Approximation Algorithms for Sequencing Problems

Viswanath Nagarajan

March 2009

Tepper School of Business
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
R. Ravi (Chair)
Gerard Cornuejols
Anupam Gupta
Michael Trick

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Algorithms, Combinatorics and Optimization.

Abstract

This thesis presents approximation algorithms for some sequencing problems, with an emphasis on vehicle routing. Vehicle Routing Problems (VRPs) form a rich class of variants of the basic Traveling Salesman Problem, that are also practically motivated. The VRPs considered in this thesis include single and multiple vehicle *Dial a Ride*, *VRP with Stochastic Demands*, *Directed Orienteering* and *Directed Minimum Latency*. Other sequencing problems studied in this thesis are *Permutation Flowshop Scheduling* and *Maximum Quadratic Assignment*.

Contents

List of Figures

Chapter 1

Introduction

Broadly speaking, a sequencing problem is an optimization problem where feasible solutions are specified by a set of orderings on some ground-set. Well-known classes of sequencing problems include machine scheduling, vehicle routing, and assignment problems. Due to the complicating nature of constraints in typical sequencing problems, most of them are $\mathcal{NP}$-complete and hence we do not expect efficient (i.e. polynomial time) exact algorithms. The two main approaches to practical solutions of such problems are (i) exact algorithms that compute the optimal solution but take exponential time in the worst case, and (ii) heuristic algorithms that run in polynomial time but find near-optimal solutions. An approximation algorithm is an efficient heuristic along with a worst-case guarantee on the quality of near-optimal solutions found by it.

This thesis presents approximation algorithms for a suite of sequencing problems, with emphasis on the class of vehicle routing problems. *Vehicle Routing Problems* (VRPs) are defined on a set of locations with distances between pairs of locations (also known as *metric space*), and involve serving a set of client requests using an available fleet of vehicles. Examples of client requests are: visiting some set of locations, moving a set of objects from their source to destination locations etc. The precise problem is determined by the nature of these client requests and additional constraints on the vehicles. More details on vehicle routing can be found in [148]. The most basic VRP is the well-studied *Traveling Salesman Problem* [129, 7] which involves computing a minimum length tour visiting all locations.

1.1 Basics

The optimization problems encountered in this thesis have the property that all feasible solutions have non-negative objective values. An optimization problem is referred to as a *minimization* or *maximization* problem depending on whether the goal is to minimize or maximize the objective value. An algorithm $\mathcal{A}$ for a minimization (resp. maximization) problem is said to be an α-approximation algorithm if for every instance $\mathcal{I}$ of the problem, $\mathcal{A}$ returns a solution with objective value at most α (resp. at least $1/\alpha$) times the optimal value of $\mathcal{I}$. The parameter α is also called the approximation guarantee or approximation ratio; α may depend on the input instance $\mathcal{I}$, and in this case it is represented as a function $\alpha(\mathcal{I})$. The approximation guarantee $\alpha = 1$ precisely for an exact algorithm; so the approximation guarantee of any algorithm is always at least 1.

We define the following three classes of algorithms based on their running time as a function of the input size n (below $c > 0$ is any constant).

- *Polynomial time:* running time is $O(n^c)$.
- *Quasi-polynomial time:* running time is $O(2^{\log^c n})$.
- *Exponential time:* running time is $O(2^{n^c})$.

As is standard, an algorithm is considered efficient if its running time is polynomial in the input size. The $\mathcal{P} \neq \mathcal{NP}$ conjecture states that $\mathcal{NP}$-complete problems do not admit exact polynomial time algorithms. However even quasi-polynomial time exact algorithms for $\mathcal{NP}$-complete problems are considered extremely unlikely.

The approximability threshold of an optimization problem refers to the best achievable approximation ratio for the problem in polynomial time, under suitable complexity-theoretic assumptions. For example, the approximability threshhold of the *minimum k-center* problem is two: it admits a 2-approximation algorithm [86], and no better guarantee is possible unless $\mathcal{P} = \mathcal{NP}$ [88]. Combinatorial optimization problems display a wide variety of approximability thresholds, as illustrated in the following.

- *Knapsack problem*: fully polynomial time approximation scheme (FPTAS) [89], weakly NP-hard.
- *Minimum makespan on identical machines*: polynomial time approximation scheme (PTAS) [87], strongly NP-hard.

- *Max-Coverage*: $\frac{e}{e-1}$ (algorithm [118, 37], hardness of approximation [94]).
- *Asymmetric k-center*: $\Theta(\log^* n)$ (algorithm [121], hardness of approximation [35]).
- *Set-Cover*: $\ln n$ (algorithm [92, 106, 36], hardness of approximation [49]).
- *Group Steiner tree on trees*: $O(\log^2 n)$-approximation algorithm [61], $\Omega(\log^{2-\epsilon} n)$ hardness of approximation for every constant $\epsilon > 0$ [77].
- *Independent Set*: trivial n-approximation algorithm, $\Omega(n^{1-\epsilon})$ hardness of approximation for every constant $\epsilon > 0$ [83].

A finite *metric space* is represented as a tuple (V, d) where V is a vertex set of finite cardinality (usually denoted $|V| = n$) and $d : V \times V \to \mathbb{R}_+$ is a distance function that satisfies the triangle inequality: $d(u, v) + d(v, w) \geq d(u, w)$ for all $u, v, w \in V$. The metric is said to be *symmetric* if $d(u, v) = d(v, u)$ for all $u, v \in V$; otherwise the metric is called *asymmetric*. For a symmetric (resp. asymmetric) metric (V, d) and subset $S \subseteq \binom{V}{2}$ (resp. $S \subseteq V \times V$), we define $d(S) := \sum_{e \in S} d(e)$. Unless mentioned otherwise, we only deal with symmetric metrics. We also assume (by scaling) that the smallest non-zero distance in any metric is at least one.

1.2 Thesis Contribution and Results

The main focus of this thesis is on Vehicle Routing Problems. The VRPs that we consider capture various aspects such as capacity constraints, multiple vehicles, transshipment, uncertain demands, and asymmetric distances.

We start with the *single vehicle Dial-a-Ride* problem in Chapter 2, where a set of objects need to be transported from their sources to respective destinations by means of a single capacitated vehicle. Then we study the Dial-a-Ride problem (Chapter 3) when *multiple vehicles* are available to move objects and the objective is to minimize the makespan of the resulting schedule. The setting in multi-vehicle Dial-a-Ride permits transshipment (also called preemption) of objects at intermediate vertices. On the other hand, the single vehicle Dial-a-Ride problem does not allow transshipment of objects. The third problem we study, *stochastic VRP* (Chapter 4), models demand uncertainty in the basic capacitated vehicle routing problem (CVRP). Here the algorithm only has access to a distribution on the demands, and the true demand at any vertex is observed only when that vertex is visited. In the final chapter on

vehicle routing (Chapter 5) we generalize two well-studied VRPs (*orienteering* and *minimum latency* problems) to the case where the underlying metric is asymmetric.

A secondary goal in this thesis is the study of other sequencing problems, and in this direction we consider the permutation flowshop scheduling and maximum quadratic assignment problems. *Permutation flowshop scheduling* (Chapter 6) is a classic machine scheduling problem that involves computing an optimal order for jobs to enter a flowshop. Finally, Chapter 7 studies the *maximum quadratic assignment* problem which is a basic problem in combinatorial optimization generalizing several known problems (eg. traveling salesman, linear arrangement, dense k subgraph). Given two $n \times n$ symmetric non-negative matrices, the objective here is to permute the two matrices so as to maximize the resulting dot-product.

We now describe these problems in some more detail, and state the main results obtained. Formal definitions, related work and detailed results appear in the respective chapters. The first four chapters are on vehicle routing.

Single Vehicle Dial a Ride. Chapter 2 studies the basic non-preemptive Dial-a-Ride problem. Given an n-vertex metric space (V, d), a depot $r \in V$, a set of m demand-pairs $\{(s_i, t_i)\}_{i=1}^{m}$, and a single vehicle of capacity k, the goal is to find a minimum length tour of the vehicle starting (and ending) at r that moves each object i from its source s_i to destination t_i such that the vehicle carries at most k objects at any point on the tour. It is required that the tour be *non-preemptive*, i.e. once an object is picked up from its source, it remains in the vehicle until dropped at its destination. We give an $O(\sqrt{\min\{n, k\}} \cdot \log^2 n)$-approximation algorithm for this problem, that improves the previously best known guarantee (in terms of n) of $O(\sqrt{k} \cdot \log n)$ [28]. An interesting aspect of our approach is that it also gives similar approximation guarantees for substantially more general cost functions. We also consider the effect of the number of preemptions in single vehicle Dial-a-Ride, and show that for every Dial-a-Ride instance, there is a tour that preempts each object *at most once* and has length at most $O(\log^2 n)$ times the optimal tour that may preempt arbitrarily. On the other hand, even for Dial-a-Ride instances on the Euclidean plane, there is an $\tilde{\Omega}(n^{1/8})$ gap between optimal preemptive and non-preemptive tours. Hence the major difference in tour-length occurs between zero and one preemption per object.

Multi-Vehicle Dial a Ride. In Chapter 3, we consider the Dial-a-Ride problem with multiple identical vehicles (i.e. same speed and capacity) where the vehicles may have different depot-locations. Here we consider the *preemptive* version where an object may be left at intermediate vertices while being transported from source

to destination, and multiple vehicles may be involved in moving the same object. The goal is to compute a schedule of all vehicles that together move objects from their sources to destinations. We give an $O(\log^3 n)$-approximation algorithm for minimizing the maximum completion time (a.k.a. makespan) in preemptive multi-vehicle Dial-a-Ride. There is an $\Omega(\log^{1/4-\epsilon} n)$ hardness of approximation for even preemptive single vehicle Dial-a-Ride [66]. We also give improved approximation ratios in the following two special cases: when the underlying metric is induced by a graph excluding some fixed minor, and when there is no capacity constraint.

Stochastic Demands VRP. In the stochastic vehicle routing problem (SVRP), a single vehicle of capacity $Q \in \mathbb{N}$ is used to distribute units of an identical item from a depot to vertices in an n-vertex metric (V, d), where demands are uncertain and represented by some probability distribution $\mathcal{D}$ over $\{0, 1, \cdots, Q\}^V$. The exact demand at a vertex is determined only when that vertex is visited. The objective is to compute a strategy of visiting vertices (starting and ending at the depot) such that all realized demands are satisfied and the *expected tour length* is minimized. We note that this strategy may be *adaptive*, i.e. at any point in the tour, the next vertex to visit may depend on the demands observed until then. The complexity of SVRP depends on the representation of distribution $\mathcal{D}$. The most general setting is where we are only given black-box access to the distribution $\mathcal{D}$; however it turns out that no $o(n)$ approximation ratio is possible in this generality. We consider two natural ways of describing $\mathcal{D}$, that make SVRP more tractable.

- **Explicit demand distribution.** Here $\mathcal{D}$ is specified by m demand scenarios, where each scenario $i \in [m]$ specifies demands $q_v^i \in \{0, 1, \cdots, Q\}$ at all vertices $v \in V$ and the probability p_i of its occurrence (where $\sum_{i=1}^m p_i = 1$). We give an $O(\log^2 n \cdot \log m)$ approximation algorithm in this setting, using a connection to the *group Steiner tree problem* [61]. We also show that this problem is at least as hard to approximate as *'latency group Steiner tree'*, for which $O(\log^2 n)$ is the best known approximation ratio, and there is an $\Omega(\log^{1-\epsilon} n)$ hardness of approximation [77].

- **Independent demand distribution.** Here $\mathcal{D}$ is specified by means of a demand random variable ξ_v (in the range $\{0, 1, \cdots, Q\}$) at each vertex $v \in V$, where the random variables $\{\xi_v\}_{v \in V}$ are independent of each other. We obtain a simple randomized approximation algorithm achieving the following worst-case guarantees: $(1 + \alpha)$ under *split-delivery* VRP (demands can be served in multiple visits), and $(2 + \alpha)$ for *unsplit-delivery* VRP (each demand must

be served in a single visit); above α denotes the best approximation ratio for the traveling salesman problem. This result matches (up to an additive $o(1)$ term) the corresponding best known guarantees for the deterministic versions [5, 6]. Moreover, our algorithm produces an *a priori* strategy that always visits vertices in the same order.

Vehicle Routing on Asymmetric Metrics. We study two vehicle routing problems on asymmetric metrics in Chapter 5. The symmetric counterparts of these problems are very well-studied and (small) constant factor approximation ratios are known. However, the problems are considerably harder in asymmetric metrics. First we consider the *directed orienteering* problem, that involves computing a bounded-length path from specified origin to destination vertices, visiting the maximum number of vertices. We provide an $O(\log^2 n)$-approximation algorithm for this problem, which is the first polynomial-time poly-logarithmic approximation guarantee. Combined with previously known reductions, this result also implies poly-logarithmic approximation ratios for other directed VRPs such as vehicle routing with time-windows. In the second part of this chapter, we study the *directed latency* problem: given an asymmetric metric and a depot vertex, the goal is to compute a tour originating at the depot that minimizes the sum of arrival times at all vertices. This is a variant of the well-known asymmetric traveling salesman problem (ATSP), where the objective is minimizing total tour length. We give an LP-based reduction (in quasi-polynomial time) from directed latency to the related *asymmetric traveling salesman path* problem (ATSP-path) that implies an $O(\rho \cdot \log^3 n)$-approximation algorithm for directed latency, where ρ is the integrality gap of a natural LP-relaxation to ATSP-path. We conjecture that $\rho = O(\log n)$; however the best bound that we obtain here is $\rho = O(\sqrt{n})$. We also show that the directed latency problem is at least as hard to approximate as ATSP, for which $O(\log n)$ is the best known approximation ratio.

The last two chapters concern other well-known sequencing problems.

Permutation Flowshop Scheduling. In a flowshop, there are m machines located in order 1 through m, and n jobs each of which consists of a sequence of operations on the machines (in the fixed order 1 to m). A permutation schedule involves processing all jobs subject to the constraint that each machine process all the jobs in the *same* order. The goal in permutation flowshop scheduling is to compute a schedule (which corresponds to a permutation on the jobs) that minimizes the completion time of the last job (i.e. makespan). In Chapter 6 we obtain an $O(\sqrt{\min\{m, n\}})$-

approximation algorithm for the permutation flowshop problem, relative to the trivial lower-bounds of maximum job-length and machine-load. This result also bounds the worst-case gap between permutation and 'non-permutation' schedules in the flowshop problem by $\Theta(\sqrt{\min\{m,n\}})$; instances showing an $\Omega(\sqrt{\min\{m,n\}})$ gap were known earlier [124]. Furthermore, the algorithm for minimizing makespan can be used to obtain an $O(\sqrt{\min\{m,n\}})$-approximation algorithm for the weighted completion time objective, improving substantially over the previously best known bound of $\epsilon \cdot m$ (for any constant $\epsilon > 0$) [142].

Maximum Quadratic Assignment. The input to the quadratic assignment problem [22] consists of two $n \times n$ symmetric non-negative matrices $W = (w_{i,j})$ and $D = (d_{i,j})$. The objective in maximum quadratic assignment (Max-QAP) is to obtain a permutation $\pi : [n] \to [n]$ that maximizes the quantity $\sum_{i,j \in [n], i \neq j} w_{i,j} \cdot d_{\pi(i),\pi(j)}$. We present an $O(\sqrt{n} \cdot \log^2 n)$-approximation algorithm for Max-QAP, which is the first non-trivial approximation guarantee for this problem. We note that Max-QAP generalizes the notorious dense-k-subgraph problem, for which $n^{1/3-\delta}$ (here $\delta > 0$ is a small fixed constant) is the best known approximation ratio. We also consider the special case when one of the matrices W or D satisfies triangle inequality (i.e. it represents distances in a symmetric metric); in this case we obtain a $\frac{2e}{e-1}$-approximation ratio, improving over the previously best-known bound of four [9].

1.3 Thesis Outline

We now discuss some common threads between different chapters and provide an outline for reading this thesis. As mentioned earlier, Chapters 2–5 are on vehicle routing, Chapter 6 is on permutation flowshop scheduling, and Chapter 7 is on maximum quadratic assignment. Moreover, Chapters 2–4 deal with VRPs on symmetric metrics, whereas Chapter 5 deals with asymmetric metrics.

There is a natural progression from Chapter 2 (single vehicle Dial-a-Ride) to Chapter 3 (Dial-a-Ride with multiple vehicles). Additionally, some structural properties of Dial-a-Ride tours (from Chapter 2) are used in Chapter 3. It is suggested that these two chapters be read in that order. The other chapters are self contained and can be read independently of each other.

Single vehicle Dial-a-Ride (Chapter 2) and stochastic demands VRP (Chapter 4) both generalize the classic capacitated VRP, albeit in different ways. Dial-a-Ride

incorporates multicommodity demands, and SVRP models probabilistic demands. Not surprisingly, the Haimovich-Kan [73] algorithm for capacitated vehicle routing is used/extended in both these chapters.

The main problems of study in Chapter 5 (asymmetric VRPs) are orienteering and minimum latency, in directed graphs. Both these objectives are also encountered in Chapter 4, however in the context of the group Steiner tree problem (in symmetric metrics). Hence the analysis in these two chapters bear some similarity.

The well-known dense-k-subgraph problem is generalized in different ways in Chapters 2 and 7. Given an undirected graph G and bound k, dense-k-subgraph involves choosing k vertices in G that induce the maximum number of edges. Maximum quadratic assignment (Chapter 7) is a generalization that involves embedding any arbitrary graph (instead of just the k-clique) onto another so as to maximize the number of common edges. The main subroutine used in the single vehicle Dial-a-Ride algorithm of Chapter 2 is the *k-forest* problem, which is another extension of dense-k-subgraph. In k-forest, the cost of choosing any vertex-subset comes from an arbitrary underlying metric (instead of being the size of this subset).

The permutation flowshop scheduling problem (Chapter 6) and the k-forest problem (Chapter 2) are both related to increasing subsequences in permutations. This connection is utilized in obtaining algorithms for both these problems. Furthermore, the approach used for minimizing weighted completion time in permutation flowshop is similar to that for directed minimum latency (Chapter 5).

Chapter 2

Single vehicle Dial-a-Ride

2.1 Introduction

This chapter studies the single vehicle Dial-a-Ride problem, where given a metric space with objects having sources and destinations, and a vehicle of some capacity k, the goal is to find a route for this vehicle so that each object can be taken from its source to destination without exceeding the capacity of the vehicle at any point, such that the length of the vehicle route is minimized. We require that the route be *non-preemptive*, i.e. once an object is picked up from its source, it remains in the vehicle until delivered to its destination. It turns out that the non-preemptive Dial-a-Ride problem is closely related to a generalization of the *Steiner forest* problem called k*-forest*, that we now introduce.

In the Steiner forest problem, we are given a set of vertex-pairs in a metric, and the goal is to find a forest such that each vertex-pair is connected in the forest. This is a generalization of the *Steiner tree* problem, where all the pairs contain a common vertex called the root; both the tree and forest versions are well-understood fundamental problems in network design, and constant factor approximation algorithms are known [130, 4, 68]. An important extension of the Steiner tree problem studied in the late 1990s was the k*-MST* problem, where one sought the least-cost tree that connected any k of the terminals: several approximation algorithms were given for the problem, culminating in the 2-approximation of [63]. The k-MST problem proved crucial in many subsequent developments in network design and vehicle routing [29, 45, 17, 13]. One can analogously define the k-forest problem where one needs to connect *only* k *of the pairs* in some Steiner forest instance: surprisingly,

very little is known about this problem, which was first studied formally only recently [74, 138]. We give simpler and improved approximation algorithms for the k-forest problem, and show how this implies good approximation bounds for the Dial-a-Ride problem.

2.1.1 The k-Forest Problem

Our starting point is the k-forest problem.

Definition 1. *Given an n-vertex metric space (V, d), and* demand pairs $\{s_i, t_i\}_{i=1}^{m} \subseteq V \times V$*, find the least-cost subgraph that connects at least k pairs.*

We note that demand pairs may be repeated; so k and m may be superpolynomial in n. The k-forest problem is also a generalization of the (minimization version of the) well-studied *dense-k-subgraph problem*, where given a graph $G = (V, E)$ and a target $k \leq |E|$, the goal is to compute a minimum cardinality subset $U \subseteq V$ of vertices that induce at least k edges. Despite several attempts, nothing better than an $O(n^{1/3-\delta})$-approximation is known for dense-k-subgraph ($\delta > 0$ is some fixed constant); improving upon this is a long-standing open question. The k-forest problem was first defined in [74], and the first non-trivial approximation was given by [138], who gave an algorithm with an approximation guarantee of $O(\min\{n^{2/3}, \sqrt{m}\} \log n)$. We obtain the following improved approximation guarantee for k-forest in Section 2.2.

Theorem 2. *There is an $O(\min\{\sqrt{n} \cdot \log k, \sqrt{k}\})$-approximation algorithm for the k-forest problem.*

The proof of this theorem involves two algorithms, both reducing the k-forest problem to the k-MST problem in different ways and achieving different approximation guarantees. The first algorithm (giving an approximation of $O(\sqrt{k})$) uses the k-MST algorithm to find good solutions on the sources and the sinks independently, and then uses the Erdős-Szekeres theorem on monotone subsequences to find a "good" subset of these sources and sinks to connect cheaply; details are given in Section 2.2.1. The second algorithm starts off with a single vertex as the initial solution, and uses the k-MST algorithm to repeatedly find a low-cost tree that satisfies a large number of pairs having one endpoint in the current solution and the other endpoint outside; this tree is then used to greedily augment the current solution and proceed. Choosing the parameters (as described in Section 2.2.2) gives us an $O(\sqrt{n})$ approximation.

2.1.2 The Dial-a-Ride Problem

The Dial-a-Ride problem is formally defined as follows.

Definition 3. *Given an n-vertex metric space (V, d), a starting vertex (or* root*) r, a set of m objects with source-destination pairs $\{(s_i, t_i)\}_{i=1}^{m}$, and a vehicle of capacity k, find a minimum length tour of the vehicle starting (and ending) at r that moves each object i from its source s_i to its destination t_i such that the vehicle carries at most k objects at any point on the tour.*

Each demand-pair in the Dial-a-Ride problem corresponds to an object that is to be moved from the specified source to destination. We use the terms: demand-pair, object, and pair interchangeably. In the *preemptive Dial-a-Ride* problem, after picking up an object from its source, it may be left at some intermediate vertices before being delivered to its destination. In this paper we will mainly be concerned with the *non-preemptive Dial-a-Ride* problem, where once an object is picked up from its source, it remains in the vehicle until dropped at its destination. A note on the parameters: using triangle inequality, any feasible non-preemptive tour can be short-cut over vertices that do not participate in any demand-pair; hence we can assume that every vertex is an end point of some demand-pair, i.e. $n \leq 2m$. Again, we allow multiple demand-pairs between the same pair of vertices; so the number of objects m and the vehicle capacity k may be larger than any polynomial in n.

We now mention two lower bounds for the preemptive Dial-a-Ride problem that are used in Chapters 2 and 3. The *Steiner lower bound* is the minimum length TSP tour on the set of all sources and destinations; and the *flow lower bound* equals $\sum_{i=1}^{m} \frac{d(s_i,t_i)}{k}$. Note that for any Dial-a-Ride instance, the cost of an optimal preemptive tour is at most that of an optimal non-preemptive tour.

The approximability of the non-preemptive Dial-a-Ride problem is not very well understood: the previous best upper bound is an $O(\sqrt{k}\log n)$-approximation algorithm due to [28], whereas the best lower bound that we are aware of is APX-hardness. We establish the following (somewhat surprising) connection between the Dial-a-Ride and k-forest problems in Section 2.3.

