-excess of a path could be much smaller than the length of path when when μ is significantly large. Thus (ϵ, μ)-approximation

provides a better bound than $(1+\epsilon)$ approximation that we usually considered. They show the increased length only stems from the dense window and in this case optimal path has large μ-excess such that the increased length of optimal can be upper bound by a factor the μ-excess of the optimal. Thus they show their approximation for rooted k-TSP is actually a (ϵ, μ)-approximation for $\mu = O(\frac{1}{\epsilon})$ on Euclidean space. We use the same definition of μ-skeleton, μ-excess and (ϵ, μ)-approximation. We use a different way of proof to extend their result to shows our approximation for point to point k-TSP on graphs with constant doubling dimension is actually a (ϵ, μ)-approximation for $\mu = O(\frac{1}{\epsilon})$ as well.

The last step is to convert a (ϵ, μ)-approximation for point to point k-TSP on graphs with constant doubling dimension to a $(1 + \epsilon)$ approximation for point to point orienteering problem on graphs with constant doubling dimension. In [10], they convert a (ϵ, μ)-approximation for rooted k-TSP on Euclidean space to a $(1 + \epsilon)$ approximation for rooted orienteering on Euclidean space. They assume the optimal value of orienteering problem k is known first and show how to get rid of this assumption by guessing all possible value of k. They show for the optimal of orienteering problem, which visits k vertices of length at most B, one can shortcut a certain ϵ portion of the path to get a new path which visits only $k' = (1 - \epsilon)k$ vertices. They consider this new path as an instance of rooted k'-TSP and find a (ϵ, μ)-approximation of it. They show this approximation visits at least k' vertices and the length of it is at most B thus it's a $(1 - \epsilon)$ approximation for rooted orienteering. We extend their result to show converting a (ϵ, μ)-approximation for point to point k-TSP to $(1 + \epsilon)$ approximation for point to point orienteering problem in the case of graphs with constant doubling dimension.

2.2.2 QPTAS for Point to Point k-TSP on Graphs with Constant Doubling Dimension

Let $G = (V, E)$ be a graph with constant doubling dimension κ, and given an integer $k > 0$ and start and end node pair $s, t \in V$ as an instance of point to point k-TSP on G. Let $n = |V|$ and $\delta = \log \Delta_G$, note δ is poly logarithmic in

the case of Δ_G is quasi polynomial. We show how to get a $(1+\epsilon)$ approximation for the point to point k-TSP instance in quasi polynomial time. We assume the optimal P^* is known first.

For any path P and any subgraph $G' \subset G$, let $P \cap G'$ be P restricted to G'. We compare the length of P in G', i.e. $||P \cap G'||$, with the diameter of G' and introduce the notion of dense and sparse.

Definition 2 *P is called η-dense with respect to G' if $||P \cap G'|| \geq \eta\Delta_{G'}$. Otherwise we say P is sparse with respect to G'.*

We will set $\eta = \frac{\log n}{\epsilon}$ and consider whether P^* is η-dense or sparse with respect to G. We introduce the notions of cover and net before describing how to handle the sparse case or the dense case.

Definition 3 *Given a weighted graph $G = (V, E)$ and $r > 0$, $N \subset V$ is a r-cover of G if for any vertex $v \in V$, v is in a ball centered at some vertex x in N of radius r, i.e. $v \in B(x, r)$ for some $x \in N$. Furthermore, N is called a r-net of G if:*

- *N is a r-cover of G*
- *For any $u, v \in N$ that $u \neq v$, $d(u, v) > r$, in other words vertices in N are well separated.*

The following lemma shows one can generate a r-net for G efficiently by greedily:

Lemma 1 *For given $r > 0$, we can find a r-net for G in time $O(n^2)$.*

Proof. We obtain a r-cover for G greedily: initially we set every vertex in V uncovered, then we iteratively pick an uncovered vertex w at random. We consider the ball $B(w, r)$ and let $B_w = \{v \in B(w, r) : v \text{ is uncovered yet}\}$. We set all vertices in B_w covered. Note for one iteration at least one uncovered vertex gets covered thus there are at most n iterations. At each iterations it takes time $O(n)$ in total to pick the uncovered vertex w, generate B_w and set

all vertices in B_w covered. Let N be the set containing all w picked in the iterations. Note $\{B_w\}_{w\in N}$ forms a partition of G, i.e. they are disjoint and the union of them is G. Notice that if $w_1, w_2 \in N$ then $d(w_1, w_2) > r$. To see this, without loss of generality assume that w_1 is picked before w_2 in N , then $w_2 \in B(w_1, r)$ thus w_2 should not appear in N. This implies for any $w_1, w_2 \in N$ then $d(w_1, w_2) > r$. Hence N is a r-net. □

We consider a r-net of G where $r = \frac{\Delta_G}{4}$, denoted as N. By the packing property in [17], $|N| \leq 4^\kappa$. Recall $\{B_w\}_{w\in N}$ is a partition of G. For each B_w, we further consider a $\beta\Delta_{B_w}$-net of it $\{B_w\}_{w\in N}$ where $\beta = \frac{\epsilon}{2\kappa'\delta}$ (where $\delta = \log \Delta_G$ and κ' will be defined soon), denoted as S_w. By packing property in [17], the number of portals for B_w, i.e. $|S_w|$, is at most $2^{\kappa\lceil \log \frac{2}{\beta}\rceil}$ which is $(\frac{\kappa'\delta}{\epsilon})^{2\kappa}$. We call $\{B_w\}_{w\in N}$ with $\{S_w\}_{w\in N}$ a random decomposition of G, denoted as π_G. We say an edge (u, v) crossing π_G if u and v are in different B_w for $w \in N$, i.e. $u \in B_{w_1}$ and $u \in B_{w_2}$ for $w_1, w_2 \in N$ that $w_1 \neq w_2$. We say P is portal respecting with respect to π_G if for any edge (u, v) of P crossing π_G, it only cross through portals, i.e. $u \in S_{w_1}$ and $v \in S_{w_2}$, for $w_1, w_2 \in N$ that $w_1 \neq w_2$.

The following lemma from [17] shows one property of edges crossing π_G :

Lemma 2 *For any edge $(u, v) \in E$, the probability that (u, v) crosses π_G is at most $\kappa'\frac{d(u,v)}{\Delta_G}$ for some constant $\kappa' = O(\kappa)$.*

If P^* is sparse with respect to G, then we consider the set of edges of P^* crossing π_G, denoted as E_{π_G}, i.e. $E_{\pi_G} = \{(u, v) \in P^* : (u, v) \text{ crosses } \pi_G\}$. We can upper bound the expected size of E_{π_G} in this case:

Lemma 3 *If P^* is sparse with respect to G, then $\mathbb{E}(|E_{\pi_G}|) \leq \frac{\kappa' \log n}{\epsilon}$ for some constant $\kappa' > 0$.*

Proof. Recall the probability that (u, v) crosses π_G is at most $\kappa'\frac{d(u,v)}{\Delta_G}$. Therefore, $\mathbb{E}(|E_{\pi_G}|) = \sum_{(u,v)\in P} \kappa'\frac{d(u,v)}{\Delta_G} = \kappa'\frac{||P_G||}{\Delta_G}$. Since P^* is sparse with respect to G, thus $\kappa'\frac{||P_G||}{\Delta_G} \leq \kappa'\frac{\eta\Delta_G}{\Delta_G} = \kappa'\eta = \kappa'\frac{\log n}{\epsilon}$. □

If P^* is η-dense with respect to G, the following lemma shows one can modify P^* to be portal respecting with respect to π_G with a small increase of length.

Lemma 4 *P^* can be modified to a path P' that is portal respecting with respect to π_G such that $\mathbb{E}(||P'||) = (1 + \frac{\epsilon}{2\delta})||P^* \cap G||$.*

Proof. Consider any edge (u, v) in P^* that crosses π_G, say $u \in B_{w_1}$ and $v \in B_{w_2}$, for $w_1, w_2 \in N$ that $w_1 \neq w_2$. Let u' be the nearest portal to u in S_{w_1}, i.e. $u' = \arg\min_{w \in S_{w_1}} d(w, u)$ and v' be the nearest portal to v in S_{w_2}, i.e. $d(v', v) = \arg\min_{w \in S_{w_2}} d(w, v)$. Replace (u, v) in P^* by the edges (u, u'), (u', v'), (v', v). The increased length incurred is $d(u, u') + d(v, v') + d(u', v') - d(u, v)$, which is at most $2d(u, u') + 2d(v, v')$ by triangle inequality. Note that because S_w is a $\beta\Delta_{B_w}$-net of B_w then $d(u, u') \leq \beta\Delta_{B_{w_1}} \leq \beta\frac{\Delta_G}{4}$ and $d(v, v') \leq \beta\Delta_{B_{w_2}} \leq \beta\frac{\Delta_G}{4}$. Thus the increased length incurred is at most $\beta\Delta_G$. Recall that for (u, v) in P, the probability that it crosses π_G is at most $\kappa'\frac{d(u,v)}{\Delta_G}$. Let P' be the path after modifying every such (u, v) in P^*. Thus in expectation the increase length of P^* after making it portal respecting with respect to π_G is at most $\sum_{(u,v)\in P^*\cap G} \kappa'\frac{d(u,v)}{\Delta_G}\beta\Delta_G = \kappa'\beta||P^*\cap G|| = \frac{\epsilon}{2\delta}||P^*\cap G||$. □

Non-deterministic Split Tree

Imagining we adapt a hierarchical decomposition T to G as follows: start from the single cluster $\{V\}$ (as the root of T) and at each step uses a random decomposition as described in the precious section to the current cluster C to decompose it into 4^κ clusters of diameter half the size. This continues until we arrive at clusters that have constant size, say at most a for some $a > 0$.

Starting from the root of T, and initially set $P' = P^*$. We modify P' as we go down the split tree T: for each node $C \in T$, if P' is sparse with respect to $G(C)$ we make no modification going down to children of C. If P' is η-dense with respect to $G(C)$, we make P' portal respecting with respect to the partition that C corresponds to. Note both Lemma 3 and Lemma 4 hold

in expectation in $G(C)$. We introduce the notion of non-deterministic split tree to get rid of the expectation. We consider multiple random decompositions independently at each step. Then for a cluster node C, if P' is sparse with respect to $G(C)$, we show with high probability at least in one of these decompositions $\pi_{G(C)}$, $|E_{G(C)}|$ is indeed close to the expected value $\kappa' \frac{\log n}{\epsilon}$. If P' is η-dense with respect to $G(C)$, we show with high probability at least in one of decompositions $\pi_{G(C)}$, the increased length of P' after making it portal respecting with respect to $\pi_{G(C)}$ is indeed closed to the expected value $\frac{\epsilon}{2\delta}||P' \cap G(C)||$.

Definition 4 *A γ-nondeterministic split tree for G is a rooted tree Γ with alternating levels of cluster nodes and split nodes.*

- *a cluster node C corresponds a subgraph $G(C) \subset G$. Each non-leaf cluster node has at most γ split nodes as its children. The root of Γ, denoted as C_0, is trivial cluster node corresponding to G and each leaf node C is a cluster node such that the number of vertices in $G(C)$ is at most some constant $a > 0$.*

- *A split node s corresponds to a random decomposition π_s of $G(C)$.*

Note if we start from root cluster node C_0 of Γ and apply the following: for each cluster node C we encounter, pick one of the children split node of C, say s, and apply π_s to $G(C)$, then repeat this process as going down Γ. The collection of cluster nodes selected corresponds to a valid hierarchical decomposition of G.

We introduce the notion of forcing to formalize the process described above. Essentially the forcing removes the nondeterministic part of nondeterministic split tree γ by mapping from a non-leaf cluster C to at most one of its children split node s.

Definition 5 *Given a γ-nondeterministic split tree Γ, a forcing ψ is a partial function from some cluster nodes of Γ to their children split nodes of γ such that:*

- *$\psi(C_0) = s$ for some child split node of C_0.*
- *if every ancestor split node of C is in the image of ψ then $\psi(C)$ is defined.*

Note a forcing induces a valid hierarchical decomposition split-tree T as follows: start at the root node C_0 of Γ, let C_0 be the root of T and repeat the following procedure: at each step, being at a cluster node C pick $s = \psi(C)$ (which is a child split node of C) and consider the cluster children of s, say $C_1, \cdots, C_g$ and let them be children nodes of C in T. Recursively repeat the procedure from each of them. This builds a split-tree tree T . We say T is induced by ψ on Γ: $T = \Gamma|_\psi$.

The following lemma shows one can compute a $O(\log n)$-nondeterministic split tree for G in time quasi-polynomial in n.

Lemma 5 *We can compute a $3 \log n$-nondeterministic split tree for G.*

PROOF. We start by making C_0 to be the root of Γ where $G(C_0) = G$ and iteratively add levels to Γ. For a cluster node C, we generate its children split nodes as follows:

We compute $3 \log n$ independent random decompositions for $G(C)$, denoted as $\{\pi_i\}$: to generate each π_i, we consider a $\frac{\Delta_{G(C)}}{4}$-net for $G(C)$, denoted as N_i. Let $\{B_w\}_{w \in N_i}$ be the partition of $G(C)$ and let R_{π_i} be the set of edges in $G(C)$ crossing π_i, i.e. $R_{\pi_i} = \{(u, v) \in G(C) : u \in B_{w_1}, v \in B_{w_2}, \text{ for } w_1, w_2 \in N_i \text{ that } w_1 \neq w_2\}$. For each B_w, we further consider the portal set for B_w. i.e. a $\beta\Delta_{B_w}$-net of it, denoted as S_w. Let R'_{π_i} be the set of edges in $G(C)$ crossing π_i only through $\{S_w\}$ i.e. $R'_{\pi_i} = \{(u, v) \in G(c) : u \in S_{w_1}, v \in S_{w_2}, \text{ for } w_1, w_2 \in N_i \text{ that } w_1 \neq w_2\}$. $|R'_{\pi_i}|$ is at most $(|N||S_w|)^2$ which is $(\frac{2\kappa'\delta}{\epsilon})^{4\kappa}$. We create a child split node s_i for each π_i.

For a split node s, let C be its parent cluster node and let $\{B_w\}_{w \in N}$ be the partition where N is a $\frac{\Delta_{G(C)}}{4}$-net for $G(C)$. Then for each $w \in N$ we create a child cluster node C_w of s corresponding to B_w. The number of the children cluster nodes of s is $|N|$ which is at most 4^κ.

Let C be any cluster node on γ, s be a child split node of C and C' be a child cluster node of s. From the construction above we know $\Delta_{G(C')} \leq \frac{\Delta_{G(C)}}{2}$.

Thus there are at most $2\log\Delta_G = 2\delta$ levels in Γ. If we define the height of Γ as the number of levels of cluster nodes then the height of Γ is at most δ. The branching factor of Γ is then the product of the branching factor of a cluster node and the branching factor of a split node, which is at most $3\log n 4^{\kappa}$. Hence the size of Γ is at most $(3\log n 4^{\kappa})^{\delta}$.

□

Structure Theorem and Dynamic Programming

The goal of the structure theorem is to show with high probability there exists a forcing ψ on Γ computed in Lemma 5, such that there is a nearly optimal solution P' that has good structural properties on the induced hierarchical decomposition $\Gamma|_{\psi}$. For a cluster node C in $\Gamma|_{\psi}$ and the split node defined by $\psi(C)$, we use $|P \cap R_{\pi_{\psi(C)}}|$ to denote the number of edges in P crossing the decomposition $\pi_{\psi(C)}$ in $G(C)$ and $|P \cap R'_{\pi_{\psi(C)}}|$ to denote the number of edges in P crossing the decomposition $\pi_{\psi(C)}$ only through portals in $G(C)$.

Theorem 8 *Let $G = (V, E)$ be a graph with constant doubling dimension κ, given an integer $k > 0$ and start and end node pair $s, t \in V$ as an instance of point to point k-TSP of G. Assume P^* is an optimal solution for the instance. Then for the $3\log n$-nondeterministic split tree Γ we compute in Lemma 5, with probability $1 - \frac{1}{n}$ there exists a forcing ψ on Γ and a nearly optimal solution P', such that P' visits at least k vertices, and for any cluster C in the induced hierarchical decomposition $T = \Gamma|_{\psi}$, we have either:*

- $|P' \cap R_{\pi_{\psi(C)}}| \leq 2\kappa' \frac{\log n}{\epsilon}$.
- $|P' \cap R_{\pi'_{\psi(c)}}| \leq (\frac{2\kappa'\delta}{\epsilon})^{4\kappa}$ *and* $||P' \cap C|| \leq (1+\epsilon)||P^* \cap C||$.

Proof.

We build P' iteratively based on P and at the same time build $\psi(\cdot)$ (hence T) from top to down in Γ: initially we set P' to be P^* and start from the root cluster node C_0 of T. At any point when we are at a cluster node C consider P' is sparse or η-dense with respect to C:

If P' is sparse with respect to C, then we don't modify $P' \cap C$ when going down from C to any split node of C. Consider any child split node s of C and let π_s be the corresponding decomposition. We consider the edges of P' crossing the decomposition π_s, which is a subset of R_{π_s}. According to Lemma 3, $\mathbb{E}(|P' \cap R_{\pi_s}|) \leq \kappa' \frac{\log n}{\epsilon}$. Let F_s be the event that $|P' \cap R_{\pi_s}|$ is at most $2\kappa' \frac{\log n}{\epsilon}$, by Markov inequality $\Pr[F_s \text{ happens }] \geq \frac{1}{2}$. Recall there are $3 \log n$ many children split nodes of c, i.e. $3 \log n$ independent random decompositions for $G(c)$, thus the probability that for at least one child split node s of c event F_s happens is at least $1 - (1 - \frac{1}{2})^{3 \log n} \geq 1 - \frac{1}{n^3}$.

If P' is η-dense with respect to c, then consider any child split node s of C and let π_s be the corresponding decomposition. We modify P' to be portal respecting with respect to π_s as described in Lemma 4: let N be the $\frac{\Delta_{G(C)}}{4}$-net for $G(C)$, $\{B_w\}_{w \in N}$ be the partition and $\{S_w\}_{w \in N}$ be the portal sets, we make P' crossing π_s only through $\{S_w\}$. Note the set of edges of P' crossing π_s after making it portal respecting with respect to π_s is a subset of R'_{π_s}. Thus $|P' \cap R'_{\pi_s}|$ is at most $|R'_{\pi_s}|$ which is at most $(\frac{2\kappa'\delta}{\epsilon})^{4\kappa}$ in this case. According to Lemma 4, the increase of length of P' after making it portal respecting with respect to π_s is at most $\frac{\epsilon}{2\delta}||P' \cap C||$ in expectation. Let F'_s be the event that the increase of length of P' after making it portal respecting with respect to π_s is at most $\frac{\epsilon}{\delta}||P' \cap C||$, by Markov inequality, $\Pr[F'_s \text{ happens}] \geq \frac{1}{2}$. Recall there are $3 \log n$ many children split nodes of C, i.e. $3 \log n$ independent random decompositions for $G(C)$, thus the probability that for at least one child split node s of C event F'_s happens is at least $1 - (1 - \frac{1}{2})^{3 \log n} \geq 1 - \frac{1}{n^3}$.

Therefore, regardless of whether P' is sparse or η-dense with respect to C such s exists with the probability at least $1 - \frac{2}{n^3}$ for C and we define $\psi(C) = s$. Once we have $\psi(\cdot)$ defined for clusters at a level of Γ, we have determined the clusters at the same level of $T = \Gamma|_\psi$. Note there are at most n cluster nodes in one level of Γ, thus with probability at least $1 - \frac{2}{n^2}$ such split nodes exist for all cluster nodes in one level. Since the height of Γ is at most δ, thus with probability at least $(1 - \frac{2}{n^2})^\delta \geq 1 - \frac{1}{n}$ (recall that δ is polylogarithmic in n) such $\psi(\cdot)$ is well defined over all levels.

Note the increase of length of P' only occurs when P' is η-dense with respect

to C for cluster nodes in $T = \Gamma|_\psi$ and the increase of length P' after modifying P' to be portal respecting with respect to $\pi_{\psi(C)}$ is at most $\frac{\epsilon}{\delta}||P' \cap C||$. Since this $(1 + \frac{\epsilon}{\delta})$-factor increase occurs only when we go down T from a cluster C to the next cluster level down and the height of T is at most δ, thus for any cluster $C \in T$: $||P' \cap C|| \leq (1 + \frac{\epsilon}{\delta})^\delta ||P \cap C|| \leq e^\epsilon ||P \cap C|| \leq (1 + \epsilon')||P \cap C||$ for some $\epsilon' = O(\epsilon)$. Replacing ϵ' with ϵ we get $||P' \cap C|| \leq (1 + \epsilon)||P \cap C||$.

□

We show a proper dynamic programming that can find a desired forcing ψ and a nearly optimal solution P' on the induced hierarchical decomposition $\Gamma|_\psi$ with the properties described above in Theorem 8. The dynamic programming is built on the non-deterministic split tree Γ we compute.

For a cluster node C in Γ, consider P' in the subgraph $G(C)$. It may enter and exit $G(c)$ multiple times. Hence P' in $G(C)$ may be a collection of disjoint paths. We use the same notion of multi-path k-TSP in Definition 1 introduced in the case of point to point orienteering on graphs with bounded treewidth to define the subproblems in the dynamic programming. We define a subproblem in the dynamic programming as an instance of multi-paths k-TSP with specified cluster node C, integer k_C, σ_C start-end node pairs $(s_i, t_i), 1 \leq i \leq \sigma_C$ and the goal is to find a set of paths $\{P_i\}_{i=1}^{\sigma_C}$ such that P_i is a s_i-t_i path in $G(c)$ and $|\cup_{i=1}^{\sigma_C} P_i| = k_C$ with minimized $\sum_{i=1}^{\sigma_C} ||P_i||$. We use $A[C, k, (s_i, t_i)_{i=1}^{\sigma_C}]$ to denote the subproblem defined above and let the entry of the table store the optimal value of the subproblem. We will show σ_C is poly-logarithmic for any cluster node C thus the total number of the subproblems is bounded by quasi-polynomial in n.

The base cases are when the cluster C has constant size. We will show such instances can be solved using exhaustive search in $O(1)$ time. In the recursion, consider an arbitrary entry $A[C, k_C, (s_i, t_i)_{i=1}^{\sigma_C}]$ where for all split nodes children of C and every cluster children of them the entries of the table are computed. Consider any child split node s of C in Γ. Let $c_1, c_2, \cdots c_g$ be the children cluster nodes of s in γ. Recall π_s is the corresponding decomposition of $G(C)$ and R_{π_s} is the set of edges in $G(c)$ crossing π_s. For each C_j let S_j be the portal

set of $G(C_j)$ and recall R'_{π_s} is the set of edges crossing π_s only through $\{S_j\}$. We guess k_{C_j} for each C_j such that $\sum_{j=1}^{g} k_{C_j} = k_C$. We show how to guess start-end node pairs $\{(s_i, t_i)_{i=1}^{\sigma_{C_j}}\}$ for each C_j and check the consistency of them: we consider two cases, we guess a subset of R_{π_s} of size at most $2\kappa' \frac{\log n}{\epsilon}$ (meaning we assume we are in the sparse case); we guess a subset of R'_{π_s} (meaning we assume we are in the dense case). Let E_{π_s} be the subset guessed in either case. Furthermore for each edge in E_{π_s}, we guess it is in which one of the σ_C paths with start-end node pair $(s_i, t_i)_{i=1}^{\sigma_C}$ and for each path with source-sink pair (s_i, t_i) we guess the order of the guessed edges appearing on the path. Specifically speaking, let $e_1, e_2 \cdots, e_l$ be the edges guessed in the path with start-end node pair (s_i, t_i) appearing in this order. Let $C_{a_1}, C_{a_2}, \cdots, C_{a_{l+1}}$ be the children cluster nodes of s that the path encounters following $e_1, e_2 \cdots, e_l$, i.e. e_1 crosses between C_{a_1} and C_{a_2}, e_2 crosses between C_{a_2} and c_{a_3}, $\cdots$, and e_l crosses between C_{a_l} and $C_{a_{l+1}}$. Then we set s_i and the endpoint of e_1 in C_{a_1} to be a start-end node pair in C_{a_1}, the endpoint of e_1 in C_{a_2} and the endpoint of e_2 in C_{a_2} to be a start-end node pair in C_{a_2}, $\cdots$, the endpoint of e_l in $C_{a_{l+1}}$ and t_i to be a start-end node pair in $C_{a_{l+1}}$. By doing so we generate start-end node pairs for each C_j and we sort them based on their appearing in s_i-t_i path. This defines σ_{C_j} start-end node pairs for each C_j.

We formalize the recursion:

- Consider any child split node s of c, let $C_1, C_2, \cdots, C_g$ be the children cluster nodes of s.
- Guess k_j for each C_j such that $\sum_{j=1}^{g} k_{C_j} = k_C$.
- Let π_s be the corresponding decomposition and R_{π_s} be the set of edges crossing π_s in $G(C)$. For each C_j let S_j be the portal set for $G(C_j)$ and let R'_{π_s} be the set of edges crossing π_s through $\{S_j\}$ in $G(C)$. We consider both of the following two cases: guess a subset of R_{π_s} of size at most $2\kappa' \frac{\log n}{\epsilon}$; we guess a subset of R'_{π_s}. In both cases we denote the set of guessed edges as E_{π_s}.
- For each edge in E_{π_s}, we guess it is in which one of the σ_C paths with

start-end node pairs $(s_i, t_i)_{i=1}^{\sigma_C}$ and for each path with start-end node pair (s_i, t_i) we guess the order of the guessed edges appearing as described above. We generate $\{(s_i, t_i)\}_{i=1}^{\sigma_{C_j}}$ for each C_j accordingly. Then:

- $A[c, k, (s_i, t_i)_{i=1}^{\sigma_c}] = \min_{s, k_{c_1}, \cdots, k_{c_g}, (s_i,t_i)_{i=1}^{\sigma_{c_1}}, \cdots, (s_i,t_i)_{i=1}^{\sigma_{c_g}}} \sum_{j=1}^{g} A[c_j, k_j, (s_i, t_i)_{i=1}^{\sigma_{c_j}}] + \sum_{(u,v)\in E_{\pi_s}} d(u, v)$.

The dynamic programming starts with $A[C_0, k, (s, t)]$ where k and (s, t) are specified in the point to point k-TSP instance. The base case is when C is a leaf cluster node in Γ, i.e. $|V(C)| = a$ for some constant $a > 0$ where $V(C)$ is the vertices set of $G(C)$. Note σ_C is at most a^2 in this case because $s_i, t_i \in V(C)$. We can enumerate all possible collections of $\{P_i\}_{i=1}^{\sigma_C}$ such that P_i is a s_i-t_i paths. Specifically speaking, we guess all possible subset of $V(C)$, which are at most 2^a many. Then for a specific set, denoted as U, for each vertex in U we guess it is in which one of the σ_C path with start-end node pairs $(s_i, t_i)_{i=1}^{\sigma_C}$. For each path with start-end node pair (s_i, t_i) we further guess the order of guessed vertices appearing on the path, which is at most $|U|!|U|^{\sigma_C} = a!a^{a^2}$ guessings. Among these enumeration of $\{P_i\}_{i=1}^{\sigma_C}$, which is at most $O(a^{a^2})$ many, we consider the one such that $|P_1 \cup \cdots \cup P_{\sigma_C}| = k_C$ with minimized $\sum_{i=1}^{\sigma_C} ||P_i||$.

We show the running time of computing one entry of the dynamic programming table is at most $n^{O((\frac{\delta}{\epsilon})^{2\kappa+1})}$.

In the recursion, for a cluster node C, there are $3 \log n$ children split nodes of C in Γ to consider. For a certain split node s, let $c_1, c_2, \cdots, c_g$ be children cluster nodes of s, there are at most n^g guesses to for $\{k_{C_j}\}$ such that $\sum_{j=1}^{g} k_{C_j} = k_C$, which is at most $n^{2^{O(\kappa)}}$ because $g \leq 2^{O(\kappa)}$. For E_{π_s}: there are two cases, if $E_{\pi_s} \subset R_{\pi_s}$ such that $|E_{\pi_s}| \leq 2\kappa' \frac{\log n}{\epsilon}$, there are at most $n^{(4\kappa' \frac{\log n}{\epsilon}+1)}$ many possible E_{π_s} to consider; if $E_{\pi_s} \subset R'_{\pi_s}$, then because $|R'_{\pi_s}| \leq (\frac{2\kappa'\delta}{\epsilon})^{4\kappa}$ in this case there are at most $2^{(\frac{2\kappa'\delta}{\epsilon})^{4\kappa}} \leq n^{(\frac{2\kappa'\delta}{\epsilon})^{4\kappa}}$ many possible E_{π_s} to consider. To generate $(s_i, t_i)_{i=1}^{\sigma_{C_j}}$ for each C_j: for a certain E_{π_s} and for each edge in E_{π_s} we guess it is in which one of σ_C path with start-end node pair $(s_i, t_i)_{i=1}^{\sigma_C}$ and for each path with start-end node pair (s_i, t_i) we guess the order of the edges appearing, which is at most $|E_{\pi_s}|!|E_{\pi_s}|^{\sigma_C}$ guessings. Note at

each recursion it may increase at most $|E_{\pi_s}|$ number of start-end node pairs and the depth of the recursion is δ. Thus $\sigma_C \leq \delta|E_{\pi_s}|$ and $|E_{\pi_s}|!|E_{\pi_s}|^{\sigma_C}$ is $(\frac{2\kappa'\delta}{\epsilon})^{4\kappa}!(\frac{2\kappa'\delta}{\epsilon})^{4\kappa\delta(\frac{2\kappa'\delta}{\epsilon})^{4\kappa}} \leq n^{O((\frac{\delta}{\epsilon})^{4\kappa+1})}$.

We show the size of the dynamic programming table is at most at most $n^{O((\frac{\delta}{\epsilon})^{4\kappa+1})}$: Recall the entry of the table is of form $A[C, k_C, (s_i, t_i)_{i=1}^{\sigma_C}]$. For C, there are at most $(3\log n \cdot 4^{\kappa})^{\delta}$ cluster nodes in Γ because the size of Γ is at most $(3\log n \cdot 4^{\kappa})^{\delta}$. For k_C, there are at most n possible value of k_C to consider. For $\{s_i, t_i\}_{i=1}^{\sigma_C}$, there are at most $n^{2\sigma_C}$ start-end node pairs to consider, which is at most $n^{2\delta(\frac{2\kappa'\delta}{\epsilon})^{4\kappa}}$ because σ_C is at most $\delta|E_{\pi_s}|$.

Therefore, computing the dynamic programming table and finding P' as in Theorem 8 takes at most $n^{O((\frac{\delta}{\epsilon})^{4\kappa+1})}$ time.

Theorem 9 *Let $G = (V, E)$ be a graph with a constant doubling dimension κ, given an integer $k > 0$ and start-end node pair $s, t \in V$ as an instance of point to point k-TSP on G, with probability at least $1 - \frac{1}{n}$ we can get a $(1+\epsilon)$-approximation for this instance in time $n^{O((\frac{\delta}{\epsilon})^{4\kappa+1})}$.*

2.2.3 (ϵ, μ)-approximation for Point to Point k-TSP on Graphs with Constant Doubling Dimension

We show a stronger bound on the length of the near optimum solution P' guaranteed by Theorem 8. We formalize the notion of excess first:

Definition 6 *Let $P = \langle v_1, v_2, \cdots, v_k \rangle$ be a path which visits k vertices. Let $1 = i_1 < i_2 < \cdots < i_\mu = k$ be a sub-sequence of indices. We say the path $\langle v_{i_1}, v_{i_2}, \cdots, v_{i_\mu} \rangle$ is a μ-skeleton of P. The optimal μ-skeleton of P is the μ-skeleton with maximum length. The μ-excess of P is the difference between the length of P and its optimal μ-skeleton, denoted as $\xi_{P,\mu}$, i.e. $\xi_{P,\mu} = ||P|| - \max_{i_1, i_2, \cdots, i_u} ||\langle v_{i_1}, v_{i_2}, \cdots, v_{i_u} \rangle||$.*

We give a formal definition of (ϵ, μ)-approximation based on μ-excess:

Definition 7 *Let $G = (V, E)$ be a graph, given an integer $k > 0$ and start and end node pair $s, t \in V$ as an instance of a point to point k-TSP on G. For any $\epsilon > 0$ and integer $\mu > 0$, a path P is a (ϵ, μ)-approximation for the point*

to point k-TSP instance if it is a s-t path and visits at least k vertices with the length at most $||P^*|| + \epsilon\xi_{P^*,\mu}$ *where* P^* *is the optimal for the instance.*

We show the path P' in Theorem 8 is in fact a (ϵ, μ) approximation for the given point to point k-TSP instance in G where $\mu = \lceil \frac{1}{\epsilon} \rceil + 1$.