Theorem 4. *Given an α-approximation algorithm for k-forest, there is an $O(\alpha \cdot \log^2 n)$-approximation algorithm for the Dial-a-Ride problem.*

In particular, combining Theorems 2 and 4 gives us an $O(\min\{\sqrt{k}, \sqrt{n}\} \cdot \log^2 n)$-approximation guarantee for Dial-a-Ride. Of course, improving the approximation guarantee for k-forest would improve the result for Dial-a-Ride as well.

Our results match the previous best bound up to a logarithmic term, and give an improvement when the vehicle capacity $k \gg n$, the number of nodes. More interestingly, our algorithm for Dial-a-Ride easily extends to generalizations of the Dial-a-Ride problem. In particular, we consider a substantially more general vehicle routing problem where the vehicle has no *a priori* capacity, and instead the cost of traversing each edge e is an arbitrary non-decreasing function $c_e(l)$ of the number of objects l in the vehicle; setting $c_e(l)$ to the edge-length d_e when $l \leq k$, and $c_e(l) = \infty$ for $l > k$ gives us back the classical Dial-a-Ride setting. In Section 2.3.2, we show that this general *non-uniform Dial-a-Ride* problem admits an $O(\sqrt{n} \cdot \log^2 m)$ approximation guarantee. Another extension we consider is the *weighted Dial-a-Ride* problem. In this, each object may have a different size, and the total size of the items in the vehicle must be bounded by the vehicle capacity; this is also known as the *pickup and delivery* problem [133]. We show in Section 2.3.3 that this problem can be reduced to the (unweighted) Dial-a-Ride problem at the loss of only a constant factor in the approximation guarantee: so weighted Dial-a-Ride admits an $O(\sqrt{n} \log^2 n)$-approximation algorithm.

We also consider the effect of preemptions in the Dial-a-Ride problem (Section 2.4). It was shown in [28] that the gap between the optimal preemptive and non-preemptive tours could be as large as $\Omega(n^{1/3})$. We show that the real difference arises between *zero* and *one* preemptions: allowing multiple preemptions does not give us much added power. In particular, we show in Section 2.4 that for any instance of the Dial-a-Ride problem, there is a tour that preempts each object *at most once* and has length at most $O(\log^2 n)$ times an optimal preemptive tour (which may preempt each object an arbitrary number of times). Motivated by obtaining a better guarantee for Dial-a-Ride on the Euclidean plane, we study the preemption gap in such instances. We show that even in this case, there are instances having an $\tilde{\Omega}(n^{1/8})$ gap between optimal preemptive and non-preemptive tours. This preemption gap relies on the connection between the Dial-a-Ride and k-forest.

2.1.3 Related Work

The k-forest problem: The k-forest problem is relatively new: it was defined by [74]. An $\tilde{O}(k^{2/3})$-approximation algorithm for even the directed k-forest problem can be inferred from [27]. Recently, [138] gave an $O(\min\{n^{2/3}, \sqrt{m}\} \log n)$ approximation algorithm for k-forest. The k-forest problem is a generalization of k-MST, for which a 2-approximation is known [63].

Dense k-subgraph: As shown in [74], the *k-forest* problem also generalizes the dense-k-subgraph problem [52]. The best known approximation guarantee for the dense-k-subgraph problem is $O(n^{1/3-\delta})$ where $\delta > 0$ is some constant, due to [52], and obtaining an improved guarantee has been a long standing open problem. Strictly speaking, [52] study a potentially harder problem: the *maximization* version of dense-k-subgraph, where one wants to pick k vertices to maximize the number of edges in the induced graph. However, nothing better is known even for the *minimization* version of dense-k-subgraph (where one wants to pick the minimum number of vertices that induce k edges). Moreover, the approximability of these two versions of dense-k-subgraph are polynomially related [74]. The minimization dense-k-subgraph problem on graph G reduces to k-forest by considering an unweighted star-metric, with leaves corresponding to vertices of G and pairs corresponding to edges of G.

Dial-a-Ride: Dial-a-Ride problems form an interesting subclass of Vehicle Routing Problems that are well studied in the operations research literature. Savelsberg and Sol [133] and Cordeau and Laporte [39] survey several variants of non-preemptive Dial-a-Ride problems that have been studied in the literature.

While the Dial-a-Ride problem has been studied extensively in the operations research literature, relatively little is known about its approximability. The currently best known approximation ratio for non-preemptive Dial-a-Ride is $O(\sqrt{k}\log n)$ due to [28]. We note that their algorithm assumes instances with unweighted objects. [100] give a 3-approximation algorithm for the Dial-a-Ride problem on a *line metric*; in fact, their algorithm finds a non-preemptive tour that has length at most 3 times the lower bounds for the preemptive version. A 2.5-approximation algorithm for *single source* special case of Dial-a-Ride (also called the *capacitated vehicle routing* problem) was given in [73]; again, this algorithm outputs a non-preemptive tour with length at most 2.5 times the preemptive lower bounds. The $k = 1$ special case of Dial-a-Ride is also known as the *stacker-crane* problem, for which a 1.8-approximation is known [58]. For the *preemptive* Dial-a-Ride problem, [28] gave the current-best $O(\log n)$ approximation algorithm, and [66] showed that it is $\Omega(\log^{1/4-\epsilon} n)$ hard to approximate. Recall that no super-constant hardness results are known for the non-preemptive Dial-a-Ride problem.

2.2 Algorithms for the k-forest problem

In this section, we study the k-forest problem, and give an approximation guarantee of $O(\min\{\sqrt{n}, \sqrt{k}\})$. Our approach is based on approximating the following "density" variant of k-forest.

Definition 5. Minimum-ratio k-forest. *Given an n-vertex metric space (V, d), m pairs of vertices $\{s_i, t_i\}_{i=1}^m$, and a target k, find a tree T that connects* at most *k pairs, and minimizes the ratio of the length of T to the number of pairs connected in T.*

Observe that given any forest F connecting some set of pairs, one of the trees in F has ratio (length to number of connected pairs) at most that of F. Hence even if we relax the above definition to consider any forest, the optimal ratio solution can be assumed to be a tree. Given any feasible solution T to minimum-ratio k-forest, $\mathsf{Ratio}(T)$ denotes the ratio of length of T to the number of pairs connected in T.

We present two different algorithms for *minimum-ratio k-forest*, obtaining approximation guarantees of $O(\sqrt{k})$ (Section 2.2.1) and $O(\sqrt{n})$ (Section 2.2.2); these are then combined to give the claimed result for the k-forest problem. Both our algorithms are based on reductions to the k-MST problem, albeit in very different ways. As is usual, when we say that our algorithm *guesses* a parameter in the following discussion, it means that the algorithm is run for each possible value of that parameter, and the best solution found over all the runs is returned. As long as only a constant number of parameters are being guessed and the number of possibilities for each of these parameters is polynomial, the algorithm is repeated only a polynomial number of times.

2.2.1 An $O(\sqrt{k})$ approximation algorithm

In this section, we give an $O(\sqrt{k})$ approximation algorithm for minimum ratio k-forest, which is based on a simple reduction to the k-MST problem. The idea is to look at the optimal solution S to minimum-ratio k-forest and consider an Euler tour of this tree S—a theorem of Erdős and Szekeres on increasing subsequences implies that there must be at least $\sqrt{|S|}$ sources which are visited in the same order as the corresponding sinks. We use this existence result to combine the source-sink pairs to create an instance of $\sqrt{|S|}$-MST from which we can obtain a good solution; the details follow. Below S denotes an optimal ratio tree, that covers q pairs and has length B; let D denote the largest distance between any demand-pair that is covered in S (note $D \leq B$).

The $O(\sqrt{k})$ approximation algorithm proceeds as below. Define a new metric l on the set $\{1, \cdots, m\}$ of pairs as follows. The distance between pairs i and j, $l_{i,j} \doteq d(s_i, s_j) + d(t_i, t_j)$, where (V, d) is the original metric. This metric represents solutions of a special structure: any tree M covering pairs Π can be expressed as $M = M_s \cup M_t \cup \{f\}$ where M_s (resp. M_t) is a tree connecting all sources (resp. destinations) in Π and f is any edge connecting a pair in Π. The algorithm guesses the number of pairs q and the largest demand-pair distance D in the optimal tree S (there are at most m choices for each of q and D). The algorithm discards all pairs (s_i, t_i) such that $d(s_i, t_i) > D$ (all the pairs covered in the optimal solution S still remain). Then the algorithm runs the unrooted k-MST algorithm [63] with target $\lfloor\sqrt{q}\rfloor$, in the metric l, to obtain a tree T on the pairs P. From T, we easily obtain trees T_1 (on all sources in P) and T_2 (on all sinks in P) in metric d such that $d(T_1) + d(T_2) = l(T)$. Finally the algorithm outputs the tree $T' = T_1 \cup T_2 \cup \{e\}$, where e is any edge joining a source in T_1 to its corresponding sink in T_2.

Due to the pruning on pairs that have large distance, $d(e) \leq D$ and the length of T', $d(T') \leq l(T) + D \leq l(T) + B$. We now argue that the cost of the solution T found by the k-MST algorithm $l(T) \leq 8B$. Consider the optimal ratio tree S (in metric d) that has q pairs $\{(s_1, t_1), \cdots, (s_q, t_q)\}$, and let τ denote an Euler tour of S. Suppose that in a traversal of τ, the *sources* of pairs in S are seen in the order $s_1, \cdots, s_q$. Then in the same traversal, the *sinks* of pairs in S will be seen in the order $t_{\pi(1)}, \cdots, t_{\pi(q)}$, for some permutation π. The following fact is well known (see, e.g., [144]).

Theorem 6. (Erdős and Szekeres) *Every permutation on $\{1, \cdots, q\}$ has either an increasing subsequence of length $\lfloor\sqrt{q}\rfloor$ or a decreasing subsequence of length $\lfloor\sqrt{q}\rfloor$.*

Using Theorem 6, we obtain a set M of $p = \lfloor\sqrt{q}\rfloor$ pairs such that (1) the sources in M appear in increasing order in a traversal of the Euler tour τ, and (2) the sinks in M appear in increasing order in a traversal of either τ or τ^R (the reverse traversal of τ). Let $j_0 < j_1 < \cdots < j_{p-1}$ denote the pairs in M in increasing order. From statement (1) above, $\sum_{i=0}^{p-1} d(s(j_i), s(j_{i+1})) \leq d(\tau)$, where the indices in the summation are modulo p. Similarly, statement (2) implies that $\sum_{i=0}^{p-1} d(t(j_i), t(j_{i+1})) \leq \max\{d(\tau), d(\tau^R)\} = d(\tau)$. Thus we obtain:

$$\sum_{i=0}^{p-1} [d(s(j_i), s(j_{i+1})) + d(t(j_i), t(j_{i+1}))] \leq 2d(\tau) \leq 4B$$

But this sum is precisely the length of the tour $j_0, j_1, \cdots, j_{p-1}, j_0$ in metric l. In other words, there is a tree of length $4B$ in metric l, that contains $\lfloor\sqrt{q}\rfloor$ vertices. So, the cost of the solution T found by the k-MST approximation algorithm is at most $8B$.

Now the final solution T' has length at most $l(T) + B \leq 9B$, and $\mathsf{Ratio}(T') \leq 9\sqrt{q}\frac{B}{q} \leq 9\sqrt{k}\frac{B}{q}$. Thus we have an $O(\sqrt{k})$ approximation algorithm for minimum ratio k-forest.

2.2.2 An $O(\sqrt{n})$ approximation algorithm

In this section, we show an $O(\sqrt{n})$ approximation algorithm for the minimum ratio k-forest problem. The approach is again to reduce to the k-MST problem; the idea is rather different: either we find a vertex v such that a large number of demand-pairs of the form $(v, *)$ can be satisfied using a small tree (the "high-degree" case), or we use a repeated greedy procedure to cover most vertices without paying too much (since we are in the "low-degree" case, covering most vertices implies covering most pairs too). The details follow.

Let S denote an optimal solution to minimum ratio k-forest, and $q \leq k$ the number of demand pairs covered in S. We define the *degree* Δ of S to be the maximum number of demand-pairs (among those covered in S) that are incident at any vertex in S. The algorithm first guesses the following parameters of the optimal solution S: its length B (within a factor 2), the number of pairs covered q, the degree Δ, and the vertex $w \in S$ that has Δ demand-pairs incident at it. Although, there may be an exponential number of choices for the optimal length, a polynomial number of guesses within a binary-search suffice to get a B such that $B \leq d(S) \leq 2 \cdot B$. The algorithm then returns the better of the two procedures described below.

Procedure 1 (high-degree case): The algorithm assigns a weight to each vertex u, equal to the number of pairs having an end point at u and the other end point at w (the guessed Δ-degree vertex in S). Then we run the k-MST algorithm [63] with root w and a target weight of Δ, resulting in a solution tree H. Since the degree of vertex w in the optimal solution S is Δ, there is tree rooted at w of length $d(S) \leq 2B$, that contains at least Δ pairs having one end point at w. Hence the k-MST instance has a feasible solution of length $2B$, and the length of solution H is at most $4B$ (since the algorithm of [63] is a 2-approximation). Thus $\mathsf{Ratio}(H) \leq 4B/\Delta = \frac{4q}{\Delta}\frac{B}{q}$.

Procedure 2 (low-degree case): Set $t = \frac{q}{2\Delta}$; note that $q \leq \frac{\Delta \cdot n}{2}$ and so $t \leq n/4$. We maintain a current tree T (which is initialized to $T \leftarrow \{w\}$), and iteratively do the following:

1. Shrink T to a single vertex s in metric (V, d), and run the k-MST algorithm [63]

with root s and a target of t new vertices. Let T_0 denote the resulting tree.

2. If T_0 has length at most $4B$, set $T \leftarrow T \cup T_0$ and continue to the next iteration.
3. If T_0 has length more than $4B$ (or if T already has all vertices) then terminate.

The tree T at the end of these iterations is output as the solution to minimum ratio k-forest. Since t new vertices are added in each iteration, the number of iterations is at most $\frac{n}{t}$; so the length of T is at most $\frac{4n}{t}B$. We now show that T contains at least $\frac{q}{2}$ demand-pairs. Consider the set $S \setminus T$ (recall, S is the optimal solution). It is clear that $|V(S) \setminus V(T)| < t$; otherwise the k-MST instance in the last iteration (with the current T) would have S as a feasible solution of length at most $2B$ (and hence would find one of length at most $4B$). So the number of pairs covered in S that have at least one end point in $S \setminus T$ is at most $|V(S) \setminus V(T)| \cdot \Delta \leq t \cdot \Delta = q/2$ (as Δ is the degree of solution S). Thus there are at least $q/2$ pairs *contained* in $S \cap T$, in particular in T. Thus T is a solution with $\mathsf{Ratio}(T) \leq \frac{4n}{t}B \cdot \frac{2}{q} = \frac{8n}{t}\frac{B}{q}$.

The better solution among H and T from the above two procedures has objective value at most $\min\{\frac{4q}{\Delta}, \frac{8n}{t}\} \cdot \frac{B}{q} = \min\{8t, \frac{8n}{t}\} \cdot \frac{B}{q} \leq 8\sqrt{n} \cdot \frac{B}{q} \leq 8\sqrt{n} \cdot \frac{d(S)}{q}$. So this algorithm is an $O(\sqrt{n})$ approximation to the minimum ratio k-forest problem.

2.2.3 Approximation algorithm for k-forest

Given the two algorithms for minimum ratio k-forest, we can use them in a standard greedy fashion (i.e., keep picking approximately minimum-ratio solutions until we obtain a forest connecting at least k pairs); the standard set cover analysis can be used to show an $O(\min\{\sqrt{n}, \sqrt{k}\} \cdot \log k)$-approximation guarantee for k-forest. The $O(\sqrt{k} \cdot \log k)$ part of the bound can be improved slightly to $O(\sqrt{k})$. This uses a tighter analysis of the greedy set-cover algorithm [27]. Lemma 1 from [27] implies the following in our context: Suppose there is an $f(k)$ approximation algorithm for minimum ratio k-forest, where $f(x)/x$ is a decreasing function of x. Then the greedy algorithm for the k-forest problem achieves an approximation guarantee of $\int_0^k f(x)/x \, dx$. Using $f(k) = O(\sqrt{k})$, we obtain an $O(\sqrt{k})$ approximation for k-forest, implying the guarantee in Theorem 2.

We note that this greedy approach to solving the k-forest problem may not even give an $o(n)$ approximation bound when k is super-polynomial in n. In this case however, our $O(\sqrt{n})$-approximation algorithm for minimum ratio k-forest can be used within the Lagrangian relaxation framework of [138] (in place of Theorem 3) to obtain an $O(\sqrt{n} \cdot \log n)$ approximation for k-forest.

2.3 Applications to Dial-a-Ride problems

In this section, we study applications of k-forest to the Dial-a-Ride problem (Definition 3), and some generalizations. A natural solution-structure for Dial-a-Ride involves servicing objects in batches of at most k each, where a batch consisting of a set S of demand-pairs is served as follows: the vehicle starts out being empty, picks up each of the $|S| \leq k$ objects from their sources, then drops off each object at its destination, and is again empty at the end. If we knew that the optimal solution has this structure, we could obtain a greedy framework for Dial-a-Ride by repeatedly finding the best 'batch' of k demand-pairs. However, the optimal solution may involve carrying almost k objects at every point in the tour, in which case it can not be decomposed to be of the above structure. In Theorem 7, we show that there is always a near optimal solution having this 'pick-drop in batches' structure. Building on Theorem 7, we obtain approximation algorithms for the classical Dial-a-Ride problem (Section 2.3.1), and two interesting extensions: non-uniform Dial-a-Ride (Section 2.3.2) and weighted Dial-a-Ride (Section 2.3.3).

Theorem 7. *Given any instance of Dial-a-Ride, there exists a feasible tour τ satisfying the following conditions:*

1. *τ can be split into a set of segments $\{S_1, \cdots, S_t\}$ (i.e., $\tau = S_1 \cdot S_2 \cdots S_t$) where each segment S_i services a set O_i of at most k objects such that S_i is a path that first picks up each object in O_i and then drops each of them.*
2. *The length of τ is at most $O(\log m)$ times the length of an optimal tour.*

Proof: Consider an optimal non-preemptive tour σ: let $d(\sigma)$ denote its length, and $|\sigma|$ denote the number of edge traversals in σ. Note that if in some visit to a vertex v in σ there is no pick-up or drop-off, then the tour can be short-cut over vertex v, and it still remains feasible. Further, due to triangle inequality, the length $d(\sigma)$ does not increase by this operation. So we may assume that each vertex visit in σ involves a pick-up or drop-off of some object. Since there is exactly one pick-up and drop-off for each object, we have $|\sigma| \leq 2m + 1$. Define the *stretch* of a demand-pair i to be the number of edge traversals in σ between the pick-up and drop-off of object i. The demand-pairs are partitioned as follows: for each $j = 1, \cdots, \lceil \log(2m) \rceil$, group G_j consists of all pairs having stretch between 2^{j-1} and 2^j. We consider each group G_j separately.

Claim 8. *For each $j = 1, \cdots, \lceil \log(2m) \rceil$, there is a tour τ_j that serves all the pairs in group G_j, satisfies condition 1 of Theorem 7, and has length at most $6 \cdot d(\sigma)$.*

Proof: Consider tour σ as a line $\mathcal{L}$, with every edge traversal in σ represented by a distinct edge in $\mathcal{L}$. Number the vertices in $\mathcal{L}$ from 0 to h, where $h = |\sigma|$ is the number of edge traversals in σ. Note that each vertex in V may be represented multiple times in $\mathcal{L}$. Each object is associated with the numbers of the vertices (in $\mathcal{L}$) where it is picked up and dropped off.

Let $r = 2^{j-1}$, and partition G_j as follows: for $l = 1, \cdots, \lceil \frac{h}{r} \rceil$, set $O_{l,j}$ consists of all objects in G_j that are picked up at a vertex numbered between $(l-1)r$ and $lr - 1$. Since every object in G_j has stretch in the interval $[r, 2r]$, every object in $O_{l,j}$ is dropped off at a vertex numbered between lr and $(l+2)r - 1$. Note that $|O_{l,j}|$ equals the number of objects in G_j carried over edge $(lr - 1, lr)$ by tour σ, which is at most k. We define segment $S_{l,j}$ to start at vertex number $(l-1)r$ and traverse all edges in $\mathcal{L}$ until vertex number $(l+2)r - 1$ (servicing all demand-pairs in $O_{l,j}$ by first picking up each object between vertices $(l-1)r$ and $lr - 1$; then dropping off each object between vertices lr and $(l+2)r - 1$), and then return (with the vehicle being empty) to vertex lr. Clearly, the number of objects carried over any edge in $S_{l,j}$ is at most the number carried over the corresponding edge traversal in σ. Also, each edge in $\mathcal{L}$ participates in at most 3 segments $\{S_{l,j} \mid 1 \leq l \leq \lceil h/r \rceil\}$, and each edge is traversed at most twice in any segment. So the total length of all segments $\{S_{l,j}\}$ is at most $6 \cdot d(\sigma)$. We define tour τ_j to be the concatenation $S_{1,j} \cdots S_{\lceil h/r \rceil, j}$. It is clear that this tour satisfies condition 1 of Theorem 7. ∎

Applying this claim to each group G_j, and concatenating the resulting tours, we obtain the tour τ satisfying condition 1 and having length at most $6 \log(2m) \cdot d(\sigma) = O(\log m) \cdot d(\sigma)$. ∎

Remark: The ratio $O(\log m)$ in Theorem 7 is almost best possible. As mentioned in [65], there are instances of Dial-a-Ride on an unweighted line, where every solution satisfying condition 1 of Theorem 7 has length at least $\Omega(\max\{\frac{\log m}{\log \log m}, \frac{k}{\log k}\})$ times the optimal non-preemptive tour. These instances consist of $n = 2^k + 1$ equally spaced vertices on a line, numbered 1 through $n+1$ from left to right, with demand-pairs $\{(j \cdot 2^i, (j+1)2^i) \mid 0 \leq i \leq k,\ 0 \leq j \leq 2^{k-i} - 1\}$. It can be seen that the optimal non-preemptive tour has length $O(n)$, whereas any tour satisfying condition 1 of Theorem 7 has length at least $\Omega(n \cdot \frac{\log n}{\log \log n})$. So, if we only use solutions of this 'pick-drop' structure, then it is not possible to obtain an approximation factor (just in terms of capacity k) for Dial-a-Ride that is better than $\Omega(k/\log k)$. The solutions found by the algorithm for Dial-a-Ride in [28] also satisfy condition 1 of Theorem 7. It is interesting to note that when the underlying metric is a hierarchically well-separated tree (HST), [28] obtain a solution of such structure having length $O(\sqrt{k})$ times the optimum, whereas there is a lower bound of $\Omega(\frac{k}{\log k})$ even for the simple

case of an unweighted line (which is not an HST).

2.3.1 Classical Dial-a-Ride

Theorem 7 suggests a greedy strategy for Dial-a-Ride, based on repeatedly finding the best batch of k objects to service. This greedy subproblem turns out to be the minimum ratio k-forest problem (Definition 5), for which we already have an approximation algorithm. The next theorem sets up this reduction.

Theorem 9. *A ρ-approximation algorithm for minimum ratio k-forest implies an $O(\rho \log^2 m)$-approximation algorithm for Dial-a-Ride.*

Proof: The algorithm for Dial-a-Ride is as follows.

1. $\mathcal{C} = \phi$.
2. Until there are no uncovered demand-pairs, do:
   (a) Solve the minimum ratio k-forest problem, to obtain a tree C covering $k_C \leq k$ new pairs.
   (b) Set $\mathcal{C} \leftarrow \mathcal{C} \cup C$.
3. For each tree $C \in \mathcal{C}$, obtain an Euler tour on C to locally service all demand-pairs (pick up all k_C objects in the first traversal, and drop them all in the second traversal). Then use a 1.5-approximate TSP tour on the sources, to connect all the local tours, and obtain a feasible non-preemptive tour.

Consider the tour τ and its segments as in Theorem 7. If the number of uncovered pairs in some iteration is m', one of the segments in τ is a solution to the minimum ratio k-forest problem of value at most $\frac{d(\tau)}{m'}$. Since we have a ρ-approximation algorithm for this problem, we would find a segment of ratio at most $O(\rho) \cdot \frac{d(\tau)}{m'}$. Now a standard set cover type argument shows that the total length of trees in $\mathcal{C}$ is at most $O(\rho \log m) \cdot d(\tau) \leq O(\rho \log^2 m) \cdot \mathsf{OPT}$, where OPT is the optimal value of the Dial-a-Ride instance. Further, the TSP tour on all sources is a lower bound on OPT, and we use a 1.5-approximate solution [34]. So the final non-preemptive tour output in step 5 above has length at most $O(\rho \log^2 m) \cdot \mathsf{OPT}$. ∎

This theorem is in fact stronger than Theorem 4 claimed earlier: any approximation algorithm for k-forest implies an algorithm with the same guarantee for minimum ratio k-forest. Note that, m and k may be super-polynomial in n. However,

we show in Section 2.3.3 that with the loss of a constant factor, even the weighted Dial-a-Ride problem can be reduced to classical Dial-a-Ride where the number of objects $m \leq n^4$. Based on this and Theorem 9, a ρ approximation algorithm for minimum ratio k-forest actually implies an $O(\rho \log^2 n)$ approximation algorithm for Dial-a-Ride. Using the approximation algorithm for minimum ratio k-forest (Section 2.2), we obtain an $O(\min\{\sqrt{n}, \sqrt{k}\} \cdot \log^2 n)$ approximation algorithm for the Dial-a-Ride problem.

Remark: If we use the $O(\sqrt{k})$ approximation for k-forest, the resulting non-preemptive tour is in fact feasible even for a $\sqrt{k}$ capacity vehicle! As noted in [28], this property is also true of their algorithm, which is based on an entirely different approach.

2.3.2 Non-uniform Dial-a-Ride

The greedy framework for Dial-a-Ride described above is actually more generally applicable than to just the classical Dial-a-Ride problem. In this section, we consider the Dial-a-Ride problem under a substantially more general class of cost functions, and show how the k-forest problem can be used to obtain an approximation algorithm for this generalization as well. In fact, the approximation guarantee we obtain by this approach matches (up to logarithmic factors) the best known for the classical Dial-a-Ride problem. Our framework for Dial-a-Ride is well suited for such a generalization since it is based on directly approximating a near-optimal solution; this approach is not too sensitive to the cost function. On the other hand, the algorithm in [28] is based on obtaining a good lower bound, which depends heavily on the cost function. Thus it is unclear whether their techniques can be extended to handle such a generalization.

Definition 10. Non-uniform Dial-a-Ride. *Given an n vertex undirected graph $G = (V, E)$, a root vertex r, a set of m demand-pairs $\{(s_i, t_i)\}_{i=1}^m$, and a non-decreasing cost function $c_e : \{0, 1, \cdots, m\} \rightarrow \mathbb{R}^+$ on each edge $e \in E$ (where $c_e(l)$ is the cost incurred by the vehicle in traversing edge e while carrying l objects), find a non-preemptive tour (starting and ending at r) of minimum* total cost *that moves each object i from s_i to t_i.*

Note that the classical Dial-a-Ride problem is a special case when the edge costs are given by: $c_e(l) = d_e$ if $l \leq k$ and $c_e(l) = \infty$ otherwise, where d_e is the edge length in the underlying metric. We may assume (without loss in generality) that for any *fixed* value $l \in [0, m]$, the edge costs $c_e(l)$ induce a metric on V. Similar to

Theorem 7, we have a near optimal solution with a 'batch' structure for the non-uniform Dial-a-Ride problem as well, which implies the algorithm in Theorem 12.

Corollary 11. *Given any instance of non-uniform Dial-a-Ride, there exists a feasible tour τ satisfying the following conditions:*

1. *τ can be split into a set of segments $\{S_1, \cdots, S_t\}$ (i.e., $\tau = S_1 \cdot S_2 \cdots S_t$) where each segment S_i services a set O_i of demand-pairs such that S_i is a path that first picks up each object in O_i and then drops each of them.*

2. *The cost of τ is at most $O(\log m)$ times the cost of an optimal tour.*

Proof: We only give a proof sketch highlighting how the proof of Theorem 7 carries over to this case. We may again assume that the number of edge traversals in the an optimal tour σ is at most $2m$: this uses triangle inequality in the edge-costs $c_e(l)$ for any fixed $l \in [m]$. The definitions of groups $\{G_j \mid j = 1, \cdots, \lceil \log(2m) \rceil\}$, and $\{O_{l,j}\}$ for each $1 \leq j \leq \lceil \log(2m) \rceil$ are identical to those in Theorem 7. The traversal $S_{l,j}$ serving any group $O_{l,j}$ has the property that the number of objects carried over any edge in $S_{l,j}$ is at most that carried over the same edge in σ: this implies that the cost of $S_{l,j}$ is at most that of σ between vertex numbers $(l-1)r$ and $(l+2)r - 1$. Finally concatenating all the local tours $S_{l,j}$, we obtain the desired property. ∎

Theorem 12. *A ρ-approximation algorithm for minimum ratio k-forest (for all values of k) implies an $O(\rho \log^2 m)$-approximation algorithm for non-uniform Dial-a-Ride. In particular, there is an $O(\sqrt{n} \log^2 m)$-approximation algorithm.*

Proof: Corollary 11 again suggests a greedy algorithm for non-uniform Dial-a-Ride based on the following *greedy subproblem*: find a set T of uncovered pairs and a path τ_0 that first picks up each object in T and then drops off each of them, such that the ratio of the cost of τ_0 to $|T|$ is minimized. However, unlike in the classical Dial-a-Ride problem, in this case the cost of path τ_0 does not come from a single metric. Nevertheless, the minimum ratio k-forest problem can be used to solve this subproblem as follows.

1. For every $k = 1, \cdots, m$:
   (a) Define length function $d_e^{(k)} = c_e(k)$ on the edges.
   (b) Solve the minimum ratio k-forest problem on metric $(V, d^{(k)})$ with bound k, to obtain tree T'_k covering $n_k \leq k$ pairs.

(c) Obtain an Euler tour T_k of T'_k that services these n_k objects, by picking up all objects in one traversal and then dropping them all in a second traversal.

2. Return the tour T_k having the smallest ratio $\frac{c(T_k)}{n_k}$ (over all $1 \leq k \leq m$).

Assuming a ρ-approximation algorithm for minimum ratio k-forest (for all values of k), we now show that the above algorithm obtains a 16ρ-approximate solution to the greedy subproblem. The cost of tour T_k in step 1c is $c(T_k) \leq 4 \cdot d^{(k)}(T'_k)$, since T_k involves traversing a tour on tree T'_k twice and the vehicle carries at most $n_k \leq k$ objects at every point in T_k. So tour T_k has $\frac{c(T_k)}{n_k} \leq 4\frac{d^{(k)}(T'_k)}{n_k} = 4 \cdot \mathsf{Ratio}(T'_k)$ (recall that $\mathsf{Ratio}(F)$ for any solution to minimum ratio k-forest is the ratio of length of F to the number of pairs connected by F). Let τ denote the optimal path for the greedy subproblem, T the set of objects that it services, and $t = |T|$. Let T_1 denote the last $\frac{3}{4}t$ objects that are picked up, and T_2 denote the first $\frac{3}{4}t$ objects that are dropped off. It is clear that $T_1 \cap T_2$ has at least $t/2$ objects; let $T' \subset T_1 \cap T_2$ be any subset with $|T'| = t/4$. Let τ' denote the portion of τ between the $\frac{t}{4}$-th pick up and the $\frac{3t}{4}$-th drop off. Note that when path τ is traversed, there are at least $\frac{t}{4}$ objects in the vehicle while traversing each edge in τ'. So the cost of τ, $c(\tau) \geq \sum_{e \in \tau'} c_e(t/4)$. Also τ' contains the end points of all objects in $T' \supseteq T_1 \cap T_2$. Hence τ' corresponds to a feasible solution F' (covering pairs T') to minimum ratio k-forest with bound $k = t/4$ in metric $d^{(t/4)}$. Solution F' has $\mathsf{Ratio}(F') = (\sum_{e \in \tau'} c_e(t/4))/\frac{t}{4} \leq \frac{4c(\tau)}{t}$. Thus the ρ-approximate solution $T'_{t/4}$ has $\mathsf{Ratio}(T'_{t/4}) \leq 4\rho\frac{c(\tau)}{t}$. So the tour $T_{t/4}$ has ratio $\frac{c(T_k)}{n_k} \leq 4 \cdot \mathsf{Ratio}(T'_k) \leq 16\rho\frac{c(\tau)}{t}$. Thus we have a 16ρ-approximation algorithm for the greedy subproblem.

Based on Corollary 11, it can now be shown (as in Theorem 9) that a ρ'-approximation algorithm for the greedy subproblem implies an $O(\rho' \cdot \log^2 m)$-approximation algorithm for non-uniform Dial-a-Ride. Using the above 16ρ-approximation for the greedy subproblem, we have the theorem. ∎

2.3.3 Weighted Dial-a-Ride

So far we worked with the unweighted version of Dial-a-Ride, where each object has the same weight. In this section, we extend our greedy framework for Dial-a-Ride to the case when objects have different sizes, and the total size of objects in the vehicle must be bounded by the vehicle capacity. Here we only extend the classical Dial-a-Ride problem and not the generalization of Section 2.3.2. The problem studied in this section is also known as the *pickup and delivery* problem [133].

Definition 13. Weighted Dial-a-Ride. *Given a vehicle of capacity $k \in \mathbb{N}$, an n-vertex metric space (V, d), a root vertex r, and a set of m objects $\{(s_i, t_i, w_i)\}_{i=1}^{m}$ (with object i having source s_i, destination t_i and an integer size $1 \leq w_i \leq k$), find a minimum length (non-preemptive) tour of the vehicle starting (and ending) at r that moves each object i from its source to its destination such that the total size of objects carried by the vehicle is at most k at any point on the tour.*

The classical Dial-a-Ride problem is a special case when $w_i = 1$ for all objects. The main result of this section (Theorem 15) reduces weighted Dial-a-Ride to the classical Dial-a-Ride problem with the additional property that the number m of objects is small (polynomial in the number of vertices n). This shows that in order to approximate weighted Dial-a-Ride, it suffices to consider instances of the classical Dial-a-Ride problem with a small number of objects. The next lemma shows that even if the vehicle is allowed to split each object over multiple deliveries, the resulting tour is *not* much shorter than the tour where each object is required to be served in a single delivery (as is the case in weighted Dial-a-Ride). This lemma is the main ingredient in the proof of Theorem 15. In the following, for any instance of weighted Dial-a-Ride, we define the *unweighted instance* corresponding to it as a classical Dial-a-Ride instance with vehicle capacity k, having w_i (unweighted) objects with source s_i and destination t_i (for each $1 \leq i \leq m$).

Lemma 14. *Given any instance $\mathcal{I}$ of weighted Dial-a-Ride, and a solution τ to the unweighted instance corresponding to $\mathcal{I}$, there is a polynomial time computable solution to $\mathcal{I}$ having length at most $O(1) \cdot d(\tau)$.*

Proof: Let $\mathcal{J}$ denote the unweighted instance corresponding to $\mathcal{I}$. Define line $\mathscr{L}$ as in the proof of Theorem 7 constructed by traversing τ from r: for every edge traversal in τ, add a new edge of the same length at the end of $\mathscr{L}$. Note that there is a 1-1 correspondence between edges in $\mathscr{L}$ and edge-traversals in τ. For each unweighted object in $\mathcal{J}$ corresponding to object i in $\mathcal{I}$, there is a segment in τ (correspondingly in $\mathscr{L}$) where it is moved from s_i to t_i. So each object $i \in \mathcal{I}$ corresponds to w_i segments in τ (each being a path from s_i to t_i). For each object i in $\mathcal{I}$, we assign i to one of its w_i segments picked uniformly at random: call this segment l_i. For an edge $e \in \mathscr{L}$, let $N_e = \sum_{i:e \in l_i} w_i$ denote the random variable which equals the *total weight* of objects whose assigned segments contain e. Note that the expected value of N_e is exactly the number of unweighted objects carried by τ when traversing the edge corresponding to e. Since τ is a feasible tour for $\mathcal{J}$, $E[N_e] \leq k$ for all $e \in \mathscr{L}$.

Consider a random instance $\mathcal{R}$ of Dial-a-Ride on line $\mathscr{L}$ with vehicle capacity k and objects as follows: for each object i in $\mathcal{I}$, an object of weight w_i is to be moved along segment l_i (chosen randomly as above). Clearly, any feasible tour for $\mathcal{R}$ corresponds to a feasible tour for $\mathcal{I}$ of the same length. Note that the flow lower bound for instance $\mathcal{R}$ is $F = \sum_{e \in \mathscr{L}} d_e \frac{N_e}{k}$, and the Steiner lower bound is $\sum_{e \in \mathscr{L}} d_e = d(\tau)$. Using linearity of expectation, $E[F] \leq \sum_{e \in \mathscr{L}} d_e \frac{E[N_e]}{k} \leq \sum_{e \in \mathscr{L}} d_e = d(\tau)$. Let R^* denote the Dial-a-Ride instance on line $\mathscr{L}$ obtained by assigning each object i in $\mathcal{I}$ to the segment corresponding to it (among its w_i segments) that has the smallest number of edges. Clearly this assignment minimizes the flow lower bound (over all assignments of objects to segments). So R^* has flow bound $\leq E[F] \leq d(\tau)$, and Steiner lower bound $d(\tau)$.

Finally, we note that the 3-approximation algorithm for Dial-a-Ride on a line [100] extends to a constant factor approximation algorithm for the case with weighted objects as well (this can be seen directly from [100]). Additionally, this approximation guarantee is relative to the preemptive lower bounds. Thus, using this algorithm on R^*, we obtain a feasible solution to $\mathcal{I}$ of length at most $O(1) \cdot d(\tau)$. ∎

Theorem 15. *Suppose there is a ρ-approximation algorithm for instances of classical Dial-a-Ride with at most $O(n^4)$ objects. Then there is an $O(\rho)$-approximation algorithm for weighted Dial-a-Ride (with any number of objects). In particular, there is an $O(\sqrt{n}\log^2 n)$ approximation for weighted Dial-a-Ride.*

Proof: Let $\mathcal{I}$ denote an instance of weighted Dial-a-Ride with objects $\{(w_i, s_i, t_i) : 1 \leq i \leq m\}$, and τ^* an optimal tour for $\mathcal{I}$. Let $\mathcal{P} = \{(s_1, t_1), \cdots, (s_l, t_l)\}$ be the distinct pairs of vertices that have some demand-pair between them, and let T_i denote the total size of *all* objects having source s_i and destination t_i. Note that $l \leq n(n-1)$. Let $\mathcal{P}_{high} = \{i \in \mathcal{P} : T_i \geq \frac{k}{2}\}$, $\mathcal{P}_{low} = \{i \in \mathcal{P} : T_i \leq \frac{k}{l}\}$, and $\mathcal{P}' = \mathcal{P} \setminus (\mathcal{P}_{high} \cup \mathcal{P}_{low})$. We now show how to separately service objects in $\mathcal{P}_{low}$, $\mathcal{P}_{high}$ and $\mathcal{P}'$.

Servicing $\mathcal{P}_{low}$: The total size in $\mathcal{P}_{low}$ is at most k; so we can service all these pairs by using a 1.5-approximate tour [34] on sources and destinations, and traversing it twice: once to pick up all objects and once to drop them. Note that the length of this tour is at most 3 times the Steiner lower bound, hence at most $3 \cdot d(\tau^*)$.

Servicing $\mathcal{P}_{high}$: Let C be a 1.5-approximate minimum tour on all the sources. The pairs in $\mathcal{P}_{high}$ are serviced by a tour τ_1 as follows. Traverse along C, and when a source s_i in $\mathcal{P}_{high}$ is visited, traverse the direct edge to the corresponding destination t_i and back, as few times as possible so as to move all the objects between s_i and t_i, as described next. Note that every object to be moved between s_i and t_i has size

(the original w_i size) at most k, and the total size of such objects $T_i \geq k/2$. So these objects can be partitioned such that the size of each part (except possibly the last) is in the interval $[\frac{k}{2}, k]$. So the number of times edge (s_i, t_i) is traversed to service the demand-pairs between them is at most $2\lceil \frac{2T_i}{k} \rceil \leq 2(\frac{2T_i}{k} + 1) \leq 8\frac{T_i}{k}$. Now, the length of tour τ_1 is at most $d(C) + \sum_{i \in \mathcal{P}_{high}} 8d(s_i, t_i)\frac{T_i}{k} \leq d(C) + \frac{8}{k}\sum_{i=1}^{m} w_i \cdot d(s_i, t_i)$. Note that $d(C)$ is at most 1.5 times the minimum tour on all sources (Steiner lower bound), and the second term above is the flow lower bound. So tour τ_1 has length at most $O(1)$ times the preemptive lower bounds for $\mathcal{I}$, which is at most $O(1) \cdot d(\tau^*)$.

Servicing $\mathcal{P}'$: We know that the total size T_i of each pair i in $\mathcal{P}'$ lies in the interval $(k/l, k/2)$. Let $\mathcal{I}'$ denote the instance of weighted Dial-a-Ride with objects $\{(s_i, t_i, T_i) : i \in \mathcal{P}'\}$ and vehicle capacity k; note that the number of objects in $\mathcal{I}'$ is at most l. The tour τ^* restricted to the objects corresponding to pairs in $\mathcal{P}'$ is a feasible solution to the *unweighted instance* corresponding to $\mathcal{I}'$ (but it may not be feasible for $\mathcal{I}'$ itself). However Lemma 14 implies that the optimal value of $\mathcal{I}'$, $\mathsf{OPT}(\mathcal{I}') \leq O(1) \cdot d(\tau^*)$.

Next we reduce instance $\mathcal{I}'$ to an instance $\mathcal{J}$ of weighted Dial-a-Ride satisfying the following conditions: **(i)** $\mathcal{J}$ has at most l objects, **(ii)** each object in $\mathcal{J}$ has size at most $2l$, **(iii)** any feasible solution to $\mathcal{J}$ is feasible for $\mathcal{I}'$, and **(iv)** the optimal value $\mathsf{OPT}(\mathcal{J}) \leq O(1) \cdot \mathsf{OPT}(\mathcal{I}')$. If $k \leq 2l$, $\mathcal{J} = \mathcal{I}'$ itself satisfies the required conditions. Suppose $k \geq 2l$, then define $p = \lfloor \frac{k}{l} \rfloor$; note that $k \geq l \cdot p \geq k - l \geq \frac{k}{2}$. Round up each size T_i to the smallest integral multiple T_i' of p, and round down the capacity k to $k' = l \cdot p$. Since each size $T_i \in (\frac{k}{l}, \frac{k}{2})$, all sizes $T_i' \in \{p, 2p, \cdots, lp\}$. Now let $\mathcal{I}''$ denote the weighted Dial-a-Ride instance with objects $\{(s_i, t_i, T_i') : i \in \mathcal{P}'\}$ and vehicle capacity $k' = lp$.

One can obtain a feasible solution for $\mathcal{I}''$ from any feasible solution σ for $\mathcal{I}'$ by traversing σ a constant number of times as follows. Consider simulating a traversal of a capacity k vehicle α along σ by 16 capacity k' vehicles $\{\beta_g\}_{g=1}^{16}$, each running in parallel along σ. The objects $\{i \mid T_i' \leq \frac{k}{4}\}$ are served by vehicles $\{\beta_g\}_{g=1}^{8}$, and the rest by vehicles $\{\beta_g\}_{g=9}^{16}$. Whenever vehicle α picks-up an object i, one of the vehicles $\{\beta_g\}_{g=1}^{16}$ picks up i: if $T_i' \leq \frac{k}{4}$, any vehicle $\{\beta_g\}_{g=1}^{8}$ that has free capacity picks up i; if $T_i' > \frac{k}{4}$, any vehicle $\{T_g'\}_{g=9}^{16}$ that is empty picks up i. It can be seen that if some object is not picked by any vehicle $\{\beta_g\}_{g=1}^{16}$, then there must be a capacity violation in α (since $k' \geq \frac{k}{2}$ and $T_i' \leq \max\{2T_i, k'\}$).

So the optimal value of $\mathcal{I}''$ is at most $O(1) \cdot \mathsf{OPT}(\mathcal{I}')$. Now note that all sizes and the vehicle capacity in $\mathcal{I}''$ are multiples of p; scaling down each of these quantities by p, we get an instance $\mathcal{J}$ *equivalent* to $\mathcal{I}''$ where the vehicle capacity is l (and every

object size is at most l). This instance $\mathcal{J}$ satisfies all the four conditions claimed above.

Since $\mathcal{J}$ has at most l objects (each of size $\leq 2l$), the unweighted instance corresponding to $\mathcal{J}$ has at most $2l^2 \leq 2n^4$ objects. Thus, this unweighted instance can be solved using the ρ-approximation algorithm for such instances, assumed in the theorem. Then using the algorithm in Lemma 14, we obtain a solution to $\mathcal{J}$, of length at most $O(\rho) \cdot \mathsf{OPT}(\mathcal{J}) \leq O(\rho) \cdot \mathsf{OPT}(\mathcal{I}') \leq O(\rho) \cdot d(\tau^*)$. Since any feasible solution to $\mathcal{J}$ corresponds to one for $\mathcal{I}'$, we have a tour servicing $\mathcal{P}'$ of length at most $O(\rho) \cdot d(\tau^*)$.

Finally, combining the tours servicing $\mathcal{P}_{low}$, $\mathcal{P}_{high}$ and $\mathcal{P}'$, we obtain a feasible tour for $\mathcal{I}$ having length $O(\rho) \cdot d(\tau^*)$, which gives us the desired approximation algorithm. ■

Theorem 15 also justifies the assumption $\log m = O(\log n)$ made at the end of Section 2.3. This is important because in general m may be super-polynomial in n.

2.4 The Effect of Preemptions

In this section, we study the effect of the number of preemptions in the Dial-a-Ride problem. We mentioned two versions of the Dial-a-Ride problem (Definition 3): in the preemptive version, an object may be preempted any number of times, and in the non-preemptive version objects are not allowed to be preempted even once. Clearly the preemptive version is least restrictive and the non-preemptive version is most restrictive. One may consider other versions of the Dial-a-Ride problem, where there is a specified upper bound P on the number of times an object can be preempted. Note that the case $P = 0$ is the non-preemptive version, and the case $P = n$ is the preemptive version. In Theorem 16, we show that for any instance of the Dial-a-Ride problem, there is a tour that preempts each object at most once (i.e., $P = 1$) and has length at most $O(\log^2 n)$ times an optimal preemptive tour (i.e., $P = n$). This implies that the real gap between preemptive and non-preemptive tours is between zero and one preemption per object. A tour that preempts each object at most once is called a *1-preemptive tour*.