Recall in the proof of Theorem 8, the increased length of P' only stems from the case that P' is η-dense with respect to C in Γ and we make it portal respecting. Consider such a dense cluster C in the hierarchical decomposition $T = \Gamma|_\psi$, i.e. P' is η-dense with respect to C. We show P' has high μ-excess such that the increased length of P' in C can be upper bounded by a factor of the μ-excess of P^*.

Lemma 6 *Let* D *be a set of disjoint clusters in* $\Gamma|_\psi$ *and* P *be a path. Then* $\xi_{P,2} \geq \sum_{C \in D}(||P \cap C|| - \Delta_{G(C)})$.

Proof. Let P_0 be the path just connecting the startig and ending node of P. By definition of the excess, $\xi_{P,2} = ||P|| - ||P_0||$. Then consider the following path $\overline{P}$, which starts at the same node as P and follows route of P but when it encounters a cluster C in D and it visits C for the first time then it directly connects start and end node of the subpath of P in C. When it encounters a cluster C in D that is visited before, then bypasses C entirely, i.e. directly connect the last vertex in P before it enters C this time and the first vertex in P after it exits C this time. From the construction of $\overline{P}$, if $C \in D$, then $||\overline{P} \cap C|| \leq \Delta_{G(C)}$. If $C \notin D$, then $||\overline{P} \cap C|| = ||Q \cap C||$. Thus $\xi_{P,2} = ||P|| - ||P_0|| \geq ||P|| - ||\overline{P}|| = \sum_{C \in D}(||P \cap C|| - ||\overline{P} \cap C||) \geq \sum_{C \in D}(||P \cap C|| - \Delta_{G(C)})$. □

Theorem 10 *Let* $G = (V, E)$ *be a graph with constant doubling dimension* κ*, given an integer* $k > 0$*, start and end node pair* $s, t \in V$ *as an instance of a point to point k-TSP on* G*. Suppose* P' *and* $\Gamma|_\psi$ *are as guaranteed in Theorem 8. Let* $\mu = \lceil \frac{1}{\epsilon} \rceil + 1$*, then* P' *is a* (ϵ, μ)*-approximation for the point to point k-TSP instance.*

PROOF.

We generate a set of disjoint dense clusters D in $\gamma|_\psi$: we start from C_0 to generate D iteratively. If P' is η-dense with respect to C then return C. If P' is sparse with respect to C and C is a non-leaf cluster node in $\gamma|_\psi$, then iteratively consider all children cluster node of $\psi(C)$. Let D be the set of clusters returned by this process. Without loss of generality, D is not empty set (otherwise $P' = P^*$). From the construction of D, clusters in D are disjoint and for each cluster $C \in D$, P' is η-dense with respect to C and $||P'|| - ||P^*|| = \sum_{C\in D}(||P' \cap C|| - ||P^* \cap C||)$.

Let $\{v_1, \cdots, v_\mu\}$ be the optimal μ-skeleton of P^* and let $P_1^*, P_2^*, \cdots, P_{\mu-1}^*$ be the subpaths divided by the vertices in this μ-skeleton, i.e. P_i^* is the subpath of P^* whose start and end node pair are v_i and v_{i+1}. By definition of excess, $\xi_{P^*,\mu} = \sum_{i=1}^{\mu-1} \xi_{P_i^*,2}$. For the set D and each P_i^*,by Lemma 6, $\xi_{P_i^*,2} \geq \sum_{C\in D}(||P_i^* \cap C|| - \Delta_{G(C)})$. Thus $\xi_{P^*,\mu} \geq \sum_{i=1}^{u-1}\sum_{C\in D}(||P_i^* \cap C|| - \Delta_{G(C)})$, which is $\sum_{C\in D}(||P^* \cap C|| - (\mu-1)\Delta_{G(C)})$. Recall from the proof of Theorem 8 we modify P' when it is η-dense with respect to C ($||P' \cap C|| \geq \frac{\log n}{\epsilon}\Delta_{G(C)}$) and we always have $||P' \cap C|| \leq (1+\epsilon)||P^* \cap C||$, which implies $||P^* \cap C|| \geq \frac{\log n}{\epsilon(1+\epsilon)}\Delta_{G(C)}$. Since $\mu = \lceil \frac{1}{\epsilon} \rceil + 1$, thus $||P^* \cap C|| - (\mu-1)\Delta_{G(C)} \geq \frac{||P^*\cap C||}{2\epsilon}$ and $\xi_{P^*,\mu} \geq \sum_{C\in D} \frac{||P^*\cap C||}{2\epsilon}$.

As in proof of Theorem 8 for any cluster C in $\Gamma|_\psi$, $||P'\cap C|| \leq (1+\epsilon)||P^* \cap C||$. Thus $||P'|| - ||P^*|| = \sum_{C\in D}(||P' \cap C|| - ||P^* \cap C||) \leq \sum_{C\in D} \epsilon||P^* \cap C|| \leq 2\epsilon^2\xi_{P^*,\mu} \leq \epsilon\xi_{P^*,\mu}$. □

Note that we can generalize this proof slightly as follows. Suppose that we pick some arbitrary μ-skeleton of P^* (instead of the optimum μ-skeleton) and consider $\tilde{P}_1^*, \cdots, \tilde{P}_{\mu-1}^*$ which are the subpaths of P^* defined by that μ-skeleton. Then the same arguments show that $\sum_{i=1}^{\mu-1} \xi_{\tilde{P}_i^*,2} \geq \sum_{i=1}^{\mu-1}\sum_{C\in D}(||\tilde{P}_i^* \cap C|| - \Delta_C) = \sum_{C\in D} ||P^*\cap C|| - (\mu-1)\Delta_C$ which implies $\sum_{i=1}^{\mu-1} \xi_{\tilde{P}_i^*,2} \geq \sum_{C\in D} \frac{||P^*\cap C||}{2}$. Using this one can show that at the end $||P'|| \leq ||P^*|| + \epsilon\sum_{i=1}^{\mu-1} \xi_{\tilde{P}_i^*,2}$. This slightly more general version will be used later when designing our algorithm for deadline TSP.

2.2.4 QPTAS for Point to Point Orienteering on Graphs with Constant Doubling Dimension

Let $G = (V, E)$ be a graph with constant doubling dimension κ, given a budget $B > 0$ and start and end node pair $s, t \in V$ as an instance of Point to Point Orienteering on G. We show how to use the results from the previous section to get a $(1+\epsilon)$-approximation for the point to point orienteering instance in quasi polynomial time.

Lemma 7 *Let P^* be the optimal for the Point to Point Orienteering instance and $k = |P^*|$, then we can get a s-t path that visits $(1-\epsilon)k$ vertices in G with the length at most B.*

Proof. Let $\mu = \lceil \frac{1}{\epsilon} \rceil + 1$ and assume $k \geq \mu^2$, otherwise we can find P^* by exhaustive search in time $O(n^{1/\epsilon^2})$. Let $P^* = \langle v_1, \cdots, v_k \rangle$. We construct subsequence of indices of $1, \cdots, k$ to define a μ-skeleton of P^*: set $a_i = \lceil \frac{(i-1)(k-1)}{\mu-1} \rceil + 1, 1 \leq i \leq \mu$. Note that $a_1 = 1$ and $a_\mu = k$. Let $P^*_1, P^*_2, \cdots, P^*_{\mu-1}$ be the subpaths of P^* divided by $\{v_{a_1}, \cdots, v_{a_\mu}\}$, i.e. P^*_i is a subpath of P^* with the start and end node pair v_{a_i} and $v_{a_{i+1}}$. For each P^*_i we consider the 2-excess of it and let P^*_j be the subpath with maximum 2-excess among $\{P_i\}_{i=1}^{\mu-1}$, i.e. $j = \arg\max_i \xi_{P^*_i,2}$. Note $|P^*_j| = a_{j+1} - a_j + 1 = (\lceil \frac{(j)(k-1)}{\mu-1} \rceil + 1) - (\lceil \frac{(j-1)(k-1)}{\mu-1} \rceil + 1) + 1 \leq \lceil \frac{k-1}{\mu-1} \rceil + 1$

Then let P' be the path exactly the same as P^* except P' directly connecting v_{a_j} and $v_{a_{j+1}}$ in P^*_j. From the construction of P',

$$|P'| = k - |P^*_j| + 2 \geq k - \lceil \frac{k-1}{\mu-1} \rceil - 1 + 2 \geq k - \lfloor \frac{k-1}{\mu-1} \rfloor \geq (1-\epsilon)k \quad (2.1)$$

and

$$||P'|| = ||P^*|| - \xi_{P^*_j,2} = B - \xi^*_{P_j,2}. \quad (2.2)$$

We consider P' as a feasible solution for a point to point k'-TSP instance with $k' = |P'| \geq (1-\epsilon)k$ and $s, t \in V$. By Theorem 10, we can compute a (ϵ, μ)-approximation for the instance where $\mu = \lceil \frac{1}{\epsilon} \rceil + 1$, denoted as P'':

$$|P''| \geq k' \geq (1-\epsilon)k \quad (2.3)$$

and

$$||P''|| \leq ||P'|| + \epsilon\xi_{P',\mu} \leq B - \xi_{P_j^*,2} + \epsilon\xi_{P',\mu}. \tag{2.4}$$

We consider the μ-skeleton of P': $\langle v_{a_1}, \cdots, v_{a_\mu}\rangle$, by the definition of excess how we obtained P' from P^* (by short-cutting P_j^*):

$$\xi_{P',\mu} \leq ||P'|| - ||\langle v_{a_1}, \cdots, v_{a_\mu}\rangle\rangle|| \leq \sum_{i=1}^{\mu-1} \xi_{P_i^*,2} - \xi_{P_j^*,2} \tag{2.5}$$

Thus $||P''|| \leq B - \xi_{P_j^*,2} + \frac{1}{\mu-1}(\sum_{i=1}^{\mu-1} \xi_{P_i^*,2} - \xi_{P_j^*,2}) = B + \frac{1}{\mu-1}(\sum_{i=1}^{\mu} \xi_{P_i^*,2} - \mu\xi_{P_j^*,2}) \leq B$, where the last inequality comes from the construction of P_j^* : $\xi_{P_j^*,2} \geq \frac{1}{\mu}\sum_{i=1}^{\mu-1} \xi_{P_i^*,2}$. □

However, for the point to point orienteering instance, P^* and k are unknown in advance. Therefore, we will consider all possible integers $1 \leq k \leq n$ and for each k we get the approximation for point to point k-TSP on G with specified k and $s, t \in V$. We return the maximum k such that the length of path computed for point to point k-TSP is at most B. This completes the proof of Theorem 2.

Chapter 3

Deadline TSP

In this chapter we consider deadline TSP on different metrics. In Section 3.1 we consider graphs with bounded treewidth and prove Theorem 3. In Section 3.2, we consider graphs with constant doubling dimension and prove Theorem 4. In setion 3.3, we show without assuming distances are integers (rational value instead) we can get bicriteria approximations thus implies Theorem 5 and 6 .

3.1 Deadline TSP on Graphs with Bounded Treewidth

3.1.1 Overview of the Technique

In this section we prove Theorem 3. The idea is inspired by [13] for $O(1)$-approximation for deadline TSP for general metrics combined with some new ideas as well as extending our idea of point to point orienteering developed in the previous section. In [13], they present the first constant factor approximation for deadline TSP running in time $n^{O(\log n\Delta)}$ assuming that all the distances are integers.

They use the notion of regret which is the same as 2-excess. Note if u and v are vertices on a path P where u is appeared before v then shortcutting the subpath of P_{uv}, i.e. directly connecting u to v in P will save a length which is exactly $\xi_{P_{uv},2}$. They guess a set of vertices $\{v_0, v_1, \cdots, v_m\}$ of the optimal (where $m = \log \Delta_G$) based on the 2-excess of the subpaths of optimal divided

by these vertices: the 2-excess of the subpath of optimal from v_i to v_{i+1} is at least α^i, where α is some constant stratifying $1 + \alpha \geq \alpha^2$. They consider a set of point to point orienteering instances: with source v_i, sink v_{i+1} and length budget $d(v_i, v_{i+1}) + \gamma^i$. These instances are not independent however, hence this is a more general problem that we call multi-groups-legs orienteering (to be defined formally soon). They show given an β-approximation for point to point orienteering, at an $O(1)$-factor loss, one can turn it into an $O(\beta)$-approximation for multi-groups-legs orienteering: they use known reductions from the problem of maximum coverage with group budgets to classic maximum coverage and use algorithm of [9] to get a constant approximation via a reduction to classic point to point orienteering. Then they concatenate these paths; these paths are not respecting the deadlines however. In order to make them deadline respecting, from every three consecutive paths they shortcut two of them (i.e. drop those vertices), so at another $O(1)$-factor loss. The saving for the shortcuting of two paths is enough for the deadline of every vertex in the third path being satisfied. Putting everything together, to obtain an $O(1)$-approximation for deadline TSP they lose $O(1)$ factor in multiple steps. First, starting from $O(1)$-approximation for point to point orienteering, one loses another $O(1)$ factor to get an approximation for multi-groups-legs orienteering. To combine the solutions and convert it to a feasible solution of the deadline TSP instance, they lose another $O(1)$ factor. In our setting in order to get a $(1+\epsilon)$-approximation for deadline TSP on graphs with bounded treewidth, we have to change all these steps so that we don't lose more than $(1 + \epsilon)$ factor in any step. As in [13], we assume distances are integers.

3.1.2 From Deadline TSP to Multi-groups-legs Orienteering

Let $G = (V, E)$ be a graph with bounded treewidth ω, given a start node $s \in V$ and $D(v)$ for all $v \in V$ as an instance of deadline TSP on G. Let $n = |V|$ and $\log \Delta_G = \delta$. Let $\mu = \lfloor \frac{1}{\epsilon} \rfloor + 1$ and $\alpha = (1 + \epsilon)$. Let P^* be the optimal for the deadline TSP instance and $\langle v_0, v_1, \cdots, v_m \rangle$ be a sequence of vertices in P^* satisfying the following properties:

- $v_0 = s$ is start node of P^*
- v_{i+1} is the first vertex in P^* after v_i such that $\xi_{P^*_{v_i v_{i+1}},2} > \alpha^i$, except possibly for v_m is the last vertex of P^*.

We also denote the vertex on P^* just before v_{i+1} by v'_i. It follows that $\xi_{P^*_{v_i v'_i}} \leq \lceil \alpha^i \rceil - 1$ and since $||P^*|| \leq n\Delta_G$, $m \leq h\delta$ for some $h = O(\frac{1}{\epsilon})$. We can assume that $|P^*_{v_i v_{i+1}}| \geq \mu^2$, otherwise we can compute $P^*_{v_i v_{i+1}}$ exactly using exhaustive search. For each $0 \leq i < m$, we break $P^*_{v_i v_{i+1}}$ into $\mu - 1$ subpaths of (almost) equal sizes, denoted as $P^*_{i,j}$ for $1 \leq j < \mu$, by selecting a μ-skeleton $J_i : v_i = u_i^1, u_i^2, \ldots, u_i^\mu = v_{i+1}$ of $P^*_{v_i v_{i+1}}$. The μ-skeleton J_i is defined as follows. Assume $P^*_{v_i v_{i+1}}$ has size k_i and say $P^*_{v_i v_{i+1}} = \langle v_{i,1}, \cdots, v_{i,k_i} \rangle$ where $v_{i,1} = v_i$ and $v_{i,k_i} = v_{i+1}$. If we let $a_j = \lceil \frac{(j-1)(k_i-1)}{\mu-1} \rceil + 1$ then $a_1 = 1$, $a_\mu = k_i$, and if we consider $v_{i,a_1}, \cdots, v_{i,a_\mu}$ then we obtain J_i by letting $v_{i,a_j} = u_i^j$. Suppose $J^*_\mu = J^*_\mu(P^*_{v_i v_{i+1}})$ is the optimum μ-skeleton of $P^*_{v_i v_{i+1}}$, which is the μ-skeleton with the maximum length. Recall that $\xi_{P^*_{v_i v_{i+1}},\mu}$ is the μ-excess of $P^*_{v_i v_{i+1}}$ and with $B_i = ||J^*_\mu(P^*_{v_i v_{i+1}})|| + \xi_{P^*_{v_i v_{i+1}},\mu}$ we have $||P^*_{v_i v_{i+1}}|| = B_i$. To simplify the notation we denote the μ-excess of subpath $P^*_{v_i v_{i+1}}$ by $\xi^*_{i,\mu}$. We also use $\xi'_{i,\mu}$ to denote the μ-excess of path $P^*_{v_i v'_i}$. Let v be an arbitrary vertex in $P^*_{v_i v_{i+1}}$ that falls in between u_i^j and u_i^{j+1}. We use $||J_i(v_i, u_i^j)||$ to denote the length of the J_i path from v_i to u_i^j (i.e. following along J_i from the start node v_i to u_i^j). Define $L_{i,j} = \sum_{j=0}^{i-1} ||P^*_{v_j v_{j+1}}|| + ||J_i(v_i, u_i^j)||$. Note that the visiting time of v in P^* (and hence the deadline of v) is lower bounded by $L_{i,j}$.