Theorem 16. *Given any instance of the Dial-a-Ride problem, there is a 1-preemptive tour of length at most $O(\log^2 n)$ times the Steiner and flow lower bounds. Such a tour can be found in randomized polynomial time.*

Proof: Let LB_{pmt} denote the preemptive lower bound for the given Dial-a-Ride

instance, namely maximum of the *Steiner* and *flow* lower bounds. We first show how general Dial-a-Ride instances can be reduced to instances where the metric is a *hierarchically well-separated* tree T having $O(\log n)$ levels. This uses the results on probabilistic tree embedding [46], and only increases the expected value of the preemptive lower bound by an $O(\log n)$ factor. Then we show how to obtain a 1-preemptive tour on such tree-instances having length $O(\log n)$ times the preemptive lower bound. The resulting 1-preemptive tour has the property that each object is moved non-preemptively in two phases: first from its source to the *least-common-ancestor* (lca) of its source and destination, and then from the lca to its destination.

Applying the probabilistic tree embedding of [46] to the given metric (V, d), we obtain a tree metric T such that the optimal preemptive tour in T has length $O(\log n)$ times that in (V, d), and any feasible solution in T corresponds to one in (V, d) of the same length. Additionally, the tree T has $O(\log \frac{d_{max}}{d_{min}})$ levels, where d_{max} and d_{min} denote the maximum and minimum (non-zero) distances in the original metric. Let OPT_{pmt} denote the optimal value of the original preemptive Dial-a-Ride instance. We first observe that using standard scaling arguments, it suffices to assume that for metric (V, d), $\frac{d_{max}}{d_{min}}$ is polynomial in n. Without loss of generality, any preemptive tour involves at most $2m \cdot n$ edge traversals: each object is picked or dropped at most $2n$ times (once at each vertex), and every visit to a vertex involves picking or dropping at least one object (otherwise the tour can be shortcut over this vertex-visit at no increase in length). By retaining only vertices within distance $\mathsf{OPT}_{pmt}/2$ from the root r, we preserve the optimal preemptive tour and ensure that $d_{max} \leq \mathsf{OPT}_{pmt}$. Now consider modifying the metric by setting all edges of length smaller than $\mathsf{OPT}_{pmt}/2mn^3$ to length 0; the new distances are shortest paths under the modified edge lengths. So any pairwise distance decreases by at most $\frac{\mathsf{OPT}_{pmt}}{2mn^2}$. Clearly the length of the optimal preemptive tour only decreases under this modification. Since there are at most $2mn$ edge traversals in any preemptive tour, the increase in tour length in going from the new metric to the original metric is at most $2mn \cdot \frac{\mathsf{OPT}_{pmt}}{2mn^2} \leq \mathsf{OPT}_{pmt}/n$. Thus at the loss of a constant factor, we may assume that $d_{max}/d_{min} \leq 2mn^3$. Furthermore, Theorem 14 also holds for preemptive Dial-a-Ride; so we may assume (at the loss of an additional constant factor) that the number of objects $m \leq O(n^4)$. So we have $d_{max}/d_{min} \leq O(n^7)$ and hence tree T has $O(\log n)$ levels.

The tree T resulting from [46] has several Steiner vertices that are not present in the original metric; so the tour that we find on T may actually preempt objects at Steiner vertices, in which case it is not feasible in the original metric. However as shown in [69], these Steiner vertices can be simulated by vertices in the original

metric (at the loss of a constant factor). Based on the preceding observations, we assume that the underlying metric is a tree T on the original vertex set having $l = O(\log n)$ levels, such that the expected value of the preemptive lower bound is $\widetilde{\mathsf{LB}}_{pmt} = O(\log n) \cdot \mathsf{LB}_{pmt}$.

We now partition the demand-pairs in T into l sets with D_i (for $i = 1, \cdots, l$) consisting of all pairs having their least common ancestor (lca) in level i. We service each D_i separately in a 1-preemptive manner using a tour of length $O(\widetilde{\mathsf{LB}}_{pmt})$. Then concatenating the tours for each level i, we obtain the theorem.

Servicing D_i: For each vertex v at level i in T, let L_v denote the pairs in D_i that have v as their lca, and let $\mathsf{LB}_{pmt}(v)$ denote the preemptive lower bound for the Dial-a-Ride instance with pairs L_v and root v. Note that for each v, the Dial-a-Ride instance on L_v is restricted to the subtree under vertex v; so $\sum_v \mathsf{LB}_{pmt}(v) \leq \widetilde{\mathsf{LB}}_{pmt}$.

Consider the following two instances of *capacitated vehicle routing* derived from L_v: $\mathcal{S}_v$ has vertex v as the common destination and $\{s_j \mid j \in L_v\}$ are the respective sources; and $\mathcal{T}_v$ has vertex v as the common source and $\{t_j \mid j \in L_v\}$ are the respective destinations. Since the $s_j - t_j$ path of each pair $j \in L_v$ crosses vertex v, we have $d(s_j, t_j) = d(s_j, v) + d(t_j, v)$. This implies that the preemptive lower bounds of both $\mathcal{S}_v$ and $\mathcal{T}_v$ are at most $\mathsf{LB}_{pmt}(v)$. Hence, using the capacitated vehicle routing algorithm of [73] gives the following two *non-preemptive* tours: σ'_v (for instance $\mathcal{S}_v$) that moves objects in L_v from their sources to v, and σ''_v (for instance $\mathcal{T}_v$) that moves objects in L_v from v to their destinations. Furthermore, the approximation guarantee from [73] (see Section 2.1) implies that $d(\sigma'), d(\sigma') \leq 2.5 \cdot \mathsf{LB}_{pmt}(v)$. Finally, the concatenation $\sigma'_v \cdot \sigma''_v$ is a 1-preemptive tour servicing L_v having length at most $5 \cdot \mathsf{LB}_{pmt}(v)$.

We now run a depth-first-search on tree T to visit all vertices v in level i, and use the algorithm described above for servicing objects L_v when v is visited in this traversal. This results in a tour servicing D_i, having length at most $2 \cdot d(T) + 5 \sum_v \mathsf{LB}_{pmt}(v)$. Since $2 \cdot d(T)$ is the Steiner lower bound, and $\sum_v \mathsf{LB}_{pmt}(v) \leq \widetilde{\mathsf{LB}}_{pmt}$. Thus the tour servicing D_i has length at most $6 \cdot \widetilde{\mathsf{LB}}_{pmt}$.

Finally concatenating the tours for each level $i = 1, \cdots, l$, we obtain a 1-preemptive tour on tree-instance T of length $O(\log n) \cdot \widetilde{\mathsf{LB}}_{pmt}$, which translates to a 1-preemptive tour on the original metric having length $O(\log^2 n) \cdot \mathsf{LB}_{pmt}$. ∎

Preemption Gap in the Euclidean Plane. Motivated by obtaining an improved approximation for Dial-a-Ride on the Euclidean plane, we next consider the worst case gap between an optimal non-preemptive tour and the preemptive lower bounds.

As mentioned earlier, [28] showed that there are instances of Dial-a-Ride where the ratio of the optimal non-preemptive tour to the optimal preemptive tour is $\Omega(n^{1/3})$. However, the metric involved in this example was the uniform metric on n points, which can not be embedded in the Euclidean plane. The following theorem shows that even in this special case, there is a polynomial gap between non-preemptive and preemptive tours, and so preemptive lower bounds do not suffice to obtain a sub-polynomial approximation ratio for non-preemptive Dial-a-Ride.

Theorem 17. *There are instances of Dial-a-Ride on the Euclidean plane where the optimal non-preemptive tour has length $\Omega(\frac{n^{1/8}}{\log^3 n})$ times the optimal preemptive tour.*

Proof: Consider a square of side 1 in the Euclidean plane, in which a set of n demand-pairs $\{s_i, t_i\}_{i=1}^n$ are distributed uniformly at random (each demand point is generated independently and is uniformly from the square). The vehicle capacity is set to $k = \sqrt{n}$. Let $\mathcal{R}$ denote a random instance of Dial-a-Ride obtained as above. We show that in this case, the optimal non-preemptive tour has length $\tilde{\Omega}(n^{1/8})$ with high probability. We first show the following claim.

Claim 18. *With high probability, the minimum length of a tree connecting k pairs in $\mathcal{R}$ is $\Omega(\frac{n^{1/8}}{\log n})$.*

Proof: Take any set S of $k = \sqrt{n}$ demand-pairs, say $\{s_i, t_i\}_{i=1}^k$. Note that the number of such sets S is $\binom{n}{k}$. Set S has $2k$ vertices, each generated uniformly at random. It is well-known that there are p^{p-2} different labeled trees on p vertices (see e.g. [149], Ch.2). The term *labeled* emphasizes that we are not identifying isomorphic graphs, i.e., two trees are counted as the same if and only if exactly the same pairs of vertices are adjacent. Thus there are at most $(2k)^{2k-2}$ such trees on set S. Consider any labeled tree T on vertices S, and root it at the source vertex with minimum label (here s_1). We assume that T has been generated using the "Principle of Deferred Decisions", i.e., vertices will be generated one by one according to some breadth-first ordering of T. We say that an edge is *short* if its length is at most $\frac{c}{\alpha k}$ (c and $\alpha \in (0, \frac{1}{2})$ will be fixed later).

If T has length at most c, it is clear that at most an α fraction of its edges are *not* short. So $Pr[\text{length of } T \leq c] \leq \sum_H Pr[\text{edges in } H \text{ are short}]$, where H in the summation ranges over all edge-subsets of T with $|H| \geq (1-\alpha)2k$. For a fixed H, we bound $Pr[\text{edges in } H \text{ are short}]$ as follows. For any edge $(v, \text{parent}(v))$ (note parent(v) is well-defined since T is rooted), assuming that parent(v) is fixed, the probability that this edge is short is $p = \pi(\frac{c}{\alpha k})^2$. So we can upper bound the

probability that edges H are short by $p^{|H|} \leq p^{(1-\alpha)2k}$. So we have $Pr[\text{length of } T \leq c] \leq 2^{2k} \cdot p^{(1-\alpha)2k}$, as the number of different edge sets H is at most 2^{2k}.

By a union bound over all such labeled trees T, the probability that the length of the minimum spanning tree on S is less than c is at most $(2k)^{2k} \cdot 2^{2k} \cdot p^{(1-\alpha)2k}$. Now taking a union bound over all k-sets S, the probability that the minimum length of a tree containing *some* k pairs is less than c is at most $\binom{n}{k}(2k)^{2k}2^{2k}p^{(1-\alpha)2k}$. Since $k = \sqrt{n}$, this term can be upper bounded as follows:

$$\begin{aligned}(ek)^k(4k)^{2k}\pi^{(1-\alpha)2k}(\frac{c}{\alpha k})^{(1-\alpha)4k} &\leq 500^k k^{3k}(\frac{c}{\alpha k})^{(1-\alpha)4k} \\ &= [500 \cdot (\frac{c}{\alpha})^{4-4\alpha}(\frac{1}{k})^{1-4\alpha}]^k \ \leq 2^{-k}\end{aligned}$$

The last inequality above holds when $c \leq \frac{\alpha}{1000} \cdot k^{1/4-3\alpha/(1-4\alpha)}$. Setting $\alpha = \frac{1}{\log k}$,

$$Pr[\exists \ \frac{k^{1/4}}{8000 \cdot \log k} \text{ length tree connecting some } k \text{ pairs in } \mathcal{R}] \leq 2^{-k}$$

So, with probability at least $1 - 2^{-\sqrt{n}}$, the minimum length of a tree containing k pairs in $\mathcal{R}$ is at least $\Omega(\frac{n^{1/8}}{\log n})$. ∎

From Theorem 7, we obtain that there is a near optimal non-preemptive tour servicing all the objects in segments, where each segment (except possibly the last) involves servicing a set of $\frac{k}{2} \leq t \leq k$ objects. Although the lower bound of $k/2$ is not stated in Theorem 7, it is easy to extend the statement to include it. This implies that any solution of this structure has at least $\frac{n}{k} = k$ segments. Since each segment covers at least $k/2$ pairs, Claim 18 implies that each of these segments has length $\Omega(n^{1/8}/\log n)$. So the best solution of the structure given in Theorem 7 has length $\Omega(\frac{n^{1/8}}{\log n}k)$. But since there is a near-optimal solution of this structure, the optimal non-preemptive tour on $\mathcal{R}$ has length $\Omega(\frac{n^{1/8}}{\log^2 n}k)$.

On the other hand, the flow lower bound for $\mathcal{R}$ is at most $\frac{n}{k} = k$, and the Steiner lower bound is at most $O(\sqrt{n}) = O(k)$ (an $O(\sqrt{n})$ length tree on the $2n$ points can be constructed using a $\sqrt{2n} \times \sqrt{2n}$ gridding). So the preemptive lower bounds are both $O(k)$; now using the algorithm of [28], we see that the optimal preemptive tour has length $O(k \log n)$. Combined with the lower bound for non-preemptive tours, we obtain the Theorem. ∎

Credits: The results in this chapter are from *"Dial a Ride from k-forest"* [70], obtained jointly with Anupam Gupta, MohammadTaghi Hajiaghayi and R. Ravi. We also thank Alan Frieze for help in proving Theorem 17.

Chapter 3

Multi vehicle Dial-a-Ride

3.1 Introduction

The *multi-vehicle Dial-a-Ride* problem involves routing a set of m objects from their sources to respective destinations using a set of q vehicles starting at t distinct depot nodes in an n-node metric. Each vehicle has a *capacity* k which is the maximum number of objects it can carry at any time. As in the single vehicle case, there are two versions of multi-vehicle Dial-a-Ride based on whether or not vehicles may use nodes in the metric as preemption (a.k.a. transshipment) points. In this chapter, our main focus is on the *preemptive* version, where an object may be left at intermediate locations while being moved from source to destination. There are two natural objectives in multi-vehicle Dial-a-Ride problems: total completion time or makespan (i.e. maximum completion time) over the q vehicles. As observed below, the total completion time objective is very similar to single vehicle Dial-a-Ride. We study the more interesting *makespan* objective in this chapter.

The total completion time objective in preemptive multi-vehicle Dial-a-Ride admits a straightforward $O(\log n)$-approximation along the lines of the single vehicle problem [28]: Using the FRT tree embedding [46], one can reduce the problem to tree-metrics at the loss of an expected $O(\log n)$ factor, and there is a simple constant approximation for this problem on trees. The maximum completion time or makespan objective turns out to be considerably harder. Due to non-linearity of the makespan objective, the above reduction to tree-metrics does not hold. Furthermore, the makespan objective does not appear easy to solve even on trees.

Unlike in the single-vehicle case, note that an object in preemptive multi-vehicle

Dial-a-Ride may be transported by several vehicles one after the other. Hence it is important for the vehicle routes to be coordinated so that the objects trace valid paths from respective sources to destinations. For example, a vehicle may have to wait at a vertex for other vehicles carrying common objects to arrive. The multi-vehicle Dial-a-Ride problem captures aspects of both *machine scheduling* and *network design* problems; this connection is more evident in the next subsection.

3.1.1 Problem Definition and Preliminaries

We represent a finite metric as (V, d) where V is the set of vertices and d is a symmetric distance function satisfying triangle inequality. For subsets $A, B \subseteq V$ we denote by $d(A, B)$ the minimum distance between a vertex in A and another in B, so $d(A, B) = \min\{d(u, v) \mid u \in A, v \in B\}$. For a subset $E \subseteq \binom{V}{2}$ of edges, $d(E) := \sum_{e \in E} d_e$ denotes the total length of edges in E.

The *multi-vehicle Dial-a-Ride problem* (mDaR) consists of an n-vertex metric (V, d) representing travel times between vertices, m objects specified as source-destination pairs $\{s_i, t_i\}_{i=1}^m$, q vehicles having respective depot-vertices $\{r_j\}_{j=1}^q$, and a common vehicle capacity k. A feasible schedule is a set of q routes, one for each vehicle (where the route for vehicle $j \in [q]$ starts and ends at r_j), such that no vehicle carries more than k objects at any time and each object is moved from its source to destination. The completion time C_j of any vehicle $j \in [q]$ is the time when vehicle j returns to its depot r_j at the end of its route (the schedule is assumed to start at time 0). The objective in mDaR is to minimize makespan, i.e. $\min \max_{j \in [q]} C_j$. We denote by $S := \{s_i \mid i \in [m]\}$ the set of sources, $T := \{t_i \mid i \in [m]\}$ the set of destinations, $R := \{r_j \mid j \in [q]\}$ the set of distinct depot-vertices, and $t := |R|$ the number of distinct depots. Unless mentioned otherwise, we only consider the *preemptive* version, where objects may be left at intermediate vertices while being moved from source to destination.

Single vehicle Dial-a-Ride. The following are lower bounds for the single vehicle problem: the minimum length TSP tour on the depot and all source/destination vertices (*Steiner* lower bound), and $\frac{\sum_{i=1}^m d(s_i,t_i)}{k}$ (*flow* lower bound). Charikar and Raghavachari [28] gave an $O(\log n)$ approximation algorithm for this problem based on the above lower bounds. A feasible solution to preemptive Dial-a-Ride is said to be *1-preemptive* if every object is preempted at most once while being moved from its source to destination. Theorem 16 in Chapter 2 showed that the single vehicle preemptive Dial-a-Ride problem always has a 1-preemptive tour of length $O(\log^2 n)$

times the Steiner and flow lower-bounds.

Lower bounds for mDaR**.** The quantity $\frac{\sum_{i=1}^{m} d(s_i,t_i)}{qk}$ is a lower bound similar to the flow bound for single vehicle Dial-a-Ride. Analogous to the Steiner lower bound above, is the optimal value of an induced *nurse-station-location* instance. In the nurse-station-location problem [44], we are given a metric (V, d), a set $\mathcal{T}$ of terminals and a multi-set $\{r_j\}_{j=1}^{q}$ of depot-vertices; the goal is to find a collection $\{F_j\}_{j=1}^{q}$ of trees that collectively contain all terminals $\mathcal{T}$ such that each tree F_j is rooted at vertex r_j and $\max_{j=1}^{q} d(F_j)$ is minimized. Even et al. [44] gave a 4-approximation algorithm for this problem. The optimal value of the nurse-station-location instance with depots $\{r_j\}_{j=1}^{q}$ (depots of vehicles in mDaR) and terminals $\mathcal{T} = S \cup T$ is a lower bound for mDaR. The following are some lower bounds implied by nurse-station-location: (a) $1/q$ times the minimum length forest that connects every vertex in $S \cup T$ to some depot vertex, (b) $\max_{i\in[m]} d(R, s_i)$, and (c) $\max_{i\in[m]} d(R, t_i)$. Finally, it is easy to see that $\max_{i\in[m]} d(s_i, t_i)$ is also a lower bound for mDaR.

3.1.2 Results

We first consider the special case of multi-vehicle Dial-a-Ride (*uncapacitated* mDaR) where the vehicles have no capacity constraints (i.e. $k \geq m$). This problem is interesting in itself, and serves as a good starting point before we present the algorithm for the general case. We prove the following theorem in Section 3.2.

Theorem 19. *There is an $O(\log t)$-approximation algorithm for uncapacitated pre-emptive* mDaR*.*

The above algorithm has two main steps: the first one reduces the instance (at a constant factor loss in the performance guarantee) to one in which all demands are between depots (a "depot-demand" instance). In the second step, we use a *sparse spanner* on the demand graph to construct routes for moving objects across depots. We also construct an instance of uncapacitated mDaR where the optimal value is $\Omega(\log t/\log\log t)$ times all our lower bounds for this problem. This suggests that stronger lower bounds are needed to obtain a better approximation ratio than what our approach provides.

We obtain an improved guarantee for the following special class of metrics (eg. planar metrics). This algorithm uses the notion of *sparse covers* in such metrics [97].

Theorem 20. *There is an $O(1)$-approximation algorithm for uncapacitated* mDaR *on metrics induced by graphs that exclude any fixed minor.*

In Section 3.3, we study the capacitated preemptive mDaR problem, and obtain the main result of this chapter. Recall that there is an $\Omega(\log^{1/4-\epsilon} n)$ hardness of approximation for even single vehicle preemptive Dial-a-Ride [66].

Theorem 21. *There is an $O(\log^3 n)$ approximation algorithm for preemptive* mDaR.

This algorithm has four key steps: (1) We *preprocess* the input so that demand points that are sufficiently far away from each other can be essentially decomposed into separate instances for the algorithm to handle independently. (2) We then solve a single-vehicle instance of the problem that obeys some additional bounded-delay property (Theorem 27) that we prove; This property combines ideas from algorithms for *light approximate shortest path trees* [95] and *capacitated vehicle routing* [73]. The bounded-delay property is useful in *randomly partitioning* the single vehicle solution among the q vehicles available to share this load. This random partitioning scheme is reminiscent of the work of Hochbaum-Maase [85], Baker [12] and Klein-Plotkin-Rao [97], in trying to average out the effect of the cutting in the objective function. (3) The partitioned segments of the single vehicle tour are assigned to the available vehicles; However, to check if this assignment is feasible we solve a matching problem that identifies cases when this load assignment must be *rebalanced*. This step identifies stronger lower bounds for subproblems where the current load assignment is not balanced. (4) We finish up by *recursing* on the load rebalanced subproblem.

3.1.3 Related Work

Dial-a-Ride problems with transshipment (the preemptive version) have been studied in [109, 110, 115]. These papers consider a more general model where preemption is allowed only at a specified subset of vertices. Our model (and that of [28]) is the special case when every vertex can serve as a preemption point. It is clear that preemption only reduces the cost of serving demands: [115] studied the maximum decrease in the optimal cost upon introducing one preemption point. [109, 110] also model time-windows on the demands, and study heuristics and a column-generation based approach; they also describe applications (eg. courier service) that allow for preemptions. Another preemptive routing problem that has been studied is *truck and trailer routing* [25, 135]. Here a number of capacitated

trucks and trailers are used to deliver all objects. Some customers are only accessible without the trailer. The trailers can be parked at any point accessible with a trailer and it is possible to shift demand loads between the truck and the trailer at the parking places.

Paepe et al. [41] provide a classification of Dial-a-Ride problems using a notation similar to that for scheduling and queuing problems: preemption is one aspect in this classification.

As mentioned in the previous chapter, the best known approximation guarantee for the preemptive single vehicle Dial-a-Ride is $O(\log n)$ [28], and an $\Omega(\log^{1/4-\epsilon} n)$ hardness of approximation (for any constant $\epsilon > 0$) is shown in [66]. The non-preemptive version appears much harder and the best known approximation ratio is $\min\{\sqrt{k}\log n, \sqrt{n}\log^2 n\}$ (Charikar and Raghavachari [28], Chapter 2); however APX-hardness is the best known lower bound.

The uncapacitated case of preemptive mDaR generalizes the *nurse-station-location* [44], for which a 4-approximation algorithm was given. Nurse-station-location is the special case of uncapacitated mDaR when each source-destination pair coincides at a single vertex.

3.2 Uncapacitated Preemptive mDaR

In this section we study the uncapacitated special case of preemptive mDaR, where vehicles have no capacity constraints (i.e. capacity $k \geq m$). We give an algorithm that achieves an $O(\log t)$ approximation ratio for this problem (recall $t \leq n$ is the number of distinct depots). Unlike in the single vehicle case, preemptive and non-preemptive versions of mDaR are very different even without capacity constraints.