Let $N_{i,j} = \{v : D(v) \geq L_{i,j}\}$. Note that if we consider $P^*_{v_i v_{i+1}}$ broken up into several legs $P^*_{u_i^j u_i^{j+1}}$, $1 \leq j < \mu$, then it is a point to point instance with budget $B_i = ||J^*_\mu(P^*_{v_i v_{i+1}})|| + \xi^*_{i,\mu}$, start and end node pair v_i and v_{i+1} with additional condition that given extra intermediate nodes u_i^j and subsets $N_{i,j} \subset V$ and the path is supposed to go through these intermediate nodes in this order, in other words it consists of $\mu - 1$ legs where leg j is between u_i^j, u_i^{j+1} and in each leg j it visits vertices in $N_{i,j}$. For a path Q, let $Q \cap N_{i,j}$ denote the number of vertices in $N_{i,j}$ visited by Q. We consider for all $0 \leq i \leq m$ concurrently and give a formal definition of multi-groups-legs orienteering:

Definition 8 *Let $G = (V, E)$ be a graph, given groups $0 \leq i < m$ and each groups with legs $1 \leq j < \mu$, start-end node pair $(s_{i,j}, t_{i,j})$, budget B_i and subset $N_{i,j} \subset V$ an instance of multi-groups-legs orienteering. The goal is to find a collection of paths $Q_{i,j}$, for $0 \leq i < m$, $1 \leq j < \mu$, such that $Q_{i,j}$ is a $s_{i,j}$-$t_{i,j}$ path such that $\sum_{j=1}^{\mu-1} ||Q_{i,j}|| \leq B_i$ and $|\cup_{i=0}^{m-1} \cup_{j=1}^{\mu-1} (Q_{i,j} \cap N_{i,j})|$ is maximized.*

Note $P^*_{i,j}$ $(0 \leq i < m, 1 \leq j < \mu)$ is a feasible solution of the multiple groups-legs orienteering instance with groups $0 \leq i < m$ and legs $1 \leq j < \mu$, start-end node pairs (u_i^j, u_i^{j+1}), budgets $B_i = ||J^*_\mu(P^*_{v_i v_{i+1}})|| + \xi^*_{i,\mu}$ and subset $N_{i,j} = \{v : D(v) \geq L_{i,j}\}$. We will show there is a nearly optimal solution, i.e. a set of paths $Q_{i,j}$ such that $Q_{i,j}$ is a u_i^j, u_i^{j+1}-path and if we define concatenation of different legs of group i by $Q_i = Q_{i,1} + \ldots + Q_{i,\mu-1}$ then the following conditions hold:

$$||Q_i|| = \sum_{j=1}^{\mu-1} ||Q_{i,j}|| \leq ||P^*_{v_i v_{i+1}}|| - \lceil \epsilon \xi^*_{i,\mu} \rceil. \tag{3.1}$$

$$|\cup_{i=0}^{m} Q_i| = |\cup_{i=0}^{m} \cup_{j=1}^{\mu-1} (Q_{i,j} \cap N_{i,j})| \geq (1 - 3\epsilon)|P^*|. \tag{3.2}$$

We also show that if v is visited by $Q_{i,j}$ then if $Q_{i,j}(u_i^j, v)$ denotes the segment of path $Q_{i,j}$ from u_i^j to v, then the length of the segment from v_i to v in Q_i can be upper bounded:

$$||Q_i(v_i, v)|| = \sum_{\ell=1}^{j-1} ||Q_{i,\ell}|| + ||Q_{i,j}(u_i^j, v)|| \leq ||J_i(v_i, u_i^j)|| + (1 - \epsilon)\xi'_{i,\mu}. \tag{3.3}$$

We will show the existence of such paths $Q_{i,j}$ and also how to find these using a dynamic programming. For now suppose we have found such paths Q_i as described above. We concatenate all these paths to obtain the final answer $Q = Q_0 + Q_1 + \ldots + Q_m$. We show the vertices in Q are visited before their deadlines and hence we obtain a feasible solution for the deadline TSP instance. Given the bounds for the sizes of Q_i's in (3.2), the number of vertices visited overall in Q (respecting their deadlines) is at least $(1 - 3\epsilon)|P^*|$. Replacing ϵ with $\frac{\epsilon}{3}$ we get a $(1 + \epsilon)$ approximation for the deadline TSP instance.

To see why the vertices in Q respect their deadlines consider an arbitrary vertex $v \in Q_i$. Note that each Q_i contains the vertices in J_i (as those are the vertices that define $\mu - 1$ legs of the i'th group). Suppose v is visited in $Q_{i,j}$, i.e. between u_i^j and u_i^{j+1}. Therefore, the visit time of v in Q, i.e. $||Q_{sv}||$ is bounded by:

$$\begin{aligned}
||Q_{sv}|| &= \sum_{\ell=0}^{i-1} ||Q_\ell|| + ||Q_i(v_i, v)|| \\
&\leq \sum_{\ell=0}^{i-1} (||P^*_{v_\ell v_{\ell+1}}|| - \lceil \epsilon \xi^*_{\ell,\mu} \rceil) + ||J_i(v_i, u_i^j)|| + (1-\epsilon)\xi'_{i,\mu} \text{using (3.1) and (3.3)} \\
&= L_{i,j} + (1-\epsilon)\xi'_{i,\mu} - \sum_{\ell=0}^{i-1} \lceil \epsilon \xi^*_{\ell,\mu} \rceil \\
&\leq D(v)
\end{aligned}$$

where the last inequality follows from the fact that $\xi'_{i,\mu} \leq \lceil \alpha^i \rceil - 1$ and $\xi^*_{\ell,\mu} \geq \alpha^\ell$ so $\sum_{\ell=0}^{i-1} \lceil \epsilon \xi^*_{\ell,\mu} \rceil \geq \sum_{\ell=0}^{i-1} \lceil \epsilon \alpha^\ell \rceil = \lceil \alpha^i \rceil - 1 \geq \xi'_{i,\mu}$.

3.1.3 Multi-groups-legs Orienteering on Graphs with Bounded Treewidth

We need to show the existence of $Q_i, 0 \leq i < m$ as described and show how to find them. To simplify notation, for each i, let P_i^* be the subpath $P^*_{v_i v_{i+1}}$ and $P^*_{i,j}$ be the subpath $P^*_{u_i^j u_i^{j+1}}$. We start from $P^*_{i,j}$ $(1 \leq j < \mu)$, for each $P^*_{i,j}$ we consider the 2-excess of it and let j' be the index that $P^*_{i,j'}$ has the largest 2-excess among indices $1, \ldots, \mu - 2$. We consider short-cutting the two subpaths $P^*_{i,j'}$ and $P^*_{i,\mu-1}$ and let the resulting path obtained from P_i^* by these short-cutting be Q_i. In other words, Q_i is the same as P_i^* except that each of $P^*_{i,j'}$ and $P^*_{i,\mu-1}$ are replaced with the direct edges $(u_i^{j'}, u_i^{j'+1})$ and $(u_i^{\mu-1}, v'_i)$, respectively. Let $D_i^2 = \xi_{P^*_{i,j'},2} + \xi_{P^*_{i,\mu-1},2}$. We have $D_i^2 \geq \lceil \frac{1}{\mu-1} \sum_{j=1}^{\mu-1} \xi_{P^*_{i,j},2} \rceil$. For vertices in Q_i we can bound the length of the subpath from v_i to v by:

$$||Q_i(v_i, v)|| \leq ||J_i(v_i, u_i^j)|| + \xi'_{i,\mu} - \frac{1}{\mu - 1} \sum_{j=1}^{\mu-2} \xi_{P^*_{i,j},2} \leq ||J_i(v_i, u_i^j)|| + (1-\epsilon)\xi'_{i,\mu}. \tag{3.4}$$

Also for the total length of Q_i we have:

$$||Q_i|| \leq ||P_i^*|| - D_i^2 \leq ||P_i^*|| - \lceil \frac{1}{\mu - 1} \sum_{j=1}^{\mu-1} \xi_{P_{i,j}^*,2} \rceil \leq ||P_{v_i v_{i+1}}^*|| - \lceil \epsilon \xi_{i,\mu}^* \rceil. \quad (3.5)$$

Note that $|P_{i,j'}^*| = a_{j'+1} - a_{j'} + 1 = (\lceil \frac{(j'+1)(k_i-1)}{\mu-1} \rceil + 1) - (\lceil \frac{j'(k_i-1)}{\mu-1} \rceil + 1) + 1 \leq \lceil \frac{k_i-1}{\mu-1} \rceil + 1$. Same bound holds for $|P_{i,\mu-1}^*|$. From the construction of Q_i: $|Q_i| = k_i - |P_{i,j'}^*| + 2 - |P_{i,\mu-1}^*| + 2 \geq k_i - 2\lceil \frac{k_i-1}{\mu-1} \rceil + 2 \geq k_i - 2\lfloor \frac{k_i-1}{\mu-1} \rfloor \geq (1-3\epsilon)k_i$, since we assumed $k_i \geq \mu^2$.

Now we describe our approximation algorithm for deadline TSP on the given instance on graph G of bounded treewidth ω. The algorithm has two phases. In Phase 1 we guess the vertices $v_1, \ldots, v_m$ of optimum as well as $u_i^1, \ldots, u_i^{\mu-1}$ for each $0 \leq i < m$, and also guess $||J_\mu^*(P_{v_i v_{i+1}}^*)||$ and $\xi_{i,\mu}^*$; so we get $B_i = ||J_\mu^*(P_{v_i v_{i+1}}^*)|| + \xi_{i,\mu}^*$ as well as sets $N_{i,j}$. For each i we can guess $k_i = |P_{v_i v_{i+1}}^*|$ and for those $k_i < \mu^2$ we guess $P_{v_i v_{i+1}}^*$ exactly. All the guesses in Phase 1 can be done in time $(n\Delta)^{O(\mu h \delta)}$.

In Phase 2 we find the nearly optimum solution Q_i's for those i's that $k_i \geq \mu^2$ using dynamic programming. In the remaining we assume all i's satisfy $k_i \geq \mu^2$. Similar to the case of point to point orienteering on the graph with bounded tree width, we could build a tree decomposition T of G such that T is binary, the height of T is $\rho \log n$ for some constant $\rho > 0$ and the width of T is at most $\omega' = 3\omega + 2 = O(\omega)$ (and we use ω instead of ω' to refer to its treewidth). The size of T is then at most $2^{\rho \log n} = n^\rho$. Recall for a bag $b \in T$, V_b is the set containing vertices associated with bag b, C_b denotes the union of associated vertices in bags below and including b and G_b denotes the corresponding subgraph in G over the vertices set C_b. Let b_1 and b_2 be children bags of b in T and R_b be the set of edges in G_b crossing b_1 and b_2, i.e. $R_b = \{(u,v) \in E(G_b) : u \in G_{b_1}, v \in G_{b_2} \text{ or } u \in G_{b_2}, v \in G_{b_1}\}$, thus $|R_b| \leq |G_{b_1}||G_{b_2}| \leq (\omega+1)^2$.

We present a dynamic programming based on the tree decomposition T that can find $Q_0, Q_1, \cdots, Q_m$. Recall for any vertex $v \in V$, the bags in T containing v, i.e. T_v is a connected subtree in T. In order to avoid overcounting, for every vertex $v \in T$, we consider placing a token on v at the root of T_v. We

adapt the notation and use $|Q_{i,j} \cap N_{i,j}|$ to refer to the number of tokens picked by $Q_{i,j}$ in $N_{i,j}$. Note for any bag $b \in T$, any group $0 \leq i < m$, and any leg $1 \leq j < \mu$, the restriction of $Q_{i,j}$ in the subgraph G_b may be a collection of paths where they all enter and exit G_b via V_b where the number of such paths is at most $O(\omega^2)$ because $|V_b| \leq \omega + 1$. We introduce the notion of multi-groups-legs multi-paths orienteering in order to precisely define subproblems in the dynamic programming:

Definition 9 *(multi-groups-legs multi-paths orienteering) Let $G = (V, E)$ be a graph, given groups $0 \leq i < m$ and legs $1 \leq j < \mu$, start and end node pairs $(s_{i,j,l}, t_{i,j,l}), 1 \leq l \leq \sigma_{i,j}$, budgets B_i and subset $N_{i,j} \subset V$ as an instance of multi-group-legs multi-paths orienteering. The goal is to find a collection of paths $Q_{i,j,l}$ such that $Q_{i,j,l}$ is a $s_{i,j,l}$-$t_{i,j,l}$ path, $\sum_{j=1}^{\mu-1} \sum_{l=1}^{\sigma_{i,j}} ||Q_{i,j,l}|| \leq B_i$ and $|\cup_{i=0}^{m-1} \cup_{j=1}^{\mu-1} ((\cup_{l=1}^{\sigma_{i,j}} Q_{i,j,l}) \cap N_{i,j})|$ is maximized.*

We define a subproblem in the dynamic progrmaming as an instance of multi-groups-legs multi-paths orienteering on G_b with groups $0 \leq i < m$ and legs $1 \leq j < \mu$, start-end node pairs $(s_{i,j,l}, t_{i,j,l}), 1 \leq l \leq \sigma_{b,i,j}$, budgets $B_{b,i}$ and subset $N_{i,j} \subset V$. The goal is to find a collection of path $Q_{i,j,l}$ such that $Q_{i,j,l}$ a $s_{i,j,l}$-$t_{i,j,l}$ path, $\sum_{j=1}^{\mu-1} \sum_{l=1}^{\sigma_{b,i,j}} ||Q_{i,j}||$ is at most $B_{b,i}$ and $|\cup_{i=0}^{m-1} \cup_{j=1}^{\mu-1} (\cup_{l=1}^{\sigma_{b,i,j}} Q_{i,j,l}) \cap N_{i,j})|$ is maximized. We use $A[b, \{B_{b,i}\}_{0 \leq i < m}, \{(s_{i,j,l}, t_{i,j,l})_{l=1}^{\sigma_{b,i,j}}\}_{0 \leq i < m; 1 \leq j < \mu}]$ to denote the subproblem defined above and entry of the table store the optimal value of the subproblem.