Preemption gap in Uncapacitated mDaR. Consider an instance of uncapacitated mDaR where the metric is induced by an unweighted star with q leaves (where q is number of vehicles), all q vehicles have the center vertex as depot, and there is a demand between every pair of leaf-vertices. A preemptive schedule having makespan 4 is as follows: each vehicle $j \in [q]$ visits leaf j and brings all demands with source j to the root, then each vehicle j visits its corresponding leaf again, this time delivering all demands with destination j. On the other hand, in any non-preemptive schedule, one of the q vehicles completely serves at least q demands (since there are q^2 demands in all). The minimum length of any tour containing

the end points of q demands is $\Omega(\sqrt{q})$, which is also a lower bound on the optimal non-preemptive makespan. Thus there is an $\Omega(\sqrt{q})$ factor gap between optimal preemptive and non-preemptive tours.

The algorithm for uncapacitated preemptive mDaR proceeds in two stages. Given any instance, it is first reduced (at the loss of a constant factor) to a depot-demand instance, where all demands are between depot vertices (Subsection 3.2.1). Then the depot-demand instance is solved using an $O(\log t)$ approximation algorithm (Subsection 3.2.2).

3.2.1 Reduction to depot-demand instances

We define *depot-demand instances* as those instances of uncapacitated mDaR where all demands are between depot vertices. Given any instance $\mathcal{I}$ of uncapacitated mDaR, the algorithm UncapMulti (Figure 3.1) reduces $\mathcal{I}$ to a depot-demand instance. We now argue that the reduction in UncapMulti only loses a constant approximation factor. Let B denote the optimal makespan of instance $\mathcal{I}$. Since the optimal value of the nurse-station-location instance solved in the first step of UncapMulti is a lower bound for $\mathcal{I}$, we have $\max_{j=1}^{q} d(F_j) \leq 4B$.

Claim 22. *The optimal makespan for the depot-demand instance $\mathcal{J}$ is at most $17B$.*

Proof: Consider a feasible schedule for $\mathcal{J}$ involving three rounds: (1) each vehicle traverses (by means of an Euler tour) its corresponding tree in $\{F_j\}_{j=1}^{q}$ and moves each object i from its source-depot (the source in instance $\mathcal{J}$) to s_i (source in original instance $\mathcal{I}$); (2) each vehicle follows the optimal schedule for $\mathcal{I}$ and moves each object i from s_i to t_i (destination in $\mathcal{I}$); (3) each vehicle traverses its corresponding tree in $\{F_j\}_{j=1}^{q}$ and moves each object i from t_i to its destination-depot (the destination in $\mathcal{J}$). Clearly this is a feasible schedule for $\mathcal{J}$. From the observation on the nurse-station-location instance, the time taken in each of the first and third rounds is at most $8B$. Furthermore, the time taken in the second round is the optimal makespan of $\mathcal{I}$ which is B. This proves the claim. ∎

Assuming a feasible schedule for $\mathcal{J}$, it is clear that the schedule returned by UncapMulti is feasible for the original instance $\mathcal{I}$. The first and third rounds in $\mathcal{I}$'s schedule require at most $8B$ time each. Thus an approximation ratio α for depot-demand instances implies an approximation ratio of $17\alpha + 8$ for general instances. In the next subsection, we show an $O(\log t)$-approximation algorithm for depot-demand instances (here t is the number of depots), which implies Theorem 19.

Input: instance $\mathcal{I}$ of uncapacitated preemptive mDaR**.**

- Solve the nurse-station-location instance with depots $\{r_j\}_{j=1}^{q}$ and all sources/destinations $S \cup T$ as terminals, using the 4-approximation algorithm [44]. Let $\{F_j\}_{j=1}^{q}$ be the resulting trees covering $S \cup T$ such that each tree F_j is rooted at depot r_j.
- Define a depot-demand instance $\mathcal{J}$ of uncapacitated mDaR on the same metric and set of vehicles, where the demands are $\{(r_j, r_l) \mid s_i \in F_j \ \& \ t_i \in F_l,\ 1 \leq i \leq m\}$. For any object $i \in [m]$ let the *source depot* be the depot r_j for which $s_i \in F_j$ and the *destination depot* be the depot r_l for which $t_i \in F_l$.
- Output the following schedule for $\mathcal{I}$:
  1. Each vehicle $j \in [q]$ traverses tree F_j by an Euler tour, picks up all objects from sources in F_j and brings them to their source-depot r_j.
  2. Vehicles implement a schedule for *depot-demand instance* $\mathcal{J}$, and all objects are moved from their source-depot to destination-depot (using the algorithm in Section 3.2.2).
  3. Each vehicle $j \in [q]$ traverses tree F_j by an Euler tour, picks up all objects having destination-depot r_j and brings them to their destinations in F_j.

Figure 3.1: Algorithm UncapMulti for uncapacitated mDaR.

3.2.2 Algorithm for depot-demand instances

Let $\mathcal{J}$ be any depot-demand instance: note that the instance defined in the second step of UncapMulti is of this form. It suffices to restrict the algorithm to the induced metric (R, d) on only depot vertices, and use only one vehicle at each depot in R. Consider an undirected graph H consisting of vertex set R and edges corresponding to demands: there is an edge between vertices r and s iff there is an object going from either r to s *or* s to r. Note that the metric length of any edge in H is at most the optimal makespan $\tilde{B}$ of instance $\mathcal{J}$. In the schedule produced by our algorithm, vehicles will only use edges of H. Thus in order to obtain an $O(\log t)$ approximation, it suffices to show that each vehicle only traverses $O(\log t)$ edges. Based on this, we further reduce $\mathcal{J}$ to the following instance $\mathcal{H}$ of uncapacitated mDaR: the underlying metric is shortest paths in graph H (on vertices R), with one vehicle at each R-vertex, and for every edge $(u, v) \in H$ there is a demand from u to

v and one from v to u. Clearly any schedule for $\mathcal{H}$ having makespan β implies one for $\mathcal{J}$ of makespan $\beta \cdot \tilde{B}$. The next lemma implies an $O(\log |R|)$ approximation for depot-demand instances.

Lemma 23. *There exists a poly-time computable schedule for $\mathcal{H}$ with makespan $O(\log t)$, where $t = |R|$.*

Proof: Let $\alpha = \lceil \lg t \rceil + 1$. We first construct a *sparse spanner* A of H as follows: consider edges of H in an arbitrary order, and add an edge $(u, v) \in H$ to A iff the shortest path between u and v using current edges of A is more than 2α. It is clear from this construction that the girth of A (length of its shortest cycle) is at least 2α, and that for every edge $(u, v) \in H$, the shortest path between u and v in A is at most 2α.

We now assign each edge of A to one of its end-points such that each vertex is assigned at most two edges. Repeatedly pick any vertex v of degree at most two in A, assign its adjacent edges to v, and remove these edges and v from A. We claim that at the end of this procedure (when no vertex has degree at most 2), all edges of A would have been removed (i.e. assigned to some vertex). Suppose for a contradiction that this is not the case. Let $\tilde{A} \neq \phi$ be the remaining graph; note that $\tilde{A} \subseteq A$, so girth of $\tilde{A}$ is at least 2α. Every vertex in $\tilde{A}$ has degree at least 3, and there is at least one such vertex w. Consider performing a breadth-first search in $\tilde{A}$ from w. Since the girth of $\tilde{A}$ is at least 2α, the first α levels of the breadth-first search is a tree. Furthermore every vertex has degree at least 3, so each vertex in the first $\alpha - 1$ levels has at least 2 children. This implies that $\tilde{A}$ has at least $1 + 2^{\alpha-1} > t$ vertices, which is a contradiction! For each vertex $v \in R$, let A_v denote the edges of A assigned to v by the above procedure; we argued that $\cup_{v \in R} A_v = A$, and $|A_v| \leq 2$ for all $v \in R$.

The schedule for $\mathcal{H}$ involves 2α rounds as follows. In each round, every vehicle $v \in R$ traverses the edges in A_v (in both directions) and returns to v. Since $|A_v| \leq 2$ for all vertices v, each round takes 4 units of time; so the makespan of this schedule is $8\alpha = O(\log t)$. The route followed by each object in this schedule is the shortest path from its source to destination in spanner A; note that the length of any such path is at most 2α. To see that this is indeed feasible, observe that every edge of A is traversed by some vehicle in each round. Hence in each round, every object traverses one edge along its shortest path (unless it is already at its destination). Thus after 2α rounds, all objects are at their destinations. ∎

3.2.3 Tight example for uncapacitated mDaR lower bounds.

We note that known lower bounds for uncapacitated preemptive mDaR are insufficient to obtain a sub-logarithmic approximation guarantee. The lower bounds we used in our algorithm are the following: $\max_{i\in[m]} d(s_i, t_i)$, and the optimal value of a nurse-station-location instance with depots $\{r_j\}_{j=1}^{q}$ and terminals $S \cup T$. We are not aware of any lower bounds stronger than these two bounds.

We show an instance $\mathcal{G}$ of uncapacitated mDaR where the optimal makespan is a factor $\Omega(\frac{\log t}{\log\log t})$ larger than both the above lower bounds. In fact, the instance we construct is a depot-demand instance that has the same special structure as instance $\mathcal{H}$ in Lemma 23. I.e. the demand graph is same as the graph inducing distances. Take $G = (V, E)$ to be a t-vertex regular graph of degree $\sim \log t$ and girth $g \sim \log t/\log\log t$ (there exist such graphs, eg. Lazebnik et al. [103]). Instance $\mathcal{G}$ is defined on a metric on vertices V with distances being shortest paths in graph G. For every edge $(u, v) \in E$ of graph G, there is an object with source u and destination v (the direction is arbitrary). There is one vehicle located at every vertex of V; so number of vehicles $q = t$.

Observe that both our lower bounds are $O(1)$: the optimal value of the nurse-station-location instance is 0, and maximum source-destination distance is 1. However as we show below, the optimal makespan for this instance is at least $g - 1 = \Omega(\log t/\log\log t)$. Suppose (for contradiction) that there is a feasible schedule for $\mathcal{G}$ with makespan $M < g - 1$. A demand $(u, v) \in E$ is said to be *completely served* by a vehicle j iff the route of vehicle j visits both vertices u and v. The number of distinct vertices visited by any single vehicle is at most $M < g$: so the number of demands that are completely served by a single vehicle is at most $M - 1$ (otherwise these demand edges would induce a cycle smaller than the girth g). Hence the number of demands that are completely served by some vehicle is at most $t \cdot g < |E|$. Let $(u, v) \in E$ be a demand that is *not* completely served by any vehicle, i.e. there is no vehicle that visits both u and v. Since we have a feasible schedule of makespan M, the path π followed by demand (u, v) from u to v (or vice versa) in the schedule has length at most M. The path π can not be the direct edge (u, v) since demand (u, v) is not completely served by any vehicle. So path π together with edge (u, v) is a cycle of length at most $M + 1 < g$ in graph G, contradicting girth of G.

3.2.4 Improved guarantee for metrics excluding a fixed minor.

We now prove Theorem 20 that gives a constant approximation algorithm for uncapacitated mDaR on metrics induced by K_r-minor free graphs (for any fixed r). This improvement comes from using the existence of good 'sparse covers' in such metrics, as opposed to the spanner based construction in Lemma 23. This guarantee is again relative to the above mentioned lower bounds.

Consider an instance of uncapacitated preemptive mDaR problem on metric (V, d) that is induced by an edge-weighted graph $G = (V, E)$ containing no K_r-minor (for some fixed $r \geq 1$). We start with some definitions [21]. A *cluster* is any subset of vertices. For any $\gamma > 0$ and vertex $v \in V$, $N(v, \gamma) := \{u \in V \mid d(u, v) \leq \gamma\}$ denotes the set of vertices within distance γ from v. As observed in Busch et al. [21], the partitioning scheme of Klein et al. [97] implies the following result.

Theorem 24 ([97])**.** *Given K_r-minor free graph $G = (V, E, w)$ and value $\gamma > 0$, there is an algorithm that computes a set $Z = \{C_1, \cdots, C_l\}$ of clusters satisfying:*

1. *The diameter of each cluster is at most $O(r^2) \cdot \gamma$, i.e. $\max_{u,v \in C_i} d(u, v) \leq O(r^2) \cdot \gamma$ for all $i \in [l]$.*
2. *For every $v \in V$, there is some cluster $C_i \in Z$ such that $N(v, \gamma) \subseteq C_i$.*
3. *For every $v \in V$, the number of clusters in Z that contain v is at most $O(2^r)$.*

The set of clusters Z found above is called a *sparse cover* of G.

Theorem 25. *There is an $O(1)$-approximation algorithm for the uncapacitated preemptive* mDaR *problem on metrics induced by graphs that exclude any fixed minor.*

Proof: The reduction in Section 3.2 implies that it suffices to consider depot-demand instances: An $O(1)$ approximation for such instances implies an $O(1)$ approximation for general instances. Let $\mathcal{J}$ be any depot-demand instance on metric (V, d) induced by K_r-minor free graph $G = (V, E)$, with a set $R \subseteq V$ of depot-vertices (each containing a vehicle), and where all demands $\{s_i, t_i\}_{i=1}^m$ are between vertices of R. The algorithm is described in Figure 3.2.

Note that γ is a lower bound on the optimal makespan of $\mathcal{J}$. We first claim that the makespan of the above schedule is at most $O(1) \cdot \gamma$. Observe that each depot is contained in at most $O(2^r)$ clusters, and the distance from any depot to the center of any cluster containing it is at most $O(r^2) \cdot \gamma$. Hence the time taken by each vehicle

Input: Depot-demand instance $\mathcal{J}$ on metric $G = (V, E, w)$, depot-vertices $R \subseteq V$, demands $\{s_i, t_i\}_{i=1}^m$.

1. Let $\gamma = \max_{i \in [m]} d(s_i, t_i)$ be the maximum source-destination distance.
2. Compute a sparse cover $Z = \{C_j\}_{j=1}^l$ given in Theorem 24 for parameter γ.
3. For each cluster $C_j \in Z$, choose an arbitrary vertex $c_j \in C_j$ as its *center*.
4. For each demand $i \in [m]$, let $\pi(i) \in [l]$ be such that $s_i, t_i \in C_{\pi(i)}$.
5. Output the following schedule for $\mathcal{J}$:
   (a) Each vehicle $r \in R$ visits the centers of all clusters containing r, and returns to r. Vehicle r carries all the objects $\{i \in [m] \mid s_i = r\}$ having source r, and drops each object i at center $c_{\pi(i)}$.
   (b) Each vehicle $r \in R$ again visits the centers of all clusters containing r. Vehicle r brings all the objects $\{i \in [m] \mid t_i = r\}$ having destination r: each object i is picked up from center $c_{\pi(i)}$.

Output: An $O(1)$-approximate minimum makespan schedule for $\mathcal{J}$.

Figure 3.2: Algorithm for uncapacitated mDaR on K_r-minor free graphs.

in either of the two rounds above is at most $O(r^2 2^r) \cdot \gamma$. Since r is a fixed constant, the final makespan is $O(1) \cdot \gamma$.

We now argue the feasibility of the above schedule. Step (4) is well-defined: for all $i \in [m]$, we have $s_i, t_i \in N(s_i, \gamma)$ and there is some $j \in [l]$ with $N(s_i, \gamma) \subseteq C_j$ (i.e. we can set $\pi(i) = j$). It is now easy to see that each object $i \in [m]$ traces the following route in the final schedule: $s_i \rightsquigarrow c_{\pi(i)} \rightsquigarrow t_i$. ∎

3.3 Preemptive multi-vehicle Dial-a-Ride

In this section we prove our main result: an $O(\log^2 m \cdot \log n)$ approximation algorithm for the preemptive mDaR problem. In the next section we show how to remove the dependence on m to obtain an $O(\log^3 n)$ approximation algorithm even for *weighted* preemptive mDaR. We first obtain a new structure theorem on single-vehicle Dial-a-Ride tours (Subsection 3.3.1) that preempts each object at most once, and where

the total time spent by objects in the vehicle is bounded. Obtaining such a single vehicle tour is crucial in our algorithm for preemptive mDaR, which appears in Section 3.3.2. The algorithm for mDaR relies on a partial coverage algorithm Partial that given subsets Q of vehicles and D of demands, outputs a schedule for Q of near-optimal makespan that covers some *fraction* of demands in D. The main steps in Partial are as follows. **(1)** Obtain a *single-vehicle* tour satisfying 1-preemptive and bounded-delay properties (Theorem 27), **(2)** Randomly partition the single vehicle tour into $|Q|$ equally spaced pieces, **(3)** Solve a matching problem to assign *some* of these pieces to vehicles of Q that satisfy a subset of demands D, **(4)** A suitable fraction of the unsatisfied demands in D are covered recursively by unused vehicles of Q.

3.3.1 Capacitated Vehicle Routing with Bounded Delay

Before we present the structural result on Dial-a-Ride tours, we consider the classic *capacitated vehicle routing problem* [73] with an additional constraint on object 'delays'. Recall from Chapter 2 that the capacitated vehicle routing problem (CVRP) is a special case of single vehicle Dial-a-Ride when all objects have the same source (or equivalently, same destination). Formally, we are given a metric (V, d), specified depot-vertex $r \in V$, and m objects all having source r and respective destinations $\{t_i\}_{i\in[m]}$. The goal is to compute a minimum length *non-preemptive* tour of a capacity k vehicle originating at r that moves all objects from r to their destinations. In *CVRP with bounded delay*, we are additionally given a *delay parameter* $\alpha > 1$, and the goal is to find a minimum length capacitated non-preemptive tour serving all objects such that the time spent by each object $i \in [m]$ in the vehicle is at most $\alpha \cdot d(r, t_i)$. The preemptive lower bounds (c.f. Section 2.1) in the context of CVRP are as follows [73]: minimum length TSP tour on $\{r\} \cup \{t_i \mid i \in [m]\}$ (Steiner lower bound), and $\frac{2}{k}\sum_{i=1}^{m} d(r, t_i)$ (flow lower bound).

Theorem 26. *There is a* $(2.5 + \frac{3}{\alpha-1})$ *approximation algorithm for CVRP with bounded delay, where* $\alpha > 1$ *is the delay parameter. This guarantee is relative to the Steiner and flow lower bounds.*

Proof: Our algorithm is basically a combination of the algorithms for *light approximate shortest path trees* [95], and capacitated vehicle routing [73]. Let LB denote the maximum of the Steiner and flow lower bounds. The minimum TSP tour on the destinations plus r is the Steiner lower bound. The first step is to compute an approximately minimum TSP tour C: Christofides' algorithm [34] gives a 1.5

approximation, so $d(C) \leq 1.5 \cdot \mathsf{LB}$. Number the vertices in the order in which they appear in C, starting with r being 0. Using the procedure in Khuller et al. [95], we obtain a set of edges $\{(0, v_1), \cdots, (0, v_t)\}$ with $0 < v_1 < v_2 < \cdots < v_t < |V|$ having the following properties (below $v_0 = 0$).

1. For $1 \leq p \leq t$, for any vertex u with $v_{p-1} \leq u < v_p$, the length of edge $(0, v_{p-1})$ plus the path along C from v_{p-1} to u is at most $\alpha \cdot d(0, u)$.

2. $\sum_{p=1}^{t} d(0, v_p) \leq \frac{1}{\alpha - 1} \cdot d(C)$.

For each $1 \leq p \leq t$, define tour C_p which starts at r, goes to v_{p-1}, traverses C until v_p, then returns to r. Assign vertices $\{v_{p-1}, \cdots, v_p - 1\}$ (and all demands contained in them) to C_p. Also define tour C_{t+1} which starts at r, goes to v_t, and traverses C until r; and assign all remaining demands to C_{t+1}. It is clear that C_p (for $1 \leq p \leq t + 1$) visits each vertex assigned to it within α times the shortest path from r (using property 1 above). Also, the total length $\sum_{p=1}^{t+1} d(C_p) = d(C) + 2\sum_{p=1}^{t} d(0, v_p) \leq (1 + \frac{2}{\alpha - 1}) d(C)$.

For each C_p, we service the set D_p of demands assigned to it separately. Index the demands in D_p in the order in which they appear on C_p (breaking ties arbitrarily). Consider a capacitated tour which serves these demands in groups of at most k each, and returns to r after serving each group. The groups are defined as follows: starting at index 1, each group contains the next k contiguous demands (until all of D_p is assigned to groups). By rotating the indexing of demands, there are k different groupings of D_p that can be obtained: each of which corresponds to a capacitated tour. As argued in [73] (and is easy to see), the average length of these k tours is at most $d(C_p) + 2\sum_{z \in D_p} \frac{d(r,z)}{k}$. So the minimum length tour γ_p among these satisfies $d(\gamma_p) \leq d(C_p) + 2\sum_{z \in D_p} \frac{d(r,z)}{k}$.

The final solution γ is the concatenation of tours $\gamma_1, \cdots, \gamma_{t+1}$. Note that the time spent in the vehicle by any demand i is at most $\alpha \cdot d(r, t_i)$. The length of tour γ is at most $\sum_{p=1}^{t+1} d(C_p) + 2\sum_{z \in D} \frac{d(r,z)}{k}$, where D is the set of all demands. Note that $2\sum_{z \in D} \frac{d(r,z)}{k}$ is the flow lower bound for this single-source instance, so it is at most LB. Hence we obtain the following.

$$d(\gamma) \leq (1 + \frac{2}{\alpha - 1}) d(C) + \mathsf{LB} \leq (1 + \frac{2}{\alpha - 1}) \frac{3}{2} \mathsf{LB} + \mathsf{LB} = (2.5 + \frac{3}{\alpha - 1}) \mathsf{LB}$$

Clearly, solution γ satisfies the desired conditions in the theorem. ∎

We now consider the *single vehicle* preemptive Dial-a-Ride problem given by metric (V, d), set D of demands, and a vehicle of capacity k. The following structural result (Theorem 27) is a simple extension of Theorem 16 (from Chapter 2). This is obtained immediately by using Theorem 26 in the proof of Theorem 16, in place of the [73] algorithm.

Theorem 27. *There is a randomized poly-time computable 1-preemptive tour τ servicing D that satisfies the following conditions (where LB_{pmt} is the maximum of the Steiner and flow lower bounds):*

1. **Total length:** $d(\tau) \leq O(\log^2 n) \cdot \mathsf{LB}_{pmt}$.
2. **Bounded delay:** $\sum_{i \in D} T_i \leq O(\log n) \sum_{i \in D} d(s_i, t_i)$ *where T_i is the total time spent by object $i \in D$ in the vehicle under the schedule given by τ.*

3.3.2 Algorithm for preemptive mDaR

We are now ready to present our algorithm for preemptive multi-vehicle Dial-a-Ride. The algorithm first guesses the optimal makespan B of the given instance of preemptive mDaR (it suffices to know B within a constant factor, which is required for a polynomial-time algorithm). Let $\alpha = 1 - \frac{1}{1+\lg m}$. For any subset $P \subseteq [q]$, we abuse notation and use P to denote both the set of vehicles P and the multi-set of depots corresponding to vehicles P.

We give an algorithm Partial that takes as input a tuple $\langle Q, D, B \rangle$ where $Q \subseteq [q]$ is a subset of vehicles, $D \subseteq [m]$ a subset of demands and $B \in \mathbb{R}_+$, with the *promise* that vehicles Q (originating at their respective depots) suffice to completely serve the demands D at a makespan of B. Given such a promise, Partial $\langle Q, D, B \rangle$ returns a schedule of makespan $O(\log n \log m) \cdot B$ that serves a good fraction of D. Algorithm Partial$\langle Q, D, B \rangle$ is given below. We set parameter $\rho = \Theta(\log n \log m)$, the precise constant in the Θ-notation comes from the analysis.

Algorithm Partial$\langle Q, D, B \rangle$ for capacitated preemptive mDaR.
Input: Vehicles $Q \subseteq [q]$, demands $D \subseteq [m]$, bound $B \geq 0$ *such that* Q can serve all demands in D at makespan B.

Preprocessing

1. If the minimum spanning tree (MST) on vertices Q contains an edge of length greater than $3B$, there is a non-trivial partition $\{Q_1, Q_2\}$ of Q with $d(Q_1, Q_2) > 3B$. For

$j \in \{1,2\}$, let $V_j = \{v \in V \mid d(Q_j, v) \leq B\}$ and D_j be all demands of D induced on V_j. Run in parallel the schedules from Partial$\langle Q_1, D_1, B\rangle$ and Partial$\langle Q_2, D_2, B\rangle$. Assume this is not the case in the following.

Random partitioning

2. Obtain single-vehicle 1-preemptive tour τ using capacity k and serving demands D, by applying Theorem 27.

3. Choose a uniformly random offset $\eta \in [0, \rho B]$ and cut edges of tour τ at distances $\{p\rho B + \eta \mid p = 1, 2, \cdots\}$ along the tour to obtain a set $\mathcal{P}$ of pieces of τ.

4. C'' is the set of objects $i \in D$ such that i is carried by the vehicle in τ over some edge that is cut in Step (3); and $C' := D \setminus C''$. Ignore C'' objects in the rest of the algorithm.

Load rebalancing

5. Construct bipartite graph H with vertex sets $\mathcal{P}$ and Q and an edge between piece $P \in \mathcal{P}$ and depot $f \in Q$ iff $d(f, P) \leq 2B$. For any subset $A \subseteq \mathcal{P}$, $\Gamma(A) \subseteq Q$ denotes the neighborhood of A in graph H. Let $\mathcal{S} \subseteq \mathcal{P}$ be any *maximal* set that satisfies $|\Gamma(\mathcal{S})| \leq \frac{|\mathcal{S}|}{2}$.

6. Compute a *2-matching* $\pi : \mathcal{P} \setminus \mathcal{S} \to Q \setminus \Gamma(\mathcal{S})$, i.e. function s.t. $(P, \pi(P))$ is an edge in H for all $P \in \mathcal{P} \setminus \mathcal{S}$, and the number of pieces mapping to any $f \in Q \setminus \Gamma(\mathcal{S})$ is $|\pi^{-1}(f)| \leq 2$.

Recursion

7. Define $C_1 := \{i \in C' \mid$ either $s_i \in \mathcal{S}$ or $t_i \in \mathcal{S}\}$; and $C_2 := C' \setminus C_1$.

8. Run in parallel the *recursive* schedule Partial$\langle \Gamma(\mathcal{S}), C_1, B\rangle$ for C_1 and the following for C_2:

   (a) Each vehicle $f \in Q \setminus \Gamma(\mathcal{S})$ traverses the pieces $\pi^{-1}(f)$, moving all C_2-objects in them from their source to preemption-vertex, and returns to its depot.

   (b) Each vehicle $f \in Q \setminus \Gamma(\mathcal{S})$ again traverses the pieces $\pi^{-1}(f)$, this time moving all C_2-objects in them from their preemption-vertex to destination, and returns to its depot.

Output: A schedule of Q of makespan $(16 + 16\rho) \cdot B$ that serves an $\alpha^{\lg \min\{|Q|, 2m\}}$ fraction of D.

Lemma 28. *If there exists a schedule of vehicles Q covering all demands D, having makespan at most B, then* Partial *invoked on $\langle Q, D, B\rangle$ returns a schedule of vehicles Q of makespan at most $(16+16\rho)\cdot B$ that covers at least an $\alpha^{\lg z}$ fraction of D, where $z := \min\{|Q|, 2m\} \le 2m$.*

The final algorithm invokes Partial iteratively until all demands are covered: each time with the entire set $[q]$ of vehicles, all uncovered demands, and bound B. If $D \subseteq [m]$ is the set of uncovered demands at any iteration, Lemma 28 implies that Partial$\langle[q], D, B\rangle$ returns a schedule of makespan $O(\log m \log n)\cdot B$ that serves at least $\frac{1}{4}|D|$ demands. Hence a standard set-cover analysis implies that all demands will be covered in $O(\log m)$ rounds, resulting in a makespan of $O(\log^2 m \log n)\cdot B$.

It remains to prove Lemma 28. We proceed by induction on the number $|Q|$ of vehicles. The base case $|Q| = 1$ is easy: the tour τ in Step (2) has length $O(\log^2 n)\cdot B \le \rho B$, and satisfies all demands (i.e. fraction 1). In the following, we prove the inductive step, when $|Q| \ge 2$.

Preprocessing. Suppose Step (1) applies. Note that $d(V_1, V_2) > B$ and hence there is no demand with source in one of $\{V_1, V_2\}$ and destination in the other. So demands D_1 and D_2 partition D. Furthermore in the optimal schedule, vehicles Q_j (any $j = 1, 2$) only visit vertices in V_j (otherwise the makespan would be greater than B). Thus the two recursive calls to Partial satisfy the assumption: there is some schedule of vehicles Q_j serving D_j having makespan B. Inductively, the schedule returned by Partial for each $j = 1, 2$ has makespan at most $(16+16\rho)\cdot B$ and covers at least $\alpha^{\lg c}\cdot|D_j|$ demands from D_j, where $c \le \min\{|Q|-1, 2m\} \le z$. The schedules returned by the two recursive calls to Partial can clearly be run in parallel and this covers at least $\alpha^{\lg z}(|D_1| + |D_2|)$ demands, i.e. an $\alpha^{\lg z}$ fraction of D. So we have the desired performance in this case.

Random partitioning. The harder part of the algorithm is when Step (1) does not apply: so the MST length on Q is at most $3|Q|\cdot B$. Note that when the depots Q are contracted to a single vertex, the MST on the end-points of D plus the contracted depot-vertex has length at most $|Q|\cdot B$ (the optimal makespan schedule induces such a tree). Thus the MST on the depots Q along with end-points of D has length at most $4|Q|\cdot B$. Based on the assumption in Lemma 28 and the flow lower bound, we have $\sum_{i\in D} d(s_i, t_i) \le k|Q|\cdot B$. It follows that for the *single vehicle* Dial-a-Ride instance solved in Step (2), the Steiner and flow lower-bounds (denoted LB_{pmt} in Theorem 27) are $O(1)\cdot|Q|B$. Theorem 27 now implies that τ is a 1-preemptive tour τ servicing D, of length at most $O(\log^2 n)|Q|\cdot B$ such that $\sum_{i\in D} T_i \le O(\log n)\cdot|D|B$, where T_i denotes the total time spent in the vehicle by demand $i \in D$. The bound

on the delay uses the fact that $\max_{i=1}^{m} d(s_i, t_i) \leq B$.

Choosing a large enough constant corresponding to $\rho = \Theta(\log n \log m)$, the length of τ is upper bounded by $\rho|Q| \cdot B$ (since $n \leq 2m$). So the cutting procedure in Step (3) results in at most $|Q|$ pieces of τ, each of length at most $2\rho B$. The objects $i \in C''$ (as defined in Step (4)) are called a *cut objects*. We restrict attention to the other objects $C' = D \setminus C''$ that are not 'cut'. For each object $i \in C'$, the path traced by it (under single vehicle tour τ) from its source s_i to preemption-point and the path (under τ) from its preemption-point to t_i are both completely contained in pieces of $\mathcal{P}$. Figure 3.3 gives an example of objects in C' and C'', and the cutting procedure.

Claim 29. *The expected number of objects in C'' is at most $\sum_{i \in D} \frac{T_i}{\rho B} \leq O(\frac{1}{\log m}) \cdot |D|$.*

Proof: The probability (over choice of η) that object $i \in D$ is cut equals $\frac{T_i}{\rho B}$ where T_i is the total time spent by i in tour τ. Using linearity of expectation and $\sum_{i \in D} T_i \leq O(\log n) \cdot |D|B$, we have the claim. ∎

We can derandomize Step (3) and pick the best offset η (there are at most polynomially many combinatorially distinct offsets). Claim 29 implies (again choosing large enough constant in $\rho = \Theta(\log n \log m)$) that $|C'| \geq (1 - \frac{1}{2 \lg m})|D| \geq \alpha \cdot |D|$ demands are *not cut*. Now onwards we only consider the set C' of uncut demands. Let $\mathcal{P}$ denote the pieces obtained by cutting τ as above, recall $|\mathcal{P}| \leq |Q|$. A piece $P \in \mathcal{P}$ is said to be non-trivial if the vehicle in the 1-preemptive tour τ carries some C'-object while traversing P. Note that the number of non-trivial pieces in $\mathcal{P}$ is at most $2|C'| \leq 2m$: each C'-object appears in at most 2 pieces, one where it is moved from source to preemption-vertex and other from preemption-vertex to destination. Retain only the non-trivial pieces in $\mathcal{P}$; so $|\mathcal{P}| \leq \min\{|Q|, 2m\} = z$. The pieces in $\mathcal{P}$ may not be one-to-one assignable to the depots since the algorithm has not taken the depot locations into account. We determine which pieces may be assigned to depots by considering a matching problem between $\mathcal{P}$ and the depots in Step (5) and (6).

Load rebalancing. The bipartite graph H (defined in Step (5)) represents which pieces and depots may be assigned to each other. Piece $P \in \mathcal{P}$ and depot $f \in Q$ are assignable iff $d(f, P) \leq 2B$, and in this case graph H contains an edge (P, f). We claim that corresponding to the 'maximal contracting' set $\mathcal{S}$ (defined in Step (5)), the 2-matching π (in Step (6)) is guaranteed to exist. Note that $|\Gamma(\mathcal{S})| \leq \frac{|\mathcal{S}|}{2}$, but $|\Gamma(\mathcal{T})| > \frac{|\mathcal{T}|}{2}$ for all $\mathcal{T} \supset \mathcal{S}$. For any $T' \subseteq \mathcal{P} \setminus \mathcal{S}$, let $\tilde{\Gamma}(T')$ denote the neighborhood of T' in $Q \setminus \Gamma(\mathcal{S})$. The maximality of $\mathcal{S}$ implies: for any non-empty $T' \subseteq \mathcal{P} \setminus \mathcal{S}$,

$\frac{|\mathcal{S}|}{2} + \frac{|T'|}{2} = \frac{|\mathcal{S} \cup T'|}{2} < |\Gamma(\mathcal{S} \cup T')| = |\Gamma(\mathcal{S})| + |\tilde{\Gamma}(T')|$, i.e. $|\tilde{\Gamma}(T')| \geq \frac{|T'|}{2}$. Hence by *Hall's condition*, there is a 2-matching $\pi : \mathcal{P} \setminus \mathcal{S} \to Q \setminus \Gamma(\mathcal{S})$. The set $\mathcal{S}$ and 2-matching π can be easily computed in polynomial time.

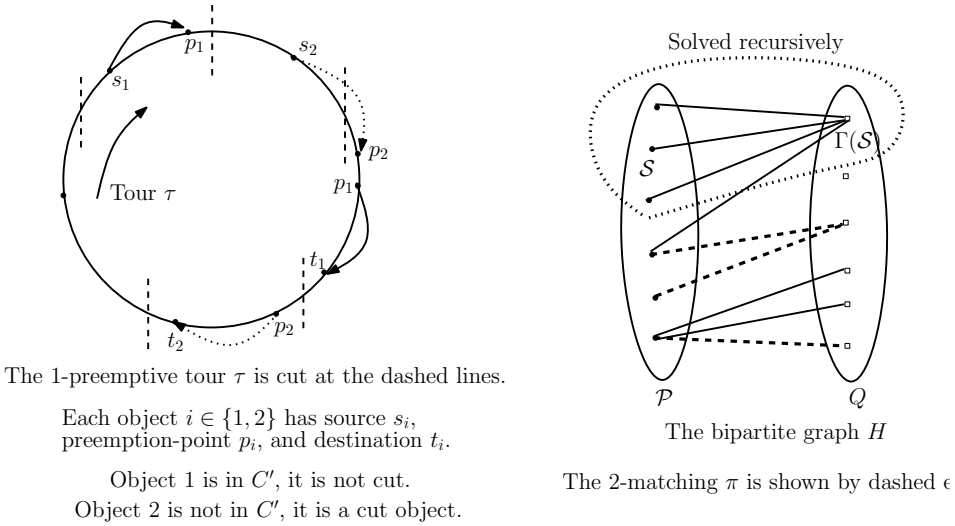


Figure 3.3: Cutting and patching steps in algorithm Partial.

Recursion. In Step (7), demands C' are further partitioned into two sets: C_1 consists of objects that are *either* picked-up *or* dropped-off in some piece of $\mathcal{S}$; and C_2-objects are picked-up *and* dropped-off in pieces of $\mathcal{P} \setminus \mathcal{S}$. The vehicles $\Gamma(\mathcal{S})$ suffice to serve all C_1 objects, as shown below.

Claim 30. *There exists a schedule of vehicles $\Gamma(\mathcal{S})$ serving demands C_1, having makespan B.*

Proof: Consider the schedule of makespan B that serves all demands $C' = C_1 \cup C_2$ using vehicles Q: this is implied by the promise on instance $\langle Q, D, B \rangle$. We claim that in this schedule, no vehicle from $Q \setminus \Gamma(\mathcal{S})$ moves any C_1 object. Suppose (for a contradiction) that the vehicle from depot $f \in Q \setminus \Gamma(\mathcal{S})$ moves object $i \in C_1$ at some point in this schedule; then it must be that $d(f, s_i)$ and $d(f, t_i) \leq 2B$. But since $i \in C_1$, at least one of s_i or t_i is in a piece of $\mathcal{S}$, and this implies that there is some piece $P \in \mathcal{S}$ with $d(f, P) \leq 2B$, i.e. $f \in \Gamma(\mathcal{S})$, which is a contradiction! Thus the only vehicles participating in the movement of C_1 objects are $\Gamma(\mathcal{S})$, which implies the claim. ∎

In the final schedule, a *large fraction* of C_1 demands are served by vehicles $\Gamma(\mathcal{S})$, and *all* the C_2 demands are served by vehicles $Q \setminus \Gamma(\mathcal{S})$. Figure 3.3 shows an example of this partition.

Serving C_1 demands. Based on Claim 30, the recursive call Partial $\langle \Gamma(\mathcal{S}), C_1, B \rangle$ (made in Step (8)) satisfies the assumption required in Lemma 28. Since $|\Gamma(\mathcal{S})| \leq \frac{|\mathcal{P}|}{2} \leq \frac{|Q|}{2} < |Q|$, we obtain inductively that Partial $\langle \Gamma(\mathcal{S}), C_1, B \rangle$ returns a schedule of makespan $(16 + 16\rho) \cdot B$ covering at least $\alpha^{\lg y} \cdot |C_1|$ demands of C_1, where $y = \min\{|\Gamma(\mathcal{S})|, 2m\}$. Note that $y \leq |\Gamma(\mathcal{S})| \leq |\mathcal{P}|/2 \leq z/2$ (as $|\mathcal{P}| \leq z$), which implies that at least $\alpha^{\lg z-1}|C_1|$ demands are covered.

Serving C_2 demands. These are served by vehicles $Q \setminus \Gamma(\mathcal{S})$ using the 2-matching π, in two rounds as specified in Step (8). This suffices to serve all objects in C_2 since for any $i \in C_2$, the paths traversed by object i under τ, namely $s_i \rightsquigarrow p_i$ (its preemption-point) and $p_i \rightsquigarrow t_i$ are contained in pieces of $\mathcal{P} \setminus \mathcal{S}$. Furthermore, since $|\pi^{-1}(f)| \leq 2$ for all $f \in Q \setminus \Gamma(\mathcal{S})$, the distance traveled by vehicle f in one round is at most $2 \cdot 2(2B + 2\rho B)$. So the time taken by this schedule is at most $2 \cdot 4(2B + 2\rho B) = (16 + 16\rho) \cdot B$.

The schedule of vehicles $\Gamma(\mathcal{S})$ (serving C_1) and vehicles $Q \setminus \Gamma(\mathcal{S})$ (serving C_2) can clearly be run in parallel. This takes time $(16 + 16\rho) \cdot B$ and covers in total at least $|C_2| + \alpha^{\lg z-1}|C_1| \geq \alpha^{\lg z-1}|C'| \geq \alpha^{\lg z}|D|$ demands of D. This proves the inductive step of Lemma 28. Using Lemma 28 repeatedly as mentioned earlier, we obtain an $O(\log^2 m \cdot \log n)$ approximation algorithm. Using some preprocessing steps (described in the next subsection), we have Theorem 21.

3.3.3 Weighted preemptive mDaR

The multi-vehicle Dial-a-Ride problem as initially defined assumes that all objects have the same 'weight', i.e. each object occupies a unit capacity. In the weighted mDaR problem, each object $i \in [m]$ also has a weight w_i, and the capacity constraint requires that no vehicle carry a total weight of more than k at any time. In this section, we obtain an $O(\log^3 n)$ approximation algorithm for weighted preemptive mDaR. The algorithm first guesses the optimal makespan B of the given instance. The algorithm serves the demands in two phases: the 1st phase involves pre-processing 'heavy demands' and has a makespan of $O(1) \cdot B$; the 2nd phase is identical to the algorithm Partial in Section 3.3.2 and covers all remaining demands. For every $u, v \in V$ let $\mathsf{dem}_{u,v}$ denote the total weight of objects having source u and destination v. Define $H = \{(u, v) \in V \times V \mid \mathsf{dem}_{u,v} \geq k/2\}$ to be the vertex-pairs

having *heavy demands*, and $\hat{H}$ the set of demands between pairs of H.

Phase I. In this pre-processing step, we cover all demands $\hat{H}$. Below we use the same notation $P \subseteq [q]$ for a set P of vehicles and the multi-set of depots corresponding to P. For a subset $D \subseteq H$ of heavy vertex-pairs, let $\hat{D}$ denote the set of demands between pairs of D. We describe an algorithm PreProc that is used to serve demands $\hat{H}$.

Lemma 31. *Given any subset $Q \subseteq [q]$ of vehicles and $D \subseteq H$ of heavy demand-pairs* such that *there is a schedule of vehicles Q covering demands $\hat{D}$ with makespan at most B,* PreProc$\langle Q, D\rangle$ *returns a schedule of vehicles Q covering demands $\hat{D}$ that has makespan $O(1) \cdot B$.*

Invoking PreProc$\langle [q], H\rangle$ gives the desired schedule covering $\hat{H}$ at makespan $O(1) \cdot B$. The algorithm PreProc and its analysis follow along the lines of algorithm Partial in Section 3.3.2. Lemma 31 is proved by induction on $|Q|$, the base case $|Q| = 1$ is trivial, and in the following we consider the inductive step (where $|Q| \geq 2$).

Algorithm PreProc first ensures that each edge of the minimum spanning tree on Q has length at most $3B$, otherwise the problem decouples into two disjoint smaller problems (as in Step (1) of Partial). Hence we also obtain that the MST on Q plus all sources/destinations in D has length at most $O(1) \cdot |Q|B$ (c.f. algorithm Partial). Demands with source u and destination v are referred to as (u, v) demands. For every $(u, v) \in D$, using a greedy procedure, one can partition all (u, v) demands such that the total weight of each part (except possibly the last) is between $\frac{k}{2}$ and k. For any $(u, v) \in D$, let $g_{u,v}$ denote the number of parts in this partition of (u, v) demands; note that $g_{u,v} \leq \frac{2}{k}\mathsf{dem}_{u,v} + 1 \leq \frac{4}{k}\mathsf{dem}_{u,v}$ (since $\mathsf{dem}_{u,v} \geq k/2$). The flow lower bound resulting from demands $\hat{D}$ equals $\frac{1}{|Q|k}\sum_{(u,v)\in D} \mathsf{dem}_{u,v} \cdot d(u, v) \geq \frac{1}{4|Q|}\sum_{(u,v)\in D} g_{u,v} \cdot d(u, v)$. In the rest of algorithm PreProc, we consider each part in the above partition of (u, v) demands (for $(u, v) \in D$) as a single object of weight k; so all the objects in one part will always be moved together. Scaling down the weights and capacity by k, we obtain the following equivalent unit-weight unit-capacity instance $\mathcal{U}$: for each $(u, v) \in D$ there are $g_{u,v}$ demands with weight 1, source u and destination v, and each vehicle in Q has capacity 1.

The flow lower bound of the demands $\hat{H}$ in the original instance implies that $\frac{1}{4|Q|}\sum_{(u,v)\in H} g_{u,v} \cdot d(u, v) \leq B$; i.e. the flow bound in $\mathcal{U}$ is at most $4|Q|B$. Since $\mathcal{U}$ has unit capacity and weights, we can use the *Stacker-crane* algorithm [58] to obtain

a single vehicle *non-preemptive* tour τ serving all demands, having length 1.8 times the preemptive lower bounds. The single vehicle preemptive lower bounds corresponding to $\mathcal{U}$ are: (Steiner bound) TSP on all sources/destinations in D, and (flow bound) $\sum_{(u,v)\in H} g_{u,v} \cdot d(u,v)$. As shown above, these are both $O(1) \cdot |Q|B$. The algorithm next cuts tour τ to obtain at most $|Q|$ pieces $\mathcal{P}$ such that each piece has length $O(1) \cdot B$. In addition it can be ensured that *no* demand is 'cut': since the vehicle carries at most one object at any time and each source-destination distance is at most B. Next, construct bipartite graph $\mathcal{H}$ with vertex sets $\mathcal{P}$ and Q and an edge between piece $P \in \mathcal{P}$ and depot $f \in Q$ iff $d(f,P) \leq 2B$; $\Gamma(A)$ denotes the neighborhood of any $A \subseteq \mathcal{P}$ in $\mathcal{H}$. The algorithm finds any maximal set $\mathcal{S} \subseteq \mathcal{P}$ with $|\Gamma(\mathcal{S})| < |\mathcal{S}|/2$ (as in Step (5) of Partial). This implies a 2-matching $\pi : \mathcal{P} \setminus \mathcal{S} \to Q \setminus \Gamma(\mathcal{S})$ such that there exists a schedule of vehicles $\Gamma(\mathcal{S})$ serving demands in the pieces $\mathcal{S}$ with makespan B (c.f. Claim 30). Let $C \subseteq D$ denote the heavy demand-pairs served in some piece of $\mathcal{S}$. The final schedule returned by PreProc$\langle Q, D\rangle$ involves: (i) schedule for vehicles $Q \setminus \Gamma(\mathcal{S})$ given by π (covering demand-pairs $D \setminus C$); and (ii) schedule for vehicles $\Gamma(\mathcal{S})$ obtained recursively PreProc$\langle \Gamma(\mathcal{S}), C\rangle$ (covering demands C). The recursively obtained schedule has makespan $O(1) \cdot B$ by the induction hypothesis (since $|\Gamma(\mathcal{S})| < |Q|$, and $\langle \Gamma(\mathcal{S}), C\rangle$ satisfies the assumption in Lemma 31); hence the final schedule also has makespan $O(1) \cdot B$, which proves the inductive step.

Phase II. Let $L = \{(u,v) \in V \times V \mid 0 < \mathsf{dem}_{u,v} < k/2\}$ be the vertex-pairs having *light demands*. The algorithm for this phase treats all (u,v) demands as a *single object* of weight $\mathsf{dem}_{u,v}$ from u to v; so there are $m = |L| \leq n^2$ distinct objects. The algorithm is identical to Partial of Section 3.3.2 for the unweighted case: Theorem 27 generalizes easily to the weighted case, and the preemptive lower bounds stay the same after combining demands in L. Thus we obtain an $O(\log^3 n)$ approximate schedule that covers all remaining demands (setting $m \leq n^2$).