We compute the entries of this dynamic programming table from bottom to up of T. The base cases are when b is a leaf bag of T, where G_b has constant size hence each such subproblem can be solved by exhaustive search (we will explain the details soon). In the recursion, consider any entry $A[b, \{B_{b,i}\}_{0 \leq i < m}, \{(s_{i,j,l}, t_{i,j,l})_{l=1}^{\sigma_{b,i,j}}\}_{0 \leq i < m; 1 \leq j < \mu}]$, let b_1 and b_2 be the children bags of b and recall R_b is the set of edges in G_b with one endpoint in b_1 and the other in b_2. For each group i and each leg j, first we guess a subset of R_b (the set of edges $Q_{i,j,l}, 1 \leq l \leq \sigma_{b,i,j}$ crossing between b_1 and b_2) such that they are disjoint (for different i, j) and for every edge in $E_b^{i,j}$ both end points are in $N_{i,j}$, denoted as $E_b^{i,j}$. For each i we guess $B_{b_1,i}$ and $B_{b_2,i}$ such

that $B_{b_1,i} + B_{b_2,i} + \sum_{j=1}^{\mu-1} \sum_{(u,v)\in E_b^{i,j}} d(u,v) = B_{b,i}$. We show how to guess $(s_{i,j,l}, t_{i,j,l})_{l=1}^{\sigma_{b_1,i,j}}$ for b_1 and $(s_{i,j,l}, t_{i,j,l})_{l=1}^{\sigma_{b_2,i,j}}$ for b_2 and check the consistency of them: for each $E_b^{i,j}$ and for each edge in $E_b^{i,j}$ we guess it is in which one of the $\sigma_{b,i,j}$ path with the start and end node pair $(s_{i,j,l}, t_{i,j,l})_{l=1}^{\sigma_{b,i,j}}$ and for each path with start and end node pair $(s_{i,j,l}, t_{i,j,l})$ we further guess the order of the guessed edges appearing on the path. Specifically speaking, let $e_1, e_2 \cdots, e_w$ be the edges guessed in the path with start and end node pair $(s_{i,j,l}, t_{i,j,l})$ appearing in this order. Without loss of generality, say $s_{i,j,l} \in V_{b_1}$ and $t_{i,j,l} \in V_{b_2}$. Then we set $s_{i,j,l}$ and the endpoint of e_1 in V_{b_1} to be a start and end node pair in group i and leg j in b_1, the endpoint of e_1 in V_{b_2} and the endpoint of e_2 in V_{b_2} to be a start and end node pair in group i and leg j in b_2, $\cdots$, the endpoint of e_w in V_{b_2} and $t_{i,j,l}$ to be a start and end node pair in group i and leg j in b_2. By doing so we generate start and end node pairs in group i and leg j for b_1 and b_2 and we sort them based on their appearing in $s_{i,j,l}$-$t_{i,j,l}$ path. This defines $\sigma_{b_1,i,j}$ start-end node pairs for b_1 and $\sigma_{b_2,i,j}$ start-end node pairs for b_2. Formally, to compute $A[b, \{B_{b,i}\}_{0\le i<m}, \{(s_{i,j,l}, t_{i,j,l})_{l=1}^{\sigma_{b,i,j}}\}_{0\le i<m;1\le j<\mu}]$:

- let b_1 and b_2 be the children bags of b. Let R_b be the set of edges in G_b crossing b_1 and b_2.
- for each i and each j, we guess a subset of R_b, denoted as $E_b^{i,j}$ such that they are disjoint (for different i, j) and for every edge in $E_b^{i,j}$ both end points are in $N_{i,j}$.
- for each i, we guess $B_{b_1,i}$ and $B_{b_2,i}$ such that $B_{b_1,i}+B_{b_2,i}+\sum_{j=1}^{\mu-1} \sum_{(u,v)\in E_b^{i,j}} d(u,v) = B_{b,i}$.
- for each $E_b^{i,j}$ and each edge in $E_b^{i,j}$, we guess it is in which one of the $\sigma_{b,i,j}$ path with start and end node pair $(s_{i,j,l}, t_{i,j,l})_{l=1}^{\sigma_{b,i,j}}$ and for each path with start-end node pair $(s_{i,j,l}, t_{i,j,l})$ we guess the order of the edges appearing on the path as described above. We generate start-end pairs in group i and leg j for b_1 and b_2 accordingly. Then:
- $A[b, \{B_{b,i}\}_{0\le i<m}, \{(s_{i,j,l}, t_{i,j,l})_{l=1}^{\sigma_{b,i,j}}\}_{0\le i<m;1\le j<\mu}] =$
  $\max[b_1, \{B_{b_1,i}\}_{0\le i<m}, \{(s_{i,j,l}, t_{i,j,l})_{l=1}^{\sigma_{b_1,i,j}}\}_{0\le i<m;1\le j<\mu}]$

$+[b_2, \{B_{b_2,i}\}_{0\le i<m}, \{(s_{i,j,l}, t_{i,j,l})_{l=1}^{\sigma_{b_2,i,j}}\}_{0\le i<m;1\le j<\mu}]$, where the maximum is taken over all tuples
$(\{B_{b_1,i}\}_{0\le i<m}, \{B_{b_2,i}\}_{0\le i<m}, \{(s_{i,j,l}, t_{i,j,l})_{l=1}^{\sigma_{b_1,i,j}}\}_{0\le i<m;1\le j<\mu}, \{(s_{i,j,l}, t_{i,j,l})_{l=1}^{\sigma_{b_2,i,j}}\}_{0\le i<m;1\le j<\mu})$
as described above.

As said earlier, we have guessed $v_0, v_1, \cdots, v_m$, $u_i^1, \ldots, u_i^{\mu-2}$ (for $0 \le i < m$), also B_i and $\xi_{i,\mu}^*$ in Phase 1. The goal is to compute $A[r, \{B_i - \epsilon\xi_{i,\mu}^*\}_{0\le i<m}, \{(u_i^j, u_i^{j+1})\}_{0\le i<m;1\le j<\mu}]$ where r is the root bag in T.

The base case is when b is a leaf bag in Γ, i.e. C_b is exactly V_b, thus $|G_b| \le \omega + 1$. For each group i and leg j, note $\sigma_{b,i,j}$ is at most $O(\omega)^2$ in this case because there are at most $(\omega+1)^2$ pairs of vertices in G_b. We guess a subset of $V_b \cup N_{i,j}$, denoted as $U_{i,j}$ such that they are disjoint for (different i and j), which are at most $(2^{\omega+1})^{m\mu}$. We can enumerate all possible disjoint collections of $\{Q_{i,j,l}\}_{l=1}^{\sigma_{b,i,j}}$ such that $Q_{i,j,l}$ is a $s_{i,j,l}$-$t_{i,j,l}$ path. Specifically speaking, for each vertex in $U_{i,j}$ we guess it is in which one of $\sigma_{b,i,j}$ path with start-end node pair $(s_i, t_i)_{i=1}^{\sigma_{b,i,j}}$. For each path with start-node end pair $(s_{i,j,l}, t_{i,j,l})$ we guess the order of vertices appearing on the path. There are at most $|U_{i,j}|!|U_{i,j}|^{\sigma_{b,i,j}}$ guessings which are at most $[(\omega+1)!(\omega+1)^{2(\omega+1)^2}]^{m\mu}$. Among these enumeration of $\{Q_{i,j,l}\}_{l=1}^{\sigma_{b,i,j}}, 0 \le i < m, 1 \le j < \mu$ which are at most $= \omega^{O(\frac{\omega^2}{\epsilon^2}\delta)}$ many, we consider the one such that $\sum_{j=1}^{\mu-1}\sum_{l=1}^{\sigma_{b,i,j}} ||Q_{i,j,l}|| \le B_{b,i}$ for all i with maximized $|\cup_{i=0}^{m-1}\cup_{j=1}^{\mu-1}((\cup_{l=1}^{\sigma_{b,i,j}} Q_{i,j,l}) \cap N_{i,j})|$.

Now we analyze the running time of dynamic programming. First we show the running time of computing one entry of the dynamic programming table is at most $n^{O((\frac{\omega\delta}{\epsilon})^2)}$. In the recursion, for bag b and for $\{E_b^{i,j}\}_{0\le i<m;1\le j<\mu}$: for each i and leg j, because $E_b^{i,j} \subset R_b$ and $|R_b| \le (\omega+1)^2$ thus there are at most $[2^{(\omega+1)^2}]^{m\mu}$ many possible $E_b^{i,j}$ to consider for all i and all j. There are at most $(n\Delta_G)^{m\mu} = n^{O((\frac{\delta}{\epsilon})^2)}$ guessings of $B_{b_1,i}$ for b_1 and $B_{b_2,i}$ b_2 such that $B_{b_1,i} + B_{b_2,i} + \sum_{j=1}^{\mu-1}\sum_{(u,v)\in E_b^{i,j}} d(u,v) = B_{b,i}$ for all i. To generate $\{(s_{i,j,l}, t_{i,j,l})_{l=1}^{\sigma_{b_1,i,j}}\}_{0\le i<m;1\le j<\mu}$ for b_1 and $\{(s_{i,j,l}, t_{i,j,l})_{l=1}^{\sigma_{b_2,i,j}}\}_{0\le i<m;1\le j<\mu}$ for b_2: for each group i and leg j and for each edge in $E_b^{i,j}$ we guess it is in which one of $\sigma_{b,i,j}$ path with start and end node pair $(s_{i,j,l}, t_{i,j,l})$, $1 \le l \le \sigma_{b,i,j}$, and for each path with start and end node pair $(s_{i,j,l}, t_{i,j,l})$ we further guess the order of

the edges appearing, which is at most $|E_b^{i,j}|!|E_b^{i,j}|^{\sigma_{b,i,j}}$ guessings. Note a start and end node pair in $\sigma_{b,i,j}$ is a pair of vertices in $V_b \cup (\{u_i^j, u_i^{j+1}\} \cap C_b)$, thus $\sigma_{b,i,j} \leq (\omega+3)^2 = O(\omega^2)$. Therefore the total guessings for all i and all j is at most $(|E_b^{i,j}|!|E_b^{i,j}|^{\sigma_{b,i,j}})^{m\mu} \leq ((\omega+1)^2!(\omega+1)^{2(\omega+3)^2})^{(m+1)\mu} = n^{O(\omega^2\delta/\epsilon)}$.

We show the size of the dynamic programming table is at most $n^{O((\frac{\delta}{\epsilon})^2)}$: Recall an entry of the table is the form of $A[b, \{B_{b,i}\}_{0\leq i<m}, \{(s_{i,j,l}, t_{i,j,l})_{l=1}^{\sigma_{b,i,j}}\}_{0\leq i<m;1\leq j<\mu}]$. For b, there are $O(n^\rho)$ many bags in T. For $B_{b,0}, \cdots, B_{b,m}$, there are at most $(n\Delta_G)^m$ choices for and for $\{(s_{i,j,l}, t_{i,j,l})_{l=1}^{\sigma_{b,i,j}}\}_{0\leq i<m;1\leq j<\mu}$, there are at most $((\omega+1)^{2(\omega+3)^2})^{m\mu}$ possible start-end node pairs to consider.

Therefore, computing the whole dynamic programming table and finding $\{Q_i\}_{0\leq i<m}$ takes at most $n^{O((\frac{\omega\delta}{\epsilon})^2)}$ time. As mentioned, we compute $A[C_0, \{B_i - \epsilon\xi^*_{i,\mu}\}_{0\leq i<m}, \{(u_i^j, u_i^{j+1})\}_{0\leq i<m;1\leq j<\mu}]$ for all guesses of $v_0, v_1, \ldots, v_m, u_i^1, \ldots, u_i^{\mu-2}$ (for $0 \leq i < m$), and also $||J^*_\mu(P^*_{v_i v_{i+1}})||$ and $\xi^*_{i,\mu}$. For each solution we consider Q that is the path obtained by concatenating $Q_0, Q_1, \cdots, Q_m$ and check if all deadlines are respected. We return the feasible solution with maximum $|Q|$. This gives us a $(1+\epsilon)$-approximation for the deadline TSP instance and completes the proof of Theorem 3.

3.2 Deadline TSP on graph with constant doubling dimension

In this section we prove Theorem 4. The idea is the same as the case of deadline TSP on graphs with bounded treewidth except we consider the multi-groups-legs orienteering on graphs with constant doubling dimension now instead of graphs with bounded treewidth. We extend the idea of nondeterministic split tree in the case of point to point k-TSP on graph with constant doubling dimension. As in previous section we assume all distances are integers.

3.2.1 From Deadline TSP to Multi-groups-legs Orienteering

Let $G = (V, E)$ be a graph with constant doubling dimension κ, given a start node $s \in V$ and deadline $D(v)$ for all $v \in V$ as an instance of deadline TSP

on G. Let $n = |V|$ and $\delta = \log \Delta_G$. Let $\mu = \lfloor \frac{1}{\epsilon} \rfloor + 1$ and $\alpha = (1 + \epsilon)$.

Similar to the case of deadline TSP on graphs with bounded treewidth, let P^* be an optimum solution for the deadline TSP instance and $\langle v_0, v_1, \cdots, v_m \rangle$ be a sequence of vertices in P^* such that $v_0 = s$ is the start node of P^* and v_{i+1} is the first vertex in P^* after v_i such that $\xi_{P^*_{v_i v_{i+1}}, i\mu} > \alpha^i$, except possibly for v_m which is the last vertex of P^*. We can assume that $|P^*_{v_i v_{i+1}}| \geq \mu^2$, otherwise we can compute $P^*_{v_i v_{i+1}}$ exactly using exhaustive search. We also denote the vertex on P^* immediately before v_{i+1} by v'_i. It follows that $\xi_{P^*_{v_i v'_i}, \mu} < \lceil \alpha^i \rceil - 1$. Note $||P^*|| \leq n\Delta_G$, thus $m \leq h\delta$ (where $\delta = \log \Delta_G$) for some constant $h = O(\frac{1}{\epsilon})$.

For each $0 \leq i < m$, we select a μ-skeleton $J_i : v_i = u_i^1, u_i^2, u_i^3, \ldots, u_i^{\mu-1}, u_i^\mu = v_{i+1}$ as follows: assume $P^*_{v_i v_{i+1}} = \langle v_{i,1}, \ldots, v_{i,k_i} \rangle$ where $v_i = v_{i,1}$ and $v_{i,k_i} = v_{i+1}$, let $a_j = \lceil \frac{(j-1)(k_i-1)}{\mu-1} \rceil + 1$ then $a_1 = 1$, $a_\mu = k_i$. Then we let $u_i^j = v_{i,a_j}$ to get J_i. Thus we break $P^*_{v_i v_{i+1}}$ into $\mu - 1$ subpaths of (almost) equal sizes, denoted as $P^*_{i,j}, 1 \leq j < \mu$. Suppose $J^*_\mu = J^*_\mu(P^*_{v_i v_{i+1}})$ is the optimum μ-skeleton of $P^*_{v_i v_{i+1}}$ Recall that $\xi_{P^*_{v_i v_{i+1}}, \mu}$ is the μ-excess of $P^*_{v_i v_{i+1}}$. We let $B_i = ||P^*_{v_i v_{i+1}}|| = ||J^*_\mu(P^*_{v_i v_{i+1}})|| + \xi_{P^*_{v_i v_{i+1}}, \mu}$. To simplify the notation we let $\xi^*_{i,\mu}$ denote the μ-excess of subpath $P^*_{v_i v_{i+1}}$ and let $\xi'_{i,\mu}$ denote the μ-excess of path $P^*_{v_i v'_i}$. Let v be an arbitrary vertex in $P^*_{v_i v_{i+1}}$ that falls in between u_i^j and u_i^{j+1}. We use $||J_i(v_i, u_i^j)||$ to denote the length of the J_i path from v_i to u_i^j. We define $L_{i,j} = \sum_{j=0}^{i-1} ||P^*_{v_j v_{j+1}}|| + ||J_i(v_i, u_i^j)||$. Note that the visiting time of v in P^* (and hence the deadline of v) is lower bounded by $L_{i,j}$.

Let $N_{i,j} = \{v : D(v) \geq L_{i,j}\}$. Observe that if we consider $P^*_{v_i v_{i+1}}$ broken up into several legs $P^*_{u_i^j u_i^{j+1}}$, $1 \leq j < \mu$, then it is a point to point instance with start node v_i, end node v_{i+1} and given extra intermediate nodes u_i^j and the path is supposed to go through these intermediate nodes in this order; so it consists of $\mu - 1$ legs where leg j is between u_i^j, u_i^{j+1} and uses vertices in $N_{i,j} \subset V$ and total budget $B_i = ||J^*_\mu(P^*_{v_i v_{i+1}})|| + \xi^*_{i,\mu}$. We consider all i concurrently and use the same notion of multi-groups-legs orienteering problem introduced in Definition 8.