Theorem 32. *There is an $O(\log^3 n)$ approximation algorithm for weighted preemptive* mDaR.

Credits: This chapter is based on joint work with Inge Li Gørtz and R.Ravi.

Chapter 4

Stochastic Demands Vehicle Routing

4.1 Introduction

The capacitated vehicle routing problem (CVRP) [73] is defined on an n-vertex metric (V, d) with root/depot $r \in V$, and involves distributing (identical) items from the depot to other vertices using a single vehicle of capacity $Q \in \mathbb{N}$. There is a demand of $q_v \in \{0, 1, \cdots, Q\}$ at each vertex $v \in V$. The objective is to find a minimum length tour of the vehicle that satisfies all demands such that the vehicle carries at most Q units at any time. There are two versions of the problem: in the *split-delivery* CVRP, the demand at a vertex may be satisfied by multiple visits of the vehicle; whereas in *unsplit-delivery* CVRP, the entire demand at a vertex must be satisfied in a single visit by the vehicle.

We consider the situation where the demands are uncertain. Here the exact demands $\{q_v\}_{v \in V}$ are not known in advance, instead we are only given a demand distribution $\mathcal{D}$. The actual demand at a vertex is observed only when that vertex is visited. In this setting, a feasible solution is any strategy of visiting vertices using the capacitated vehicle such that all realized demands are satisfied. We note that this strategy may be *adaptive*, i.e. at any point in the tour, the next vertex to visit may depend on the demands observed until then. The *stochastic vehicle routing* problem (SVRP) is: given metric (V, d) with root $r \in V$, a vehicle of capacity $Q \in \mathbb{N}$, and demand distribution $\mathcal{D}$ over $\{0, 1, \cdots, Q\}^V$, compute a strategy of visiting vertices (starting and ending at r) that satisfies all realized demands, and minimizes the *expected* tour length.

The most general setting of SVRP is where we are only given a black-box access

to the distribution $\mathcal{D}$. However as we show in Section 4.4, no $o(n)$ approximation ratio can be achieved for the black-box model, assuming a polynomial number of samples from $\mathcal{D}$. We consider the following two natural ways of describing the demand distribution, that make SVRP more tractable.

- **Independent demand distribution.** Here, the demand at each vertex $v \in V$ is specified by a random variable ξ_v (in the range $\{0, 1, \cdots, Q\}$), and the random variables $\{\xi_v\}_{v \in V}$ are independent of each other. This setting is extensively studied in the operations research literature [145, 16, 137], and is known as the *vehicle routing problem with stochastic demands* (VRPSD).
- **Explicit demand distribution.** Here we are given m demand scenarios, where each scenario $i \in [m]$ specifies demands $q_v^i \in \{0, 1, \cdots, Q\}$ at all vertices $v \in V$ and a probability p_i (where $\sum_{i=1}^{m} p_i = 1$). For each scenario $i \in [m]$, let $\overline{q}^i \in \{0, 1, \cdots, Q\}^V$ denote the vector of demands $\{q_v^i\}_{v \in V}$. In the explicit-demand setting, the realized demands will always be one of $\overline{q}^1, \cdots, \overline{q}^m$, where $\overline{q}^i$ occurs with probability p_i (for all $i \in [m]$).

We refer to any tour that starts and ends at r as an r-tour.

4.1.1 Results

We first consider SVRP under independent demand distributions. We obtain a randomized approximation algorithm for VRPSD achieving the following worst-case guarantees: $(1 + \alpha)$ for *split-delivery VRPSD*, and $(2 + \alpha)$ for *unsplit-delivery VRPSD*, where α denotes the best approximation ratio for the traveling salesman problem. This result matches (up to an additive $o(1)$ term) the corresponding best known guarantees for the deterministic CVRP. Furthermore, this algorithm gives an *a priori* strategy, that visits all vertices in a fixed order. For the split-delivery VRPSD, [16] had suggested a 'cyclic heuristic' and conjectured that it achieves a constant approximation guarantee; we show that this is indeed the case by giving a $1 + 2\alpha$ worst-case bound.

In Section 4.2 we consider SVRP under explicit demands, and present an $O(\log^2 n \cdot \log m)$ approximation algorithm (recall, n is the number of vertices and m is the number of scenarios). Our algorithm applies to both split and unsplit delivery versions of SVRP. We show that even a special case of SVRP under explicit demands is at least as hard to approximate as the 'latency group Steiner tree problem', for which there is an $\Omega(\log^{1-\epsilon} n)$ hardness of approximation [77] and $O(\log^2 n)$ is the

best known approximation guarantee. The main step of this algorithm lies in solving the related *metric isolation problem* that seeks to identify the realized scenario at minimum expected cost (formal definition in Section 4.3).

The adaptive strategy for the metric isolation problem proceeds in $O(\log m)$ phases, where in each phase it reduces the number of candidate scenarios by a constant factor: so that after the last phase, it uniquely identifies the realized scenario. In each phase of this algorithm, we solve a variant of the *group Steiner tree* problem. We show that the LP-rounding for group Steiner tree combined with the greedy approach for the *minimum latency problem* [29, 45] gives an $O(\log^2 n)$ approximation algorithm for the 'latency group Steiner tree' problem. This leads to an $O(\log^2 n \cdot \log m)$ approximation for metric isolation. We also observe that the approximability of this problem is at least as hard as *group Steiner tree* [61], for which $O(\log^2 n \cdot \log m)$ is the best known approximation ratio, and there is an $\Omega(\log^{2-\epsilon} n)$ hardness of approximation [77].

We note that in the special case of a weighted-star metric, the metric isolation problem corresponds to the 'optimal split tree' problem [99] (discussed further in Section 4.3.3). In this case, our approximation guarantee improves to $O(\log m)$. Hence we also obtain an $O(\log m)$-approximation algorithm for the split tree problem (where m is the number of 'items'), answering an open question from [2].

Finally in Section 4.4, we study SVRP under black-box distributions. We show that any algorithm that makes at most n^c samples from the black-box distribution (for any $c \ll \frac{n}{\log n}$) has an approximation guarantee $\Omega(\frac{n}{c})$. This is an information theoretic lower bound and requires no assumption on the running time of the algorithm. The basic idea is that a small number of samples from the distribution is insufficient to learn enough about it to output a good adaptive strategy (whereas an optimal strategy that knows the precise distribution can perform much better).

Remark. A crucial difference in the explicit-demand and independent-demand models is the following. As argued in Section 4.2, under independent demands, the minimum length TSP on metric (V, d) is a lower bound. However the optimal value of SVRP under explicit demand distributions may be much smaller than the minimum length TSP. This partly explains the difference in approximability of these two versions of SVRP.

4.1.2 Related Work

Capacitated VRP. Let $\alpha \geq 1$ denote the best approximation guarantee for the *Traveling Salesman Problem*; so $\alpha = \frac{3}{2}$ for general metrics [34], and $\alpha = 1 + \epsilon$ (for any constant $\epsilon > 0$) for constant dimensional Euclidean metrics [11, 108]. The best known approximation guarantee for split-delivery CVRP is $1 + \alpha \cdot (1 - \frac{1}{Q})$ [73, 6], and for unsplit-delivery CVRP is $2 + \alpha \cdot (1 - \frac{2}{Q})$ [5]. These bounds have been improved slightly (when $Q \geq 3$) to $1 + \alpha \cdot (1 - \frac{1}{Q}) - \frac{1}{3Q^3}$ and $2 + \alpha \cdot (1 - \frac{2}{Q}) - \frac{1}{3Q^3}$ respectively [19].

Independent demands VRP. SVRP under independent stochastic demands (known as VRPSD in the literature) is extensively studied from a computational viewpoint [145, 42, 14, 43, 137]. In terms of worst-case bounds, a 'cyclic heuristic' for split-delivery VRPSD was suggested in [16]. This was shown to achieve a $(1 + \alpha + o(1))$-approximation in the special case of *identical demand distributions*; for general distributions, it was shown to be a $Q + \alpha$ approximation, and obtaining a tighter bound was left open. Bertsimas [16] considered two settings for VRP with independent demands: only the first setting (called Strategy **a**) applies to VRPSD as defined in this chapter. The second setting (Strategy **b**) [16] is more along the lines of *a priori* optimization, which is discussed next.

Optimal Split Trees. When the underlying metric is a star, the metric isolation problem turns out to be the optimal split tree problem, which was first studied in [99]. We give a formal definition and discuss this problem in Section 4.3.3. Kosaraju et al. [99] gave a greedy algorithm for metric isolation on *unweighted star-metrics* that achieves an $O(\log m)$ approximation ratio (recall, m is the number of scenarios). Later, Adler and Heeringa [2] analyzed the same greedy algorithm for this problem and gave an $\ln m + 1$ worst-case bound, and showed that this result also holds for weighted star-metrics but under a *uniform probability* distribution. Chakravarthy et al. [24] showed that the optimal split tree problem is $\Omega(\log m)$ hard to approximate in general (this uses a non-uniform probability distribution); and that it is hard to approximate better than 4 when the probability distribution is uniform. This line of work leaves open the question of approximating general optimal split trees [2] (i.e. metric isolation on weighted star-metrics with non-uniform probabilities).

***a priori* Traveling Salesman.** This is another stochastic model for the TSP [91, 15], which is quite different from our setting. Here we are given a metric (V, d) and distribution $\mathcal{D}$ on the demands; for TSP (which is CVRP when $Q = \infty$), a scenario is just a subset of vertices that need to be visited, and $\mathcal{D}$ is some distribution over

subsets of V. The goal is to compute an *a priori* (or a master) tour τ through all vertices of V, that minimizes the expectation (over scenarios in $\mathcal{D}$) of the length of tour τ after short-cutting it to include only the vertices of the realized scenario. This corresponds to a situation where demands at all vertices are known *before* the vehicle starts its route (so that it can shortcut accordingly); however, we require a *single* master tour to be computed beforehand. On the other hand, in SVRP (that we study in this chapter) the demand at any vertex is revealed only when that *vertex is visited*; and we allow the vehicle-route to be *adaptive*. The complexity of *a priori* TSP also depends on how the distribution $\mathcal{D}$ is represented. Schalekamp and Shmoys [134] give an $O(\log n)$ approximation for this problem even under black-box distributions; in fact their master tour does not even depend on the distribution. Shmoys and Talwar [141] give a randomized 4-approximation (and a deterministic 8-approximation) algorithm for *a priori* TSP under independent demands; a constant factor approximation for this setting can also be inferred from [64].

4.2 SVRP under Independent Demands

In this section, we consider SVRP under independent demand distributions. We are given a metric (V, d) with root $r \in V$, and a demand distribution $\mathcal{D}$ specified by a random variable ξ_v (in the range $\{0, 1, \cdots, Q\}$) for each vertex $v \in V$, where the random variables $\{\xi_v\}_{v \in V}$ are independent of each other. The objective is to compute a strategy of visiting vertices (starting and ending at r) that serves all realized demands and minimizes the expected tour-length. This is known as the *vehicle routing problem with stochastic demands* (VRPSD) in the literature, eg. [145, 16, 137]. Throughout this section, we let α denote the best approximation ratio for the TSP.

Without loss of generality, we may assume that none of the demand random variables is identically zero (such vertices may be safely ignored). Under this assumption, any feasible policy must visit every vertex with probability 1: otherwise, since demands are independent, there is a non-zero probability that some demand is not satisfied (implying that the policy is infeasible). Therefore the minimum length TSP tour on all vertices in metric (V, d) is a lower bound for VRPSD.

This suggests the following obvious strategy of serving demands: (1) first visit all vertices using an α-approximate TSP tour and *observe* all demands; (2) then (since exact demands are known) serve them using the approximation algorithm

for CVRP [73, 5, 6]. This gives approximation ratios of $(2\alpha + 1)$ for split-delivery SVRP, and $(2\alpha + 2)$ for unsplit-delivery SVRP.

We improve over the above naïve approach in two ways. Firstly, our algorithm achieves improved approximation ratios, that match the best known for the deterministic versions. Secondly, our algorithm also works in the setting where demands are observed only when they are served (see also Section 4.3.4); whereas the naïve approach clearly depends on observing demand before serving it.

Given the demand realization q_i at each vertex $i \in V$, $\mathsf{LB}(q) := \frac{2}{Q} \sum_{i \in V} q_i \cdot d(r, i)$ is a lower bound on the optimal solution length for split-delivery VRP (c.f. Haimovich and Kan [73]). Hence the expected value, $E[\mathsf{LB}(q)] = \frac{2}{Q} \sum_{i \in V} E[\xi_i] \cdot d(r, i)$ is a lower bound for VRPSD (both split/unsplit delivery versions). This lower bound and the TSP bound were also used in Bertsimas [16].

We now present our algorithms for VRPSD.

Theorem 33. *There is a randomized $(1 + \alpha)$-approximation algorithm for VRPSD with split-deliveries.*

Proof: The algorithm SplitALG proceeds as follows.

1. Compute an α approximate TSP tour τ on all vertices.
2. Number vertices such that r is 0 and τ visits vertices in order $0, 1, 2, \cdots, |V| - 1$.
3. Choose a uniformly random demand $q_0 \in [0, Q]$ as demand at the depot.
4. The vehicle starts at vertex 0 (filled to capacity Q), and for $i = 0, 1, \cdots, |V| - 1$ does:
   (a) Let $\tilde{Q}_i$ be the units (of the item) carried by the vehicle when it visits vertex i.
   (b) Let q_i be the demand observed at vertex i, as it is being served.
   (c) If $q_i \leq \tilde{Q}_i$ then serve the demand at i and move on to vertex $i + 1$ (with $\tilde{Q}_{i+1} \leftarrow \tilde{Q}_i - q_i$).
   (d) If $q_i > \tilde{Q}_i$ then serve $\tilde{Q}_i$ units of demand at i and make a visit to-and-from r:
   - The vehicle fills up to Q units at r, returns to i and serves the remaining $q_i - \tilde{Q}_i$ demand at i.

   Then move on to vertex $i + 1$ with $\tilde{Q}_{i+1} \leftarrow Q + \tilde{Q}_i - q_i$.

We will bound the expected length of the solution obtained by the above algorithm. In the analysis, we first *condition* on the realization q_i of demands at all

vertices $i \in V \setminus \{r\}$. The demand $q_r = q_0$ at the depot r remains uniformly random in $[0, Q]$. A vertex $i \in V$ is called a *break-point* if the vehicle executes a refill trip to r from i (i.e. Step 4d applies at vertex i). This happens precisely when the vehicle becomes empty at vertex i while there is still unserved demand at i. Observe that vertex i is a break-point iff there is $p \in \mathbb{N}$ such that $\sum_{j<i} q_j \leq p \cdot Q < q_i + \sum_{j<i} q_j$. Since q_0 (the artificial demand at r) is the only random variable, and it is is uniform in $[0, Q]$, we have:

$$\Pr[i \text{ is break-point}] = \frac{q_i}{Q}$$

The solution length (conditioned on demands $\{q_i\}_{i \in V \setminus \{r\}}$) equals $d(\tau) + 2 \sum_{i \neq r} d(r, i) \cdot I(i \text{ break-point})$, where $I(i \text{ break-point})$ is the indicator random variable for i being a break-point (for each $i \neq r$). Hence the expected solution length (conditioned on the demands $\{q_i\}_{i \in V \setminus \{r\}}$) is:

$$d(\tau) + 2 \sum_{i \neq r} \Pr[i \text{ is break-point}] \cdot d(r, i) = d(\tau) + \frac{2}{Q} \sum_{i \neq r} q_i \cdot d(r, i) = d(\tau) + \mathsf{LB}(q)$$

Finally, the expected (unconditional) solution length equals $d(\tau) + E[\mathsf{LB}(q)]$. Since τ is an α-approximate minimum TSP tour and $E[\mathsf{LB}(q)]$ is a lower bound for VRPSD, the expected length of this solution is at most $(1 + \alpha)$ times the optimal value of the VRPSD instance. Note that algorithm SplitALG only observes demand at a vertex while serving it; so it gives a feasible strategy even in the $\mathsf{SVRP}_{\mathsf{obs}}$ setting. ∎

Theorem 34. *There is a randomized $(2 + \alpha)$-approximation algorithm for VRPSD with unsplit-deliveries.*

Proof: The algorithm UnsplitALG is very similar to the split-delivery case and is given below.

1. Compute an α approximate TSP tour τ on all vertices.
2. Number vertices such that r is 0 and τ visits vertices in order $0, 1, 2, \cdots, |V| - 1$.
3. Choose a uniformly random demand $q_0 \in [0, Q]$ as demand at the depot.
4. The vehicle starts at vertex 0 (filled to capacity Q), and for $i = 0, 1, \cdots, |V| - 1$ does:
   (a) Let $\tilde{U}_i$ be the units (of the item) carried by the vehicle when it visits vertex i.
   (b) Let q_i be the demand observed at vertex i.
   (c) If $q_i \leq \tilde{U}_i$ then serve demand i and move on to vertex $i+1$ (with $\tilde{U}_{i+1} \leftarrow \tilde{U}_i - q_i$).

(d) If $q_i > \tilde{U}_i$ then make *two* visits to-and-from r:

- In the first visit, the vehicle fills up to q_i units at r and serves demand i.
- In the second visit, the vehicle fills up to $Q + \tilde{U}_i - q_i$ units at r, and returns.

Then move on to vertex $i+1$ with $\tilde{U}_{i+1} \leftarrow Q + \tilde{U}_i - q_i$.

We first condition on the realization q_i of demands at all vertices $i \in V$. Again, vertex $i \in V$ is called a *break-point* if the vehicle executes a refill trip to r from i (i.e. Step 4d applies at vertex i). We claim that the breakpoints encountered by Algorithms SplitALG and UnsplitALG (for the same realization of demands) are *identical*. This follows from the observation that for all vertices $i \in V$, $\tilde{Q}_i$ (in SplitALG) equals $\tilde{U}_i$ (in UnsplitALG).

Now conditioning only on demands $\{q_i\}_{i \in V \setminus \{r\}}$, from the proof of Theorem 33, we have Pr[i is break-point] $= \frac{q_i}{Q}$ for algorithm UnsplitALG as well. Note that the solution length in UnsplitALG equals $d(\tau) + 4 \sum_{i \neq r} d(r,i) \cdot I(i \text{ break-point})$, where $I(i \text{ break-point})$ is the indicator random variable for i being a break-point (for each $i \neq r$). Hence the expected solution length (conditioned on $\{q_i\}_{i \in V \setminus \{r\}}$) is:

$$d(\tau) + 4 \sum_{i \neq r} \Pr[i \text{ is break-point}] \cdot d(r,i) = d(\tau) + 2 \cdot \frac{2}{Q} \sum_{i \neq r} q_i \cdot d(r,i) = d(\tau) + 2 \cdot \mathsf{LB}(q)$$

Unconditionally, the expected solution length from UnsplitALG is $d(\tau) + 2 \cdot E[\mathsf{LB}(q)]$. Noting that τ is an α-approximate TSP tour, and $E[\mathsf{LB}(q)]$ is a lower bound for even split-delivery VRPSD, we obtain that UnsplitALG achieves a $(2+\alpha)$ approximation for unsplit-delivery VRPSD. ∎

We now consider the *cyclic heuristic* for split-delivery VRPSD suggested in [16]. This involves computing an α-approximate TSP tour τ, and visiting vertices in the order given by τ while returning to the depot to fill up whenever the vehicle is empty. This is precisely algorithm SplitALG without the randomly chosen demand at the depot (i.e. without Step 3). In fact, the algorithm suggested in [16] considers n different strategies: based on rotating the vertex ordering in τ (obtained by varying the first vertex visited from r), and picks the one with least expected value. We prove a constant-factor upper bound for even the (possibly weaker) algorithm that only considers a single vertex ordering on τ. It was shown [16] that the cyclic heuristic (using n rotations) achieves a worst case guarantee of $\alpha + 1 + o(1)$ in the case of *identical demand distributions* at all vertices, and a $Q + \alpha$ guarantee for general demands ([16], Theorem 4). Moreover it was conjectured that the worst case guarantee of the cyclic heuristic is a constant (independent of Q) even with

general demand distributions. The following theorem shows that this is indeed the case.

Theorem 35. *The cyclic heuristic is a* $(1 + 2\alpha)$*-approximation algorithm for split-delivery VRPSD.*

Proof: Let τ visit vertices in the order $0, 1, 2, \cdots, |V| - 1$ with r being numbered 0. As in the proof of Theorem 33, we first condition on the demand realization q_i at each vertex $i \in V$. A vertex $i \in V$ is called a *break-point* if the vehicle executes a refill trip to r from i. Let U be the set of all break-points (including r) and $|U| = u$. Then the length of the vehicle's route is $d(\tau) + 2\sum_{w \in U} d(r, w)$. We now establish the following key claim.

Claim 36. *We have* $2 \cdot \sum_{w \in U} d(r, w) \leq d(\tau) + \frac{2}{Q} \sum_{v \in V} q_v \cdot d(r, v)$.