Note $P^*_{u_i^j, u_i^{j+1}}$ $(0 \leq i < m, 1 \leq j < \mu)$ is a feasible solution of the multiple groups-legs orienteering instance with groups $0 \leq i < m$ and legs $1 \leq j < \mu$, start-end node pairs (u_i^j, u_i^{j+1}), budgets $B_i = ||J^*_\mu(P^*_{v_i v_{i+1}})|| + \xi^*_{i,\mu}$ and subset

$N_{i,j} = \{v : D(v) \geq L_{i,j}\}$. We will show there is a nearly optimal solution, i.e. a set of paths $Q'_{i,j}$ such that $Q'_{i,j}$ is a u_i^j, u_i^{j+1}-path and if we define concatenation of different legs of group i by $Q'_i = Q'_{i,1} + \cdots + Q'_{i,\mu-1}$ then the following conditions hold:

$$||Q'_i|| = \sum_{j=1}^{\mu-1} ||Q'_{i,j}|| \leq ||P^*_{v_i v_{i+1}}|| - \lceil \epsilon \xi^*_{i,\mu} \rceil. \tag{3.6}$$

$$| \cup_{i=0}^{m} Q'_i| = | \cup_{i=0}^{m} \cup_{j=1}^{\mu-1} (Q'_{i,j} \cap N_{i,j})| \geq (1 - 4\epsilon)|P^*|. \tag{3.7}$$

We also show that if v is visited is visited by $Q'_{i,j}$ then if $Q'_{i,j}(u_i^j, v)$ denotes the segment of path $Q'_{i,j}$ from u_i^j to v, then the length of the segment from v_i to v in Q_i can be upper bounded:

$$||Q'_i(v_i, v)|| = \sum_{\ell=1}^{j-1} ||Q'_{i,\ell}|| + ||Q'_{i,j}(u_i^j, v)|| \leq ||J_i(v_i, u_i^j)|| + (1 - \epsilon)\xi'_{i,\mu}. \tag{3.8}$$

We will show the existence of such paths $Q'_{i,j}$ and also show how to find these using a dynamic programming. For now suppose we have found such paths Q'_i as described above. We concatenate all these paths to obtain the final answer $Q = Q'_0 + Q'_1 + \ldots + Q'_m$. We show the vertices in Q are visited before their deadlines and hence we get a feasible solution for the deadline TSP instance. Given the bounds for the sizes of Q_i's in (3.7), the number of vertices visited overall in Q is at least $(1 - 4\epsilon)|P^*|$. Replacing ϵ with $\frac{\epsilon}{4}$ we get a $(1 + \epsilon)$ approximation for the deadline TSP instance.

To see why the vertices in Q respect their deadlines consider an arbitrary vertex $v \in Q'_i$. Note that each Q'_i contains the vertices in J_i. Suppose v is visited in $Q'_{i,j}$, i.e. between u_i^j and u_i^{j+1}. Therefore, the visit time of v in Q, i.e. $||Q_{sv}||$ is bounded by:

$$
\begin{aligned}
||Q_{sv}|| &= \sum_{\ell=0}^{i-1} ||Q'_\ell|| + ||Q'_i(v_i, v)|| \\
&\leq \sum_{\ell=0}^{i-1} (||P^*_{v_\ell v_{\ell+1}}|| - \lceil \epsilon \xi^*_{\ell,\mu} \rceil) + ||J_i(v_i, u_i^j)|| + (1-\epsilon)\xi'_{i,\mu} \quad \text{using (3.6) and (3.8)} \\
&= L_{i,j} + (1-\epsilon)\xi'_{i,\mu} - \sum_{\ell=0}^{i-1} \lceil \epsilon \xi^*_{\ell,\mu} \rceil \\
&\leq D(v)
\end{aligned}
$$

where the last inequality follows from the fact that $\xi'_{i,\mu} \leq \lceil \alpha^i \rceil - 1$ and $\xi^*_{\ell,\mu} \geq \alpha^\ell$ so $\sum_{\ell=0}^{i-1} \lceil \epsilon \xi^*_{\ell,\mu} \rceil \geq \sum_{\ell=0}^{i-1} \lceil \epsilon \alpha^\ell \rceil = \lceil \alpha^i \rceil - 1 \geq \xi'_{i,\mu}$.

3.2.2 Multi-groups-legs Orienteering on Graphs with Constant Doubling Dimension

We need to show the existence of Q'_i as described and how to find them. We extend our idea in the case of point to point k-TSP on graphs with constant doubling dimension. We use the same notion of random decomposition π_G. Recall π_G is a partition $\{B_w\}_{w \in N}$ of G with the portal sets $\{S_w\}_{w \in N}$ where N is a $\frac{\Delta_G}{4}$-net for G and S_w is a $\beta \Delta_{B_w}$-net of B_w for $\beta = \frac{\epsilon}{2\kappa' \delta}$. R_{π_i} be the set of edges in $G(C)$ crossing π_i. Recall R_{π_G} be the set of edges in G crossing π_G and R'_{π_G} be the set of edges in G crossing π_G only through $\{S_w\}$. Note $|R'_{\pi_G}|$ is at most $(|N||S_w|)^2$ which is $(\frac{2\kappa' \delta}{\epsilon})^{4\kappa}$.

We use the same notion of non-deterministic split tree as in Definition 4. Similar to Lemma 5, the following lemma shows one can compute a $n^{O(h\delta)}$-nondeterministic split tree for G in quasi polynomial time in n:

Lemma 8 *We can compute a $n^{3h\delta}$-nondeterministic split tree for G.*

Similar to the proof of Lemma 5, we start by making C_0 to be the root of Γ where $G(C_0) = G$ and iteratively add levels to Γ. For a cluster node C, we compute $n^{3h\delta}$ independent random decompositions for $G(C)$. For each decomposition π_i, i.e. a partition $\{B_w\}_{w \in N_i}$ with the portal sets $\{S_w\}_{w \in N_i}$. We create a child split node for each π_i. For a split node s, let π_s be the corresponding decomposition and $\{B_w\}_{w \in N}$ be the partition, then for each

$w \in N$ we create a child cluster node of s. The number of the children cluster nodes of s is $|N|$ which is at most 4^κ. From the construction above if we define the height of Γ as the number of levels of cluster nodes then the height of Γ is at most δ. Thus the size of γ is at most $(n^{3h\delta}4^\kappa)^\delta$.

We also use the same notion of forcing in Definition 5. Recall ψ removes nondeterministic part of Γ by mapping a non-leaf cluster C to at most one of its children split node s. Let $\Gamma|_\psi$ be the induced hierarchical decomposition of G by ψ.

The following theorem is an extension of Theorems 8 and 10.

Theorem 11 *(multi-groups-legs orienteering structure theorem) Let $G = (V, E)$ be a graph with constant doubling dimension κ, given $s \in V$ and $D(v)$ for all v as an instance of deadline TSP and P^* be an optimal solution. Let v_i $(0 \le i < m)$, u_i^j $(0 \le i < m,\ 1 \le j < \mu)$, B_i, and $N_{i,j}$ as described in eariler this section, for $\mu = \lfloor \frac{1}{\epsilon} \rfloor + 1$. Consider a multi-groups-legs orienteering instance with groups $0 \le i < m$ and legs $1 \le j < \mu$, start and end node pairs (u_i^j, u_i^{j+1}), budgets B_i, and subset $N_{i,j}$. Then for the $n^{3h\delta}$-split-tree we computed in Lemma 8 with probability at least $1 - \frac{1}{n}$ there exists a forcing ψ on Γ and corresponding split-tree hierarchical decomposition $T = \Gamma|_\psi$ of G and a nearly optimal of the multi-groups-legs orienteering instance $Q'_{i,j}$, $(0 \le i < m,\ 1 \le j < \mu)$ such that if we define $Q'_i = Q'_{i,1} + \cdots + Q'_{i,\mu-1}$ for each $0 \le i < m$, then:*

- $|\cup_{i=0}^{m-1} \cup_{j=1}^{\mu-1} (Q'_{i,j} \cap N_{i,j})| \ge (1-4\epsilon)|\cup_{i=0}^{m-1} \cup_{j=1}^{\mu-1} P^*_{u_i^j, u_i^{j+1}}| = (1-4\epsilon)|P^*|$.
- $||Q'_i|| = \sum_{j=1}^{\mu-1} ||Q'_{i,j}|| \le B_i - \lceil \epsilon \xi^*_{i,\mu} \rceil$

  and for any vertex in Q'_i, say v visited in $Q'_{i,j}$ if we let $Q'_i(v_i, v)$ be the path from v_i to v in Q'_i, then: $||Q'_i(v_i, v)|| \le ||J^*_\mu(P^*_{v_i, v'_i})|| + (1-\epsilon)\xi'_{i,\mu}$.
- *For any cluster $C \in T$ and a decomposition $\pi_{\Phi(C)}$ of C, we have either:*
  - $|Q'_i \cap R_{\pi_{\Phi(C)}}| \le 2\kappa' \frac{\log n}{\epsilon}$ *or*
  - $|Q'_i \cap R'_{\pi_{\Phi(C)}}| \le (\frac{2\kappa'\delta}{\epsilon})^{4\kappa}$.

PROOF.

To simplify notation, for each i, let P_i^* be the subpath $P^*_{v_iv_{i+1}}$ and $P^*_{i,j}$ be the subpath $P^*_{u_i^j u_i^{j+1}}$. For each $P^*_{i,j}$ we consider the 2-excess of it and let j', j'' be the two indices that $P^*_{i,j'}, P^*_{i,j''}$ have the largest two 2-excess among indices $1, \ldots, \mu-2$. We consider short-cutting the three subpaths $P^*_{i,j'}, P^*_{i,j''}$, and $P^*_{i,\mu-1}$ and let the resulting path obtained from P_i^* by these short-cutting be Q_i. In other words, Q_i is the same as P_i^* except that each of $P^*_{i,j'}$, $P^*_{i,j''}$, and $P^*_{i,\mu-1}$ are replaced with the direct edges $(u_i^{j'}, u_i^{j'+1})$, $(u_i^{j''}, u_i^{j''+1})$, and $(u_i^{\mu-1}, t_i)$, respectively. Let $D_i^2 = \xi_{P^*_{i,j'},2} + \xi_{P^*_{i,j''},2}$ and $D_i^3 = D_i^2 + \xi_{P^*_{i,\mu-1},2}$. We have $D_i^2 \geq \lceil \frac{2}{\mu-1} \sum_{j=1}^{\mu-2} \xi_{P^*_{i,j},2} \rceil$. Recall that v_i' is the vertex on P^* just before v_{i+1}, so $||P_i^*(v_i, v_i')|| = ||J_\mu^*(P_i^*(v_i, v_i'))|| + \xi'_{i,\mu}$. So for vertices in Q_i we can bound the length of the subpath from v_i to v by:

$$||Q_i(v_i, v)|| \leq ||J_i(v_i, v))|| + \xi'_{i,\mu} - \frac{2}{\mu-1} \sum_{j=1}^{\mu-2} \xi_{P^*_{i,j},2}. \tag{3.9}$$

Also for the total length of Q_i we have:

$$||Q_i|| \leq ||P_i^*|| - D^3 \leq ||P_i^*|| - \lceil \frac{2}{\mu-1} \sum_{j=1}^{\mu-1} \xi_{P^*_{i,j},2} \rceil. \tag{3.10}$$

Note that $|P^*_{i,j'}| = a_{j'+1} - a_{j'} + 1 = (\lceil \frac{(j'+1)(k_i-1)}{\mu-1} \rceil + 1) - (\lceil \frac{j'(k_i-1)}{\mu-1} \rceil + 1) + 1 \leq \lceil \frac{k_i-1}{\mu-1} \rceil + 1$. Same bound holds for $|P^*_{i,j''}|$ and $|P^*_{i,\mu-1}|$. From the construction of Q_i: $|Q_i'| = k_i - |Q^*_{i,j'}| + 2 - |P^*_{i,j''}| + 2 - |P^*_{i,\mu-1}| + 2 \geq k_i - 3\lceil \frac{k_i-1}{\mu-1} \rceil + 3 \geq k_i - 3\lfloor \frac{k_i-1}{\mu-1} \rfloor \geq (1-4\epsilon)k_i$ since we assumed $k_i \geq \mu^2$.

We build $Q'_{i,j}$, $0 \leq i < m$, $1 \leq j < \mu$ iteratively based on $Q_{i,j}$ and at the same time build $\psi(\cdot)$ (hence T) from the top to bottom in Γ: initially we set $Q'_{i,j}$ to be $Q_{i,j}$ for $0 \leq i < m$,$1 \leq j < \mu$, and start from the root cluster node C_0 of T. We use the same notion of sparse and dense as in Definition 2. At any point when we are at a cluster node C we consider whether $\cup_{j=1}^{\mu-1} Q'_{i,j}$ it is sparse or η-dense with respect to C:

For any group i that $\cup_{j=1}^{\mu-1} Q'_{i,j}$ is sparse with respect to C, then we don't modify $(\cup_{j=1}^{\mu-1} Q'_{i,j}) \cap C$ when going down from C to any split node of C. Consider any child split nodes s of C and let π_s be the corresponding decomposition. For each j, we consider the edges of $Q'_{i,j}$ crossing the decomposition

π_s, i.e. $Q'_{i,j} \cap R_{\pi_s}$. According to Lemma 3: $\mathbb{E}(|(\cup_{j=1}^{\mu-1} Q'_{i,j}) \cap R_{\pi_s}|) \leq \kappa' \frac{\log n}{\epsilon}$. Let the event $F_{i,s}$ be the event that $|(\cup_{j=1}^{\mu-1} Q'_{i,j}) \cap R_{\pi_s}| \leq 2\kappa' \frac{\log n}{\epsilon}$. By Markov inequality $\Pr[F_{i,s}$ happens$] \geq \frac{1}{2}$. Recall there are $\gamma = n^{3h\delta} = 2^{3h\delta \log n}$ many children split nodes of C, i.e. γ random decompositions of $G(C)$ and there are at most $m = h\delta$ many paths Q'_i that are sparse with respect to C. Thus the probability that for at least one child split node s of C such that $F_{i,s}$ holds for all Q'_i's that are sparse with respect to C is at least $1 - [1 - (\frac{1}{2})^{h\delta}]^{\gamma} \geq 1 - [1 - \frac{1}{2^{h\delta}}]^{2^{3h\delta \log n}} \geq 1 - \frac{1}{n^3}$.

For any group i such that $\cup_{j=1}^{\mu-1} Q'_{i,j}$ is η-dense with respect to C, consider any child split nodes s of C and let π_s be the corresponding decomposition. We modify $\cup_{j=1}^{\mu-1} Q'_{i,j}$ to be portal respecting with respect to π_s as described in Lemma 4. Note the set of edges of $\cup_{j=1}^{\mu-1} Q'_{i,j}$ crossing π_s after making it portal respecting with respect to π_s is a subset of R'_{π_s}. Thus $|(\cup_{j=1}^{\mu-1} Q'_{i,j}) \cap R'_{\pi_s}| \leq R'_{\pi_s} \leq (\frac{2\kappa'\delta}{\epsilon})^{4\kappa}$ in this case. According to Lemma 4, the increase of length of $\cup_{j=1}^{\mu-1} Q'_{i,j}$ after making it portal respecting with respect to π_s is at most $\frac{\epsilon}{2\delta}||(\cup_{j=1}^{\mu-1} Q'_{i,j}) \cap C||$ in expectation. Let $F'_{i,s}$ be the event that the increase of length of $\cup_{j=1}^{\mu-1} Q'_{i,j}$ after making it portal respecting with respect to π_s is at most $\frac{\epsilon}{\delta}||(\cup_{j=1}^{\mu-1} Q'_{i,j}) \cap C||$. By Markov inequality, $\Pr[F'_{i,s}$ happens$] \geq \frac{1}{2}$. Recall there are γ many children split nodes of C and there are at most $m = h\delta$ many paths Q'_i that are η-dense with respect to C. Thus the probability that for at least one child split node s of C, events $F'_{i,s}$ happens for all Q'_i's that are η-dense with respect to C is at least $1 - [1 - (\frac{1}{2})^{h\delta}]^{\gamma} \geq 1 - [1 - \frac{1}{2^{h\delta}}]^{2^{3h\delta \log n}} \geq 1 - \frac{1}{n^3}$.

Therefore, such s exists for cluster C with the probability at least $1 - \frac{2}{n^3}$ for all i and we can define $\psi(C)$. Once we have $\psi(\cdot)$ defined for all clusters at a level of Γ, we have determined the clusters at the same level of $T = \Gamma|_{\Phi}$. Note there are at most n cluster nodes in one level of T. Thus with probability at least $1 - \frac{2}{n^2}$ such split nodes exist for all cluster nodes in one level. Since the height of Γ is at most δ, thus with probability at least $(1 - \frac{2}{n^2})^{\delta} \geq 1 - \frac{1}{n}$ (assuming that δ is polylogarithmic in n) such $\psi(\cdot)$ is well defined over all levels.

Note $Q'_{i,j}$ visits all the of vertices from $N_{i,j}$ that $Q_{i,j}$ was visiting, so

$$|\cup_{i=0}^{m-1}\cup_{j=1}^{\mu-1}(Q'_{i,j}\cap N_{i,j})| \geq (1-4\epsilon)|\cup_{i=0}^{m-1}\cup_{j=1}^{\mu-1}P^*_{u_i^j,u_i^{j+1}}| = (1-4\epsilon)|P^*|.$$

For any cluster $C \in T$, the increase in length of $Q'_i = \cup_{j=1}^{\mu-1}Q'_{i,j}$ only stems from when it is η-dense with respect to C. Using Theorem 10, $Q'_i = \cup_{j=1}^{\mu-1}Q'_{i,j}$ is a (ϵ,μ)-approximation of Q_i. Thus, combined with (3.10):

$$||Q'_i|| \leq ||Q_i|| + \epsilon\xi_{Q_i,\mu} = ||P^*_i|| - \lceil\frac{2}{\mu-1}\sum_{j=1}^{\mu-1}\xi_{P^*_{i,j},2}\rceil + \epsilon\xi_{Q_i,\mu} \tag{3.11}$$

Now we do similar to what we did in Lemma 7 to bound $\xi_{Q_i,\mu}$. We consider the μ-skeleton of Q_i based on $\langle u_i^1,\cdots,u_i^\mu\rangle$, which are the same nodes in the μ-skeleton of P_i^* (J_i) from which we obtained Q_i. By the definition of excess and how we obtained Q_i from P_i^*:

$$\xi_{Q_i,\mu} \leq ||Q_i|| - ||\langle u_i^1,\cdots,u_i^\mu\rangle|| = \sum_{j=1}^{\mu-1}\xi_{P^*_{i,j},2} - \xi_{P^*_{i,j'},2} - \xi_{P^*_{i,j''},2} - \xi_{P^*_{i,\mu-1},2} \leq \sum_{j=1}^{\mu-1}\xi_{P^*_{i,j},2} \tag{3.12}$$

Thus, using (3.11) and (3.12) and noting that $\mu = \lfloor\frac{1}{\epsilon}\rfloor + 1$:

$$||Q'_i|| \leq ||P^*_i|| - \sum_{j=1}^{\mu-1}\lceil 2\epsilon\xi_{P^*_{i,j},2}\rceil + \epsilon\xi_{P^*_{i,j},\mu} \leq ||P^*_i|| - \lceil\epsilon\xi^*_{i,\mu}\rceil$$

To prove the upper bound required on $||Q'_i(v_i,v)||$ when considering any vertex v in $Q'_{i,j}$ (that was visited in Q'_i between u_i^j, u_i^{j+1}) we use the slightly more general version mentioned right after Theorem 10. Using that argument and assuming that we take the μ-skeleton of P_i^* from which we derived Q_i instead, the extra length of Q'_i compared to Q_i at any dense cluster $C \in D$ is bounded by $\epsilon\sum_{j=1}^{\mu-1}\xi_{Q_{i,j},2}$.

$$
\begin{aligned}
||Q'_i(v_i,v)|| &\leq ||Q_i(v_i,v)|| + \epsilon \sum_{j=1}^{\mu-2} \xi_{Q_{i,j},2} \\
&\leq ||Q_i(v_i,v)|| + \epsilon \sum_{j=1}^{\mu-2} \xi_{P^*_{i,j},2} \\
&\leq ||J_i(v_i,v)|| + \xi'_{i,\mu} - \frac{2}{\mu-1} \sum_{j=1}^{\mu-2} \xi_{P^*_{i,j},2} + \epsilon \sum_{j=1}^{\mu-2} \xi_{P^*_{i,j},2} \qquad \text{using (3.9)} \\
&\leq ||J_i(v_i,v)|| + (1-\epsilon)\xi'_{i,\mu}.
\end{aligned}
$$

□

Now we describe how we can find such nearly optimum solution as guaranteed by Theorem 11 using dynamic programming. Recall we use P^* denote an optimum for the deadline TSP instance. Suppose we have guessed the vertices $\langle v_0, \ldots, v_m \rangle$ for P^*, for each $0 \leq i < m$ we have guessed the vertices of the μ-skeleton J_i for $P^*_{v_i v_{i+1}}$ where $J_i : v_i = u_i^1, u_i^2, u_i^3, \ldots, u_i^{\mu-1}, u_i^\mu = v_{i+1}$. We also guess $||J^*_\mu(P^*_{v_i v_{i+1}})||$ and also $\xi^*_{i,\mu}$, thus we get $B_i = ||J^*_\mu(P^*_{v_i v_{i+1}})| + \xi^*_{i,\mu}$ as well as sets $N_{i,j}$. For each i, we can guess $k_i = |P^*_{v_i v_{i+1}}|$ and for those that $k_i < \mu^2$ we can guess P^*_i exactly. Note that all these guesses can be done in time $(n\Delta)^{O(\mu h \delta)}$. We call these guessings Phase 1. In the remaining (which we call Phase 2) we find Q'_i's for those i for which $k_i \geq \mu^2$ for those i's that $k_i \geq \mu^2$ using dynamic programming. To simplify the notation we assume all i's satisfy $k_i \geq \mu^2$.

The dynamic programming is built on the γ-nondeterministic split tree Γ we computed. Consider for any cluster node $C \in \Gamma$, any group $0 \leq i < m$ and any leg $1 \leq j < \mu$, the restriction of $Q'_{i,j}$ in the subgraph $G(C)$, which might be a collection of paths because $Q'_{i,j}$ can enter and exit C multiple times. This causes $Q'_{i,j}$ be chopped up into multiple segments. For C, group i and leg j suppose that it is broken into $\sigma_{C,i,j}$ many pieces. We use the same notion of multi-groups-legs multi-paths orienteering as in Definition 9 to preciously define the subproblem in the dynamic programming. We define a subproblem in the dynamic programming as an instance of multi-groups-legs multi-paths orienteering on C with groups $0 \leq i < m$ and legs $1 \leq j < \mu$, start and

end node pairs $(s_{i,j,l}, t_{i,j,l}), 1 \leq l \leq \sigma_{C,i,j}$, budgets $B_{C,i}$ and subset $N_{i,j}$. The goal is to find a collection of paths $Q_{i,j,l}$ such that $Q_{i,j,l}$ is a $s_{i,j,l}$-$t_{i,j,l}$ path, $\sum_{j=1}^{\mu-1}\sum_{l=1}^{\sigma_{C,i,j}} ||Q_{i,j,l}|| \leq B_{C,i}$ and $|\cup_{i=0}^{m-1}\cup_{j=1}^{\mu-1}((\cup_{l=1}^{\sigma_{i,j}} Q_{i,j,l}) \cap N_{i,j})|$ is maximized. We use $A[C, \{B_{C,i}\}_{0\leq i<m}, \{(s_{i,j,l}, t_{i,j,l})_{l=1}^{\sigma_{C,i,j}}\}_{0\leq i<m;1\leq j<\mu}]$ to denote the subproblem described above and the entry of the table to store the optimal value of the subproblem. We will show $\sigma_{C,i,j}$ is poly-logarithmic.

We compute the entries of this dynamic programming table from bottom to up of Γ. The base cases are when C has constant size $|C| = a$ for some $a > 0$ thus such subproblem can be solved by exhaustive search (we will explain the details soon). In the recursion, consider any entry $A[C, \{B_{C,i}\}_{0\leq i<m}, \{(s_{i,j,l}, t_{i,j,l})_{l=1}^{\sigma_{C,i,j}}\}_{0\leq i<m;1\leq j<\mu}]$ and any child split node s of C in Γ and let $C_1, C_2, \cdots, C_g$ be the children cluster nodes of s in Γ. Recall π_s is the corresponding partition of C and R_{π_s} is the set of edges in $G(C)$ crossing π_s. For each C_b let S_b be its portal set and recall R'_{π_s} is the set of edges in $G(C)$ crossing π_s only through $\{S_b\}$.

Note for each i we consider two cases (the sparse case or the dense case). We will first guess two sets I_1^C and I_2^C that are disjoint and the union of them is $\{0, \cdots, m-1\}$. The intention is that I_1^C is the set containing those i's that are (guessed to be) sparse with respect to C and I_2^C is the set of those i's that are (guessed to be) dense with respect to C. For each $i \in I_1^C$ and for each j, we guess a subset of R_{π_s}, denoted as $E_{\pi_s}^{i,j}$ such that for each edge in $E_{\pi_s}^{i,j}$ both endpoints are in $N_{i,j}$, they are disjoint (for different i, j) and the size of $|\cup_{j=1}^{\mu-1} E_{\pi_s}^{i,j}|$ is at most $2\kappa'\frac{\log n}{\epsilon}$. For $i \in I_2^C$ and for each j, we guess a subset of R'_{π_s}, also denoted as $E_{\pi_s}^{i,j}$ that size of $|\cup_{j=1}^{\mu-1} E_{\pi_s}^{i,j}|$ is at most $(\frac{2\kappa'\delta}{\epsilon})^{4\kappa}$. For each i we guess $B_{C_1,i}, \cdots, B_{C_g,i}$, such that $\sum_{j=1}^{g} B_{C_j,i} + \sum_{j=1}^{\mu-1}\sum_{(u,v)\in E_{\pi_s}^{i,j}} d(u,v) = B_{C,i}$.

We show how to guess start-end pairs $\{(s_{i,j,l}, t_{i,j,l})_{l=1}^{\sigma_{C,i,j}}\}_{0\leq i<m;1\leq j<\mu}$ for each C_b and check the consistency of them: for each $E_{\pi_s}^{i,j}$ and for each edge in $E_{\pi_s}^{i,j}$, we guess it is in which one of the $\sigma_{C,i,j}$ paths with start-end pair $(s_{i,j,l}, t_{i,j,l})_{l=1}^{\sigma_{C,i,j}}$ and for each path with start-end pair $(s_{i,j,l}, t_{i,j,l})$ we guess the order of the guessed edges appearing on the path. Specifically speaking, let $e_1, e_2 \cdots, e_w$ be the edges guessed in the path with start-end pair $(s_{i,j,l}, t_{i,j,l})$ appearing in this order. Let $C_{a_1}, C_{a_2}, \cdots, C_{a_{w+1}}$ be the children cluster nodes of s that the

path encounters following $e_1, e_2 \cdots, e_w$, i.e. e_1 crosses C_{a_1} and C_{a_2}, e_2 crosses C_{a_2} and C_{a_3}, $\cdots$, e_w crosses C_{a_w} and $C_{a_{w+1}}$. Then we set $s_{i,j,l}$ and the endpoint of e_1 in C_{a_1} to be a start-end pair in group i and leg j in C_{a_1}, the endpoint of e_1 in C_{a_2} and the endpoint of e_2 in C_{a_2} to be a start-end pair in group i and leg j in C_{a_2}, $\cdots$, the endpoint of e_w in $C_{a_{w+1}}$ and $t_{i,j,l}$ to be a start-end pair in group i and leg j in $C_{a_{w+1}}$. By doing so we generate start-end pairs in group i and leg j for each C_b and we sort them based on their ordering in $s_{i,j,l}$-$t_{i,j,l}$ path. This defines $\sigma_{C_b,i,j}$ start-end pairs for each C_b. Formally, to compute $A[C, \{B_{C,i}\}_{0\leq i<m}, \{(s_{i,j,l}, t_{i,j,l})_{l=1}^{\sigma_{C,i,j}}\}_{0\leq i<m;1\leq j<\mu}]$:

- Consider any child split node s of C and let $C_1, C_2, \cdots, C_g$ be the children cluster nodes of s.
- Let π_s be the corresponding decomposition of C and R_{π_s} be the set of edges in $G(C)$ crossing π_s. For each C_b let S_b be its portal set and let R'_{π_s} be the set of edges in $G(C)$ crossing π_s only through $\{S_b\}$.
- Guess sets I_1^C and I_2^C such that they are disjoint and the union of them is $\{0, \cdots, m-1\}$.
- For $i \in I_1^C$, we guess a subset of R_{π_s}, denoted as $E_{\pi_s}^{i,j}$ such that for each edge in $E_{\pi_s}^{i,j}$ both endpoints are in $N_{i,j}$, they are disjoint (for different i, j) and the size of $|\cup_{j=1}^{\mu-1} E_{\pi_s}^{i,j}|$ is at most $2\kappa' \frac{\log n}{\epsilon}$. For $i \in I_2^C$ and for each j, we guess a subset of R'_{π_s}, also denoted as $E_{\pi_s}^{i,j}$ that size of $|\cup_{j=1}^{\mu-1} E_{\pi_s}^{i,j}|$ is at most $(\frac{2\kappa'\delta}{\epsilon})^{4\kappa}$
- For each i we guess $B_{C_1,i}, \cdots, B_{C_g,i}$, such that $\sum_{j=1}^{g} B_{C_j,i} + \sum_{j=1}^{\mu-1} \sum_{(u,v)\in E_{\pi_s}^{i,j}} d(u,v) = B_{C,i,\ell}$.
- For each $E_{\pi_s}^{i,j}$ and for each edge in $E_{\pi_s}^{i,j}$, we guess it is in which one of the $\sigma_{C,i,j}$ paths with start-end pair $(s_{i,j,l}, t_{i,j,l})_{l=1}^{\sigma_{C,i,j}}$ and for each path with start-end pair $(s_{i,j,l}, t_{i,j,l})$ we guess the order of the guessed edges appearing on the path. as described above. We generate start-end pairs in group i and leg j for each C_b accordingly. Then:

- We set $A[C, \{B_{C,i}\}_{0\le i<m}, \{(s_{i,j,l}, t_{i,j,l})_{l=1}^{\sigma_{C,i,j}}\}_{0\le i<m;1\le j<\mu}] =$ $\max \sum_{b=1}^{g} A[C_b, \{B_{C_b,i}\}_{0\le i<m}, \{(s_{i,j,l}, t_{i,j,l})_{l=1}^{\sigma_{C_b,i,j}}\}_{0\le i<m;1\le j<\mu}]$, where the maximum is taken over all tuples $(s, \{B_{C_b,i}\}_{1\le b\le g;0\le i<m}, \{(s_{i,j,l}, t_{i,j,l})_{l=1}^{\sigma_{C_b,i,j}}\}_{1\le b\le g;0\le i<m;1\le j<\mu})$ as described above.

As said earlier, we are going to guess all $v_0, v_1, \ldots, v_m,\ u_i^1, \ldots, u_i^{\mu-2}$ (for $0 \le i < m$), also B_i and $\xi_{i,\mu}^*$ in Phase 1. The goal is to compute $A[C_0, \{B_i - \epsilon\xi_{i,\mu}^*\}_{0\le i<m}, \{(u_i^j, u_i^{j+1})\}_{0\le i<m;1\le j<\mu}]$.

The base case is when C is a leaf cluster node in Γ, i.e. $V(c) = a$ for some constant $a > 0$ where $V(C)$ is the vertices set of $G(C)$. For each group i and leg j, note $\sigma_{C,i,j}$ is at most a^2 in this case because $s_{i,j,l}, t_{i,j,l} \in V(C)$. We guess a subset of $V_C \cup N_{i,j}$, denoted as $U_{i,j}$ such that they are disjoint for (different i and j), which are at most $(2^a)^{m\mu}$ many. We can enumerate all possible disjoint collections of $\{Q_{i,j,l}\}_{l=1}^{\sigma_{b,i,j}}$ such that $Q_{i,j,l}$ is a $s_{i,j,l}$-$t_{i,j,l}$ path. Specifically speaking, for each vertex in $U_{i,j}$ we guess it is in which one of $\sigma_{C,i,j}$ path with start-end node pair $(s_i, t_i)_{i=1}^{\sigma_{C,i,j}}$. For each path with start-end node end pair $(s_{i,j,l}, t_{i,j,l})$ we guess the order of vertices appearing on the path. There are at most $|U_{i,j}|!|U_{i,j}|^{\sigma_{C,i,j}}$ guessings which are at most $[a!a^{2a^2}]^{m\mu}$. Among these enumeration of $\{Q_{i,j,l}\}_{l=1}^{\sigma_{b,i,j}}, 0 \le i < m, 1 \le j < \mu$ which are at most $a^{O(\frac{a^2}{\epsilon^2}\delta)}$ many, we consider the one such that $\sum_{j=1}^{\mu-1}\sum_{l=1}^{\sigma_{C,i,j}} ||Q_{i,j,l}|| \le B_{C,i}$ for all i with maximized $|\cup_{i=0}^{m-1}\cup_{j=1}^{\mu-1}((\cup_{l=1}^{\sigma_{C,i,j}} Q_{i,j,l}) \cap N_{i,j})|$.

Now we analyze the running time of dynamic programming. First we show the running time of computing one entry of the dynamic programming table is at most $n^{O((\frac{\delta}{\epsilon})^{2\kappa+2})}$. In the recursion, for a cluster node C, there are $\gamma = n^{3h\delta}$ children split nodes of C to consider. For a certain split node s, let $C_1, \cdots, C_g$ be the children cluster nodes of s. There are 2^m guessing for I_1^C, I_2^C because they are disjoint and the union of them is $\{0, \cdots, m-1\}$. For $\{E_{\pi_s}^{i,j}\}_{0\le i<m;1\le j\le\mu=1}$: for each group i and leg j, if $i \in I_1^C$, then because $|\cup_{j=1}^{\mu-1} E_{\pi_s}^{i,j}| \le 2\kappa'\frac{\log n}{\epsilon}$, there are at most $n^{4\kappa'\frac{\log n}{\epsilon}}$ many possible $E_{\pi_s}^{i,j}$ to consider; if $i \in I_2^C$ then because $E_{\pi_s}^{i,j} \subset R'_{\pi_s}$ and $|R'_{\pi_s}| \le (\frac{2\kappa'\delta}{\epsilon})^{4\kappa}$ in this case, there are at most $2^{(\frac{2\kappa'\delta}{\epsilon})^{4\kappa}} \le n^{(\frac{2\kappa'\delta}{\epsilon})^{4\kappa}}$ many possible $E_{\pi_s}^{i,j}$ to consider. Hence there are at most $[n^{(\frac{2\kappa'\delta}{\epsilon})^{4\kappa}}]^{m\mu}$ many possible $E_{\pi_s}^{i,j}$ to consider for all

i and all j. There are at most $[(n\Delta_G)^g]^m$ guessings for $\{B_{C_1,i}, \cdots, B_{C_g,i}\}$ such that $\sum_{b=1}^{g} B_{C_b,i} + \sum_{j=1}^{\mu-1} \sum_{(u,v)\in E_s^{i,j}} d(u,v) = B_{C,i}$ for all i. To generate $\{(s_{i,j,l}, t_{i,j,l})_{l=1}^{\sigma_{C,i,j}}\}_{0\le i<m;1\le j<\mu}$ for each C_b: for each group i and leg j and for each edge in $E_s^{i,j}$, we guess it is in which one of $\sigma_{C,i,j}$ path with start-end pair $(s_{i,j,l}, t_{i,j,l})_{l=1}^{\sigma_{c,i,j}}$ and for each path with start-end pair $(s_{i,j,l}, t_{i,j,l})$ we guess the order of the edges appearing, which is at most $(|E_{\pi_s}^{i,j}|!|E_{\pi_s}^{i,j}|^{\sigma_{C,i,j}})$ guessings. Note at each recursion for group i and leg j it may increase at most $|E_{\pi_s}^{i,j}|$ the number of start-end pairs and the depth of the recursion is δ. Thus $\sigma_{C,i,j} \le \delta|E_{\pi_s}^{i,j}|$ and the total guessings for all i and all j is at most $[|E_{\pi_s}^{i}|!|E_{\pi_s}^{i}|^{\sigma_{c,i}}]^{(m+1)\mu} \le ((\frac{2\kappa'\delta}{\epsilon})^{4\kappa}!(\frac{2\kappa'\delta}{\epsilon})^{4\kappa\delta(\frac{2\kappa'\delta}{\epsilon})^{4\kappa}})^{(m+1)\mu} \le n^{O((\frac{\delta}{\epsilon})^{4\kappa+2})}$.

We show the size of the dynamic programming table is at most $n^{O((\frac{\delta}{\epsilon})^{4\kappa+2})}$. Recall the entry of the table is of form $A[C, \{B_{C,i}\}_{0\le i<m}, \{(s_{i,j,l}, t_{i,j,l})_{l=1}^{\sigma_{C,i,j}}\}_{0\le i<m;1\le j<\mu}]$. For C, there are at most $(n^{3h\delta}4^{\kappa})^{\delta}$ cluster nodes in Γ . For $B_{C,0}, \cdots, B_{C,m}$, there are at most $(n\Delta_G)^m$ many possible value to consider. For $\{(s_{i,j,l}, t_{i,j,l})_{l=1}^{\sigma_{C,i,j}}\}_{0\le i<m;1\le j<\mu}$, there are at most $[n^{2\sigma_{C,i,j}}]^{m\mu} = n^{O((\frac{\delta}{\epsilon})^{4\kappa+2})}$ possible start-end pairs to consider.

Therefore, computing the whole dynamic programming table and finding $\{Q_i^{'}\}_{0\le i<m}$ as in Theorem 11 takes at most $n^{O((\frac{\delta}{\epsilon})^{4\kappa+2})}$ time. As mentioned, we compute $A[C_0, \{B_i - \epsilon\xi_{i,\mu}^*\}_{0\le i<m}, \{(u_i^j, u_i^{j+1})\}_{0\le i<m;1\le j<\mu}]$ for all guesses of $v_0, v_1, \ldots, v_m,\ u_i^1, \ldots, u_i^{\mu-2}$ (for $0 \le i < m$), and also $||J_\mu^*(P_{v_iv_{i+1}}^*)||$ and $\xi_{i,\mu}^*$. For each solution we consider Q that is the path obtained by concatenating $Q_0', Q_1', \cdots, Q_m'$ and check if all deadlines are respected. We return the feasible solution with maximum $|Q|$. This gives us a $(1+\epsilon)$-approximation for the deadline TSP instance and completes the proof of Theorem 4.

3.3 Bicriteria Approximation

In this section we show removing the assumption that all distances are integers, i.e. all distances have fractional values ($\mathbb{Q}^+$) instead, then we adapt the analysis in Section 3.1 and 3.2 to get Theorem 5 and 6. At high level we bucket the distances and budgets based on powers of $\lambda = (1 + \frac{\epsilon^2}{\delta})$. For any value x let $L(x)$ be the nearest power of λ that is at most x, which we set to be the rounded down value of x. Similarly we let $R(x)$ be the nearest power of λ that

is at least x, which we set to be the rounded up value of x. Whenever we consider budget value we consider taking $R(\cdot)$ and whenever we consider distance values in our calculations we consider taking $L(\cdot)$. Note a feasible solution after rounding up budget and rounding down distance remains feasibility.

Precisely speaking, when we guess the length of the optimal μ-skeleton as well as μ-excess for group i in optimal solution, i.e. $||J^*_\mu(P^*_{v_iv_{i+1}})||$ and $\xi_{P^*_{v_iv_{i+1}},\mu}$, we actually conside rrounded down values of power of λ: $L(||J^*_\mu(P^*_{v_iv_{i+1}})||)$ and $L(\xi_{P^*_{v_iv_{i+1}},\mu})$. Then we set the budget for group i, i.e. B_i to be the sum of these two then round up the nearest power of λ: $B_i = R(||J^*_\mu(P^*_{v_iv_{i+1}})|| + \xi_{P^*_{v_iv_{i+1}},\mu})$. In our dynamic programming table, when we consider the budgets $B_{C,i}$, they are based on rounded up values of power of λ (if in the case of graphs with bounded treewidth, we consider $B_{b,i}$ based on rounded up values of power of λ). In the recursion, we will guess the budgets to be rounded up values as well, i.e. for cluster C and group i when we guess $B_{C_1,i}, \cdots, B_{C_g,i}$, we consider $R(\sum_{j=1}^{g} B_{C_j,i} + \sum_{j=1}^{\mu-1}\sum_{(u,v)\in E^{i,j}_{\pi_s}} d(u,v)) = B_{C,i}$ where $B_{C,i}, B_{C_1,i}, \cdots, B_{C_g,i}$ are all rounded up values of power of λ and $\{d(u,v)\}_{(u,v)\in E^{i,j}_{\pi_s}}$ are all rounded down values of power of λ (in the case of graphs with bounded treewidth, for bag b and group i when we guess $B_{b_1,i}$ and $B_{b_2,i}$, we consider $R(B_{b_1,i} + B_{b_2,i} + \sum_{j=1}^{\mu-1}\sum_{(u,v)\in E^{i,j}_b} d(u,v)) = B_{b,i}$ where $B_{b,i}, B_{b_1,i}, B_{b_2,i}$ are all rounded up values of power of λ and $\{d(u,v)\}_{(u,v)\in E^{i,j}_b}$ are all rounded down values of power of λ). Recall the height of Γ as well as the depth of our dynamic programming recursion is $O(\delta)$ (in the case of graphs with bounded treewidth, the height of T as well as the depth of our dynamic programming recursion is $\rho \log n$ for some constant $\rho > 0$). Thus the number of operations on distances as well as budgets are at most $O(\delta)$. Note we always underestimate the distances and overestimate the budgets. The rounding error is accumulative. For a solution we find by the dynamic programming, the actual value of distance might be at most $\lambda^{O(\delta)} \leq e^{O(\epsilon^2)} = 1 + \epsilon'$ factor of the rounded down value of distance for some $\epsilon' = O(\epsilon)$, similarly the actual value of budget might be at least $\frac{1}{1+\epsilon'}$ factor of the rounded up value of budget. In other words our solution might be violating the deadlines by at most $(1+\epsilon')^2 \leq 1 + 3\epsilon'$ factor. By replacing $3\epsilon'$ with ϵ it implies Theorem 5 and 6.

Chapter 4

Conclusion and Discussion for future work

In this thesis, we consider both point to point orienteering and deadline TSP on different metrics. For point to point orienteering, we solve it exactly in polynomial time on graphs with bounded treewidth and we show a QPTAS on graphs with constant doubling dimension when distances are quasi-poly bounded. For deadline TSP, we present a QPTAS for both graphs with bounded treewidth and graphs with constant doubling dimension when distances are quasi-poly bounded and integers.

We show some potential directions for future work and the corresponding difficulties. An immediate question one may wonder is that whether it is possible to turn the above quasi polynomial time approximation schemes into polynomial time schemes. We point out the running time in Theorems 2 includes $\log \Delta_G$ in the exponent because our algorithm is based on the hierarchical decomposition of doubling metrics (with some adaptation) in [17], which produces a decomposition of height $O(\log \Delta_G)$. Note for some vehicle routing problems like TSP on doubling metrics, we can assume distances to be polynomial in n by standard scaling with a loss of $(1+\epsilon)$ factor. This however does not work for orienteering because distance is hard constraint and scaling may violate the budget unless we are shooting for a bicriteria approximation. Also the nondeterministic split tree Γ we construct has the same height which leads to the size of the dynamic programming includes $\log \Delta_G$ in the exponent as well. Our running time in Theorems 3 and 4 includes $\log \Delta_G$ in the expo-

nent because our algorithm guess m vertices $(v_i, 0 \leq i \leq m)$ of the optimal solution where we showed $m = h \log \Delta_G$ for some $h = O(\frac{1}{\epsilon})$. For each group i we further guess its optimal μ-skeleton and μ-excess where $\mu = O(\frac{1}{\epsilon})$. The time of guessing all includes $\log \Delta_G$ in the exponent. It's hard to improve what mentioned above unless using a totally different framework.

Another question is whether the results can be extended to other metrics, for example planar graphs. In [11] the author use similar notion of sparse and dense and similar idea of non-deterministic split tree to get a QPTAS of k-MST on planar graphs (also extend to minor-free graphs). In sparse case he uses the idea of padded decomposition in [15], in this scenario the analysis is very similar to our sparse case using the hierarchical decomposition. The difference is in dense case, instead of us sticking to the hierarchical decomposition, he uses the idea of balanced separator in [18] to separate the graph by computing three shortest paths and removing them. We refer to [11] for more details of constricting non-deterministic split tree and recursion of the dynamic programming based on the split tree. It remains unclear that if our results can be adapted to planar graphs. The notion of non-deterministic split tree might be adapted considering parallel split nodes mixed with hierarchical decomposition to take care of sparse case and shortest paths separator for the dense case. The difficulty is that even for point to point k-TSP on planar graphs it's not straightforward to properly define the subproblem of the dynamic programming unlike the case of bounded treewidth or doubling metrics because in dense case there are too many interactions between different subproblems via the shortest paths separator and it's hard to decompose them independently and combine them with consistency in the recursion.

References

[1] E. M. Arkin, J. S. Mitchell, and G. Narasimhan, "Resource-constrained geometric network optimization," in *Proceedings of the fourteenth annual symposium on Computational geometry*, 1998, pp. 307–316.

[2] S. Arora, "Nearly linear time approximation schemes for euclidean tsp and other geometric problems," in *Proceedings 38th Annual Symposium on Foundations of Computer Science*, IEEE, 1997, pp. 554–563.

[3] S. Arora, M. Grigni, D. R. Karger, P. N. Klein, and A. Woloszyn, "A polynomial-time approximation scheme for weighted planar graph tsp.," in *SODA*, vol. 98, 1998, pp. 33–41.

[4] N. Bansal, A. Blum, S. Chawla, and A. Meyerson, "Approximation algorithms for deadline-tsp and vehicle routing with time-windows," in *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, 2004, pp. 166–174.

[5] Y. Bartal, L.-A. Gottlieb, and R. Krauthgamer, "The traveling salesman problem: Low-dimensionality implies a polynomial time approximation scheme," in *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, 2012, pp. 663–672.

[6] A. Blum, S. Chawla, D. R. Karger, T. Lane, A. Meyerson, and M. Minkoff, "Approximation algorithms for orienteering and discounted-reward tsp," *SIAM Journal on Computing*, vol. 37, no. 2, pp. 653–670, 2007.

[7] H. L. Bodlaender and T. Hagerup, "Parallel algorithms with optimal speedup for bounded treewidth," *SIAM Journal on Computing*, vol. 27, no. 6, pp. 1725–1746, 1998.

[8] C. Chekuri, N. Korula, and M. Pál, "Improved algorithms for orienteering and related problems," *ACM Transactions on Algorithms (TALG)*, vol. 8, no. 3, pp. 1–27, 2012.

[9] C. Chekuri and A. Kumar, "Maximum coverage problem with group budget constraints and applications," in *International Workshop on Randomization and Approximation Techniques in Computer Science*, Springer, 2004, pp. 72–83.

[10] K. Chen and S. Har-Peled, "The euclidean orienteering problem revisited," *SIAM Journal on Computing*, vol. 38, no. 1, pp. 385–397, 2008.

[11] V. Cohen-Addad, "Bypassing the surface embedding: Approximation schemes for network design in minor-free graphs," in *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, 2022, pp. 343–356.

[12] B. Farbstein and A. Levin, "Deadline tsp," *Theoretical Computer Science*, vol. 771, pp. 83–92, 2019.

[13] Z. Friggstad and C. Swamy, "Constant-factor approximation to deadline tsp and related problems in (almost) quasi-polytime," in *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*, Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

[14] L.-A. Gottlieb, R. Krauthgamer, and H. Rika, "Faster algorithms for orienteering and k-tsp," *Theoretical Computer Science*, vol. 914, pp. 73–83, 2022.

[15] P. Klein, S. A. Plotkin, and S. Rao, "Excluded minors, network decomposition, and multicommodity flow," in *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, 1993, pp. 682–690.

[16] J. S. Mitchell, "Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric tsp, k-mst, and related problems," *SIAM Journal on computing*, vol. 28, no. 4, pp. 1298–1309, 1999.

[17] K. Talwar, "Bypassing the embedding: Algorithms for low dimensional metrics," in *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, 2004, pp. 281–290.

[18] M. Thorup, "Compact oracles for reachability and approximate distances in planar digraphs," *Journal of the ACM (JACM)*, vol. 51, no. 6, pp. 993–1024, 2004.

[19] D. B. West *et al.*, *Introduction to graph theory*. Prentice hall Upper Saddle River, 2001, vol. 2.

[20] D. P. Williamson and D. B. Shmoys, *The design of approximation algorithms*. Cambridge university press, 2011.