Proof: Let the break-points U consist of $r = \beta_0, \beta_1, \cdots, \beta_{u-1}$ in that order along τ. For any $l \in \{0, 1, \cdots, u - 1\} := [u]$, let τ_l denote the portion of tour τ between vertices β_l and β_{l+1} (the indices are modulo u). Note that τ is the concatenation $\tau_0 \cdot \tau_1 \cdots \tau_{u-1}$, i.e. $\sum_{l=0}^{u-1} d(\tau_l) = d(\tau)$. For each $l \in [u]$, define a subtour originating and ending at r as $\pi_l := (r, \beta_l) \cdot \tau_l \cdot (\beta_{l+1}, r)$. Observe that the route τ' traced by the vehicle is precisely the concatenation $\pi_0 \cdot \pi_1 \cdots \pi_{u-1}$. Since the vehicle capacity is Q and it makes refill-trips only when it runs out of items, the vehicle delivers exactly Q units in each segment π_l (for $0 \leq l \leq u - 2$), and $Q' \leq Q$ units in the last segment π_{u-1}. For each $l \in [u]$ and vertex $i \in \tau_l$, let $C_l(i)$ denote the number of units delivered at vertex i by the vehicle in segment π_l. For technical reasons, set $C_{u-1}(r) := Q - Q'$. From the preceding discussion, we obtain:

$$\sum_{i \in \tau_l} C_l(i) = Q, \ \forall l \in [u], \qquad \text{and} \qquad \sum_{l | i \in \tau_l} C_l(i) = q_i, \ \forall i \in V \setminus \{r\} \tag{4.1}$$

Consider any fixed segment π_l (for $l \in [u]$). For any vertex $i \in \tau_l$, let $t(i, \beta_l)$ (resp. $t(i, \beta_{l+1})$) denote the length along τ_l, from β_l to i (resp. i to β_{l+1}). It follows that $t(i, \beta_l) + t(i, \beta_{l+1}) = d(\tau_l)$ for all vertices $i \in \tau_l$. By the triangle inequality, we have:

$$\left.\begin{array}{rcccl} d(\beta_l, r) & \leq & d(\beta_l, i) + d(i, r) & \leq & t(\beta_l, i) + d(i, r) \\ d(\beta_{l+1}, r) & \leq & d(\beta_{l+1}, i) + d(i, r) & \leq & t(\beta_{l+1}, i) + d(i, r) \end{array}\right\} \text{ for all vertices } i \in \tau_l$$

Taking a convex combination of the first (resp. second) set of inequalities, with multiplier $C_l(i)/Q$ for each $i \in \tau_l$, we obtain the following. (Equation (4.1) implies

that these are indeed convex multipliers.)

$$d(\beta_l, r) \leq \sum_{i \in \tau_l} \frac{C_l(i)}{Q} \cdot (t(\beta_l, i) + d(i, r)); \qquad d(\beta_{l+1}, r) \leq \sum_{i \in \tau_l} \frac{C_l(i)}{Q} \cdot (t(\beta_{l+1}, i) + d(i, r))$$

Adding these two inequalities (using properties of $t(\cdot)$ and $C_l(\cdot)$ from above),

$$\begin{aligned} d(\beta_l, r) + d(\beta_{l+1}, r) &\leq \sum_{i \in \tau_l} \frac{C_l(i)}{Q} \cdot (t(\beta_l, i) + t(\beta_{l+1}, i)) + 2 \sum_{i \in \tau_l} \frac{C_l(i)}{Q} \cdot d(i, r) \\ &= d(\tau_l) \sum_{i \in \tau_l} \frac{C_l(i)}{Q} + \frac{2}{Q} \sum_{i \in \tau_l} C_l(i) \cdot d(i, r) \\ &= d(\tau_l) + \frac{2}{Q} \sum_{i \in \tau_l} C_l(i) \cdot d(i, r) \end{aligned}$$

Finally adding the above inequality over $l \in [u]$, where the indices l are modulo u,

$$\begin{aligned} 2 \cdot \sum_{w \in U} d(w, r) &= \sum_{l=0}^{u-1} (d(\beta_l, r) + d(\beta_{l+1}, r)) \leq \sum_{l=0}^{u-1} d(\tau_l) + \frac{2}{Q} \sum_{i \in V} \sum_{l | i \in \tau_l} C_l(i) \cdot d(i, r) \\ &\leq d(\tau) + \frac{2}{Q} \sum_{i \in V \setminus r} q_i \cdot d(i, r) \end{aligned}$$

The last inequality uses Equation (4.1). Thus we obtain the claim. ∎

Claim 36 gives $2 \cdot \sum_{w \in U} d(r, w) \leq d(\tau) + \mathsf{LB}(q)$, which implies that conditioned on demands $\{q_i\}_{i \in V}$, the solution length is at most $2 \cdot d(\tau) + \mathsf{LB}(q)$. Taking expectations, we obtain the desired bound on the cyclic heuristic. ∎

Remarks: We note that our algorithms for VRPSD do not even require knowledge of the demand distributions at different vertices: it suffices to know just which vertices have a non-zero demand distribution. This information is used to compute the α-approximate TSP tour τ on relevant vertices. Furthermore, the algorithms suggest an *a priori* strategy that involves visiting vertices in the order given by τ.

Theorem 33 also gives an *a priori* strategy for SVRP under black-box distributions, having expected tour length at most $\alpha \cdot \mathsf{Tsp} + \mathsf{OPT}$, where OPT is the optimal value of the SVRP instance and Tsp is the minimum length tour on all vertices in (V, d). This strategy does not even depend on the demand-distribution. However Tsp is not a lower bound for SVRP in general, hence this algorithm does not provide

any multiplicative approximation guarantee. In fact, as shown in Section 4.4, any algorithm making at most polynomial number of samples achieves only an $\Omega(n)$-approximation for SVRP under black-box distributions.

4.3 SVRP under Explicit Demands

In explicit demands SVRP, the distribution $\mathcal{D}$ is given by m scenarios, where each scenario $i \in [m]$ specifies demands $q_v^i \in \{0, 1, \cdots, Q\}$ at all vertices $v \in V$ and its associated probability p_i (such that $\sum_{i=1}^m p_i = 1$). For each scenario $i \in [m]$, $\overline{q}^i \in \{0, 1, \cdots, Q\}^V$ denotes the vector $\{q_v^i\}_{v \in V}$ of demands. We assume (without loss of generality) that $\overline{q}^i \neq \overline{q}^j$ for all $1 \leq i < j \leq m$. The objective is to compute an adaptive strategy of visiting vertices starting from r (and finally returning to r) such that all realized demands are satisfied, and the expected tour length is minimized. For ease of exposition, we focus only on the split delivery version of SVRP; it is easy to extend our algorithm to the unsplit delivery version as well. We note that the minimum length TSP is not a lower bound for SVRP under explicit demands; hence an approach as in Section 4.2 does not suffice here.

Our algorithm for SVRP relies on the closely related *metric isolation problem* (Metric Isolation) which we now define. The input to the metric isolation problem is the same as that for SVRP: metric (V, d) with root $r \in V$, and a demand distribution $\mathcal{D}$ specified by m demand-vectors $\{\overline{q}^i\}_{i=1}^m$ with associated probabilities $\{p_i\}_{i=1}^m$ (where $\sum_{i=1}^m p_i = 1$). However in this case, we seek a strategy of visiting vertices (starting and ending at r) that precisely determines (i.e. 'isolates') the realized scenario among $[m]$, and has the minimum expected tour length. Again this strategy may be adaptive. Note that unlike SVRP we *do not* require the realized demands to be satisfied; it suffices to observe demands at vertices and stop once the realized scenario is inferred.

Definition 37. *Any feasible solution to* Metric Isolation *can be represented as a* decision tree T *with nodes labeled by vertices of the metric, where:*

1. *The root node of T and all leaf-nodes are labeled r.*
2. *Every internal node v of T has at most $Q + 1$ children $\{c_i(v) \mid 0 \leq i \leq Q\}$, where $c_i(v)$ (if present) denotes the next vertex to be visited from v if the demand observed at v equals i.*

3. *For each scenario $i \in [m]$, there is a* distinct leaf node l_i *in T such that when decision tree T is run with realized demands $\overline{q}^i$, the path from the root node to l_i is traced in T.*

For each $i \in [m]$, let π_i denote the path traced in T under scenario i, i.e. the path from root node to leaf l_i; this also corresponds to the r-tour traversed in the metric when i is the realized scenario. The expected isolation cost of the strategy given by decision tree T is then:

$$\mathsf{IsolateTime}(T) = \sum_{i=1}^{m} p_i \cdot d(\pi_i). \tag{4.2}$$

The first condition implies that the route traced by the vehicle under each scenario is an r-tour. The second condition is the definition of a decision tree. The crucial condition is the last one, which ensures that each scenario is correctly isolated. We may assume that every node in T lies in the path traced under some scenario; otherwise the subtree under such nodes can be pruned from T. So the last condition also implies that the leaf nodes of T are in one-to-one correspondence with the scenarios $[m]$.

Lemma 38. *A ρ-approximation algorithm for* Metric Isolation *implies a $(\rho + \frac{5}{2})$-approximation algorithm for SVRP.*

Proof: Note that by the assumption $\overline{q}^i \neq \overline{q}^j$ for all distinct $i, j \in [m]$, Metric Isolation is always feasible: the realized scenario can be correctly determined by observing demands at all vertices. Let σ denote an optimal adaptive strategy for the given SVRP instance, and OPT its expected length. We show below that the optimal value of Metric Isolation is at most OPT. For each $i \in [m]$, when σ is run under scenario i, let $\overline{S}^i$ denote the demands that are observed at those vertices visited by σ. The feasibility of σ implies that $\overline{S}^i$ contains all the non-zero demands in $\overline{q}^i$. It follows that $\overline{S}^i \neq \overline{S}^j$ for all $1 \leq i < j \leq m$ (since $\overline{q}^i \neq \overline{q}^j$). Thus the demands observed when σ is executed, identify the realized scenario uniquely. In other words σ induces a feasible solution to Metric Isolation of value OPT.

The algorithm for SVRP returns the following strategy, that runs in two phases: (1) Solve Metric Isolation (using the ρ-approximation algorithm), and execute the resulting strategy to determine the realized scenario (this phase only observes demands); (2) Then serve only the demands in the realized scenario using the 2.5 approximation algorithm for CVRP [73]. The expected cost spent in the second

phase is at most $\frac{5}{2} \cdot \sum_{i=1}^{m} p_i \cdot C_i^* \leq \frac{5}{2} \cdot \mathsf{OPT}$, where C_i^* (for each $i \in [m]$) denotes the minimum length CVRP tour that serves demands $\overline{q}^i$. Furthermore, as observed above, the optimal value of Metric Isolation is at most OPT. Hence the expected cost in the first phase is at most $\rho \cdot \mathsf{OPT}$. This finishes the proof of the lemma.

We note that for unsplit-delivery SVRP, we could just use the corresponding algorithm [6] in phase (2), to obtain a $\rho + \frac{7}{2}$ approximation algorithm. ∎

Based on this lemma, it suffices to study Metric Isolation. In the next subsection, we study two variants of the group Steiner tree problem [61] that are useful in the approximation algorithm for Metric Isolation that we present in Section 4.3.2.

Remark. The strategy for SVRP given in Lemma 38 first explores the metric space and observes demands (while not serving them), and then serves just the realized scenario. This reduction depends crucially on the fact that the demand at any vertex may be observed by just *visiting* it, as opposed to actually *serving* its demand. We show in Section 4.3.4 that our algorithm can be used to obtain an $O(\log^2 n \cdot \log m)$ approximate adaptive strategy even in the (more constrained) setting where any demand is determined only upon serving it.

4.3.1 Two Auxiliary Problems

The input to the *group Steiner tree problem* (GST) consists of metric (V, d) with root $r \in V$ and groups $\{X_i \subseteq V\}_{i=1}^{g}$; the goal is to compute a minimum length r-tour covering all groups. A group $i \in [g]$ is said to be covered if any vertex from X_i is visited by the tour. The best approximation guarantee known for GST is $O(\log^2 n \cdot \log g)$ [61], and there is an $\Omega(\log^{2-\epsilon} n)$ hardness of approximation [77]. In this section, we study two variants of group Steiner tree that are useful in obtaining an approximation algorithm for Metric Isolation.

In the *latency* group Steiner tree problem, in addition to the GST input we are given non-negative weights for each group, and the objective is to compute an r-tour covering all groups that minimizes the sum of the weighted arrival times (the arrival time of a group is the distance from r along the tour to the first vertex in that group). This objective is closely related to Metric Isolation, which measures the expected time to isolate a scenario, although in an adaptive (rather than linear) fashion; this observation is formalized and used in the algorithm of Section 4.3.2.

In the group Steiner *orienteering* problem, apart from the GST input there are profits on groups and a length bound; the goal is to compute an r-tour of length at most B that covers the maximum possible profit. It is well-known (see [18, 29, 45] on the basic latency problem) that approximations for the orienteering objective lead to approximations for the latency objective. Following this, we start with an algorithm for the group Steiner orienteering. Then we show how this implies an algorithm for latency group Steiner, in fact for a generalization that is encountered in Section 4.3.2.

Group Steiner Orienteering

The *group Steiner orienteering* (GSO) problem takes as input, a metric (V, d) with root $r \in V$, g groups of vertices $\{X_i \subseteq V\}_{i=1}^{g}$ with associated profits $\{v_i\}_{i=1}^{g}$, and a length bound B. The goal is to compute an r-tour of length at most B that collects the maximum possible profit. We show that the deterministic algorithm for group Steiner tree [26] can be used within a standard greedy framework to obtain the following result for GSO. An algorithm for GSO is said to be an (a, b) bicriteria approximation if on any instance of the problem (as above), it outputs a solution of length at most $b \cdot B$ that obtains at least $\frac{1}{a}$ times the profit obtained by any r-tour of length B.

Theorem 39. *There is a $(4, O(\log^2 n))$ bicriteria approximation algorithm for* GSO*, where n is the number of vertices in the metric.*

Proof: Let OPT denote the optimal profit of the given GSO instance. We first preprocess the given metric to only include vertices within distance $B/2$ of the root r (this preserves the optimal solution). Thus the profit contained in any single vertex is at most OPT. Since the sum of profits at all vertices is at least $\sum_{i=1}^{g} v_i$, we have $\frac{1}{n}\sum_{i=1}^{g} v_i \leq \mathsf{OPT}$. Our algorithm for GSO follows a standard greedy approach (see eg. Garg [62]). Below we set $\alpha = O(\log^2 n)$ where the precise constant comes from the analysis.

1. Solution $S \leftarrow \emptyset$. Mark all groups as uncovered.
2. Until the length of S exceeds $\alpha \cdot B$, do:
   (a) Set residual profits:

$$\tilde{v}_i := \begin{cases} 0 & \text{for each covered group } i \in [g] \\ v_i & \text{for each uncovered group } i \in [g] \end{cases}$$

(b) Solve the following LP for the residual GSO to obtain solution (x, y):

$$\begin{array}{lll} \max & \sum_{i=1}^{g} \tilde{v}_i \cdot y_i & \\ \text{s.t.} & x(\delta(S)) \geq y_i & \forall S \subseteq V : r \notin S,\ X_i \subseteq S;\ \ \forall i \in [g] \\ & y_i \leq 1 & \forall i \in [g] \\ & \sum_e d_e \cdot x_e \leq B & \\ & x, y \geq 0 & \end{array}$$

(c) As described in [26] (using probabilistic tree embedding), obtain a spanning tree T in metric (V, d) and capacities x_T on edges of T such that: $\sum_{e \in T} d_e \cdot x_T(e) \leq O(\log n) \cdot \sum_e d_e \cdot x_e \leq O(\log n) \cdot B$, and x_T supports at least y_i flow from r to X_i for each $i \in [g]$.

(d) Round down each $x_T(e)$ to an integral multiple of $\frac{1}{n^3}$.

(e) For each group $i \in [g]$, let y'_i be the maximum flow value from r to X_i under capacities x_T, and let $f_i^T(e)$ (for all $e \in T$) denote the flow variables realizing this flow.

(f) Using x_T and f_i^Ts, run the deterministic algorithm for 'density group Steiner' (Section 3.2 of [26]) to obtain r-tree A covering groups $\hat{A}$ such that:

$$\frac{d(A)}{\sum_{i \in \hat{A}} \tilde{v}_i} \leq \alpha \cdot \frac{B}{\sum_{i=1}^{g} \tilde{v}_i \cdot y'_i}, \qquad \text{where } \alpha = O(\log^2 n)$$

(g) If $d(A) \leq \alpha B$ then $S' \leftarrow S \cup A$.

(h) If $d(A) > \alpha B$ then: (1) Partition tree A into at most $2 \cdot \frac{d(A)}{\alpha B}$ subtrees, each of length at most αB; let A' denote the subtree containing maximum profit. (2) Set $S' \leftarrow S \cup A' \cup \{f\}$ where f is any edge from r to A'.

(i) $S \leftarrow S'$. Mark all groups visited in S as covered.

3. Output an Euler tour on r-tree S.

We now prove that this algorithm achieves a $(4, 4\alpha + 2)$ bicriteria approximation. It suffices to show that the final solution S has length at most $(2\alpha + 1) \cdot B$ and obtains profit at least $\frac{\mathsf{OPT}}{4}$.

The increase in length of S in any iteration is at most $(\alpha + 1) \cdot B$ (recall every vertex is at distance at most B from r); so the final length $d(S) \leq (2\alpha + 1) \cdot B$. We claim that the final solution S contains at least $\frac{\mathsf{OPT}}{4}$ profit. At any iteration, let $p(S)$

denote the profit of S, and $d(S)$ its length. Since $d(S) > \alpha B$ upon termination, it suffices to show that at all iterations in the above algorithm,

$$p(S) \geq \min\left\{\frac{\mathsf{OPT}}{4}, \frac{\mathsf{OPT}}{2\alpha B} \cdot d(S)\right\} \tag{4.3}$$

We show (4.3) inductively: the base case $S = \emptyset$ is trivial. Consider any iteration where $p(S) < \mathsf{OPT}/4$ (otherwise the claim is trivial). Hence the optimal value of the LP solved in this iteration $\sum_{i=1}^{g} \tilde{v}_i \cdot y_i \geq \frac{3}{4} \cdot \mathsf{OPT}$. After the pruning step (that rounds down capacities) we reduce the capacity of each edge in T by at most $\frac{1}{n^3}$. Since any cut in the tree T has at most n edges, the capacity of any cut decreases by at most $\frac{1}{n^2}$ after Step 2d; and by the max-flow min-cut theorem, $y'_i \geq y_i - \frac{1}{n^2}$ for each $i \in [g]$ (in Step 2e). Furthermore, for any $i \in [g]$, all the flow variables f_i^T corresponding to a maximum $r - X_i$ flow are integral multiples of $\frac{1}{n^3}$ (since edge capacities x_T are). This latter condition is necessary (due to technical reasons in [26]) for the algorithm used in Step 2f. We now have:

$$\sum_{i=1}^{g} \tilde{v}_i \cdot y'_i \geq \sum_{i=1}^{g} \tilde{v}_i \cdot y_i - \frac{1}{n^2}\sum_{i=1}^{g} \tilde{v}_i \geq \frac{3}{4} \cdot \mathsf{OPT} - \frac{1}{n^2}\sum_{i=1}^{g} v_i \geq \frac{3}{4} \cdot \mathsf{OPT} - \frac{\mathsf{OPT}}{n} \geq \frac{\mathsf{OPT}}{2}$$

where the second last inequality follows from $\frac{1}{n}\sum_{i=1}^{g} v_i \leq \mathsf{OPT}$ (by the preprocessing). Hence $d(A)/p(A) \leq 2\alpha\frac{B}{\mathsf{OPT}}$. We finish by handling the two possible cases (Steps 2g and 2h):

- $d(A) \leq \alpha B$. Now $p(S') = p(S) + p(A) \geq \frac{\mathsf{OPT}}{2\alpha B} \cdot d(S) + \frac{\mathsf{OPT}}{2\alpha B} \cdot d(A) = \frac{\mathsf{OPT}}{2\alpha B} \cdot d(S')$.
- $d(A) > \alpha B$. By averaging, we have $p(A') \geq \frac{\alpha B}{2 \cdot d(A)} p(A) \geq \frac{\mathsf{OPT}}{4}$; so $p(S') \geq \frac{\mathsf{OPT}}{4}$.

In either case we have Equation (4.3) for solution S'. ∎

Partial Latency Group Steiner

In the *partial latency group Steiner* (Min-LPGST) problem, we are given a metric (V, d), g groups of vertices $\{X_i \subseteq V\}_{i=1}^{g}$ with associated weights $\{w_i\}_{i=1}^{g}$, root $r \in V$, and a target $h \leq g$. A group $i \in [g]$ is said to be *covered* (or visited) by r-tour τ if any vertex in X_i is visited, and the arrival time of such a group i is the length of the shortest prefix of τ that contains an X_i-vertex. The arrival times of all uncovered groups are set to be the tour-length. The weighted sum of arrival times of all groups

is termed *latency* of the tour. The objective in Min-LPGST is to compute a minimum latency r-tour that covers at least h groups. We present an $(O(\log^2 n), 4)$ bicriteria approximation algorithm for Min-LPGST, i.e. the solution tour visits at least $\frac{h}{4}$ groups and has latency at most $O(\log^2 n)$ times the optimal latency of a tour that visits h groups. In the following, let Lat^* denote the optimal value of the given instance of Min-LPGST, and OPT^* the length of this tour. The approximation algorithm for Min-LPGST proceeds as follows (below, we set $\beta := \frac{5}{4}$, and $\rho = O(\log^2 n)$ such that the algorithm of Theorem 39 is a $(4, \rho)$ bicriteria approximation for GSO).

1. Guess an integer l such that $\beta^{l-1} < \mathsf{OPT}^* \leq \beta^l$.
2. Mark all groups as *uncovered*.
3. For $i = 1, \cdots, l$ do:
   (a) Run the GSO algorithm (Theorem 39) on the instance with groups $\{X_i\}_{i=1}^g$, root r, length bound β^{i+1}, and profits:
   $$v_i := \begin{cases} 0 & \text{for each covered group } i \in [g] \\ w_i & \text{for each uncovered group } i \in [g] \end{cases}$$
   (b) Let $\tau^{(i)}$ denote the r-tour obtained above. Mark all groups visited by $\tau^{(i)}$ as *covered*.
4. Construct tour $\tau \leftarrow \tau^{(1)} \cdots \tau^{(l)}$, the concatenation of the r-tours found in the above iterations. If $d(\tau) < \rho \cdot \beta^l$ then increase its length to $\rho \cdot \beta^l$ (this may only increase the latencies of groups).
5. Run the GSO algorithm on the instance with groups $\{X_i\}_{i=1}^g$, root r, length bound β^l, and *unit profit* for each group. Let σ denote the resulting r-tour.
6. Output the tour $\pi := \tau \cdot \sigma$ as solution to the Min-LPGST instance.

Claim 40. *The tour τ has length $\Theta(\rho) \cdot \mathsf{OPT}^*$ and latency $O(\rho) \cdot \mathsf{Lat}^*$.*

Proof: The guarantee $O(\rho) \cdot \mathsf{OPT}^*$ on the length bound is immediate from Theorem 39: the length of each $\tau^{(i)}$ is at most $\rho \cdot \beta^i$. It is also clear that $d(\tau) \geq \rho \cdot \mathsf{OPT}^*$, by the increase in Step 4. The proof for bounding the latency is identical to the analysis for the *Minimum Latency Problem* [29, 45]. We include it here for completeness.

Fix an optimal solution ζ to the Min-LPGST instance, where $d(\zeta) = \mathsf{OPT}^* \in (\beta^{l-1}, \beta^l]$. For each $i \in [l]$, let N_i^* denote the total weight of groups visited in ζ by

time β^i; note that N_l^* equals the total weight of the groups covered by ζ. Similarly, for each $i \in [l]$, let N_i denote the total weight of groups visited in $\tau^{(1)} \cdots \tau^{(i)}$ (i.e. by iteration i of the algorithm). Set $N_0 = N_0^* := 0$, and $W := \sum_{i=1}^{g} w_i$ the total weight of all groups. The latency of tour τ is upper bounded by $T := \frac{\rho}{\beta-1} \sum_{i=0}^{l} \beta^{i+2} \cdot (W - N_i)$. Note also that the latency of tour ζ, $\mathsf{Lat}^* \geq \frac{\beta-1}{\beta} \sum_{i=0}^{l} \beta^i (W - N_i^*)$.

Consider any iteration $i \in [l]$ of the algorithm in Step 3. Note that the optimal value of the GSO instance solved in this iteration is at least $N_i^* - N_{i-1}$: the β^i length prefix of tour ζ corresponds to a feasible solution to this GSO instance. Theorem 39 implies that the profit obtained in $\tau^{(i)}$, i.e. $N_i - N_{i-1} \geq \frac{1}{4} \cdot (N_i^* - N_{i-1})$, i.e. $W - N_i \leq \frac{3}{4} \cdot (W - N_{i-1}) + \frac{1}{4} \cdot (W - N_i^*)$. Using this,

$$\begin{aligned}
(\beta - 1)\frac{T}{\rho} &= \sum_{i=0}^{l} \beta^{i+2} \cdot (W - N_i) \\
&\leq 4W + \frac{1}{4} \sum_{i=1}^{l} \beta^{i+2} (W - N_i^*) + \frac{3}{4} \sum_{i=1}^{l} \beta^{i+2} (W - N_{i-1}) \\
&\leq O(1) \cdot \mathsf{Lat}^* + \frac{3\beta}{4} \sum_{i=0}^{l-1} \beta^{i+2} (W - N_i) \\
&\leq O(1) \cdot \mathsf{Lat}^* + \frac{3\beta}{4} \cdot (\beta - 1)\frac{T}{\rho}
\end{aligned}$$

Since $\beta = \frac{5}{4}$, this implies $T = O(\rho) \cdot \mathsf{Lat}^*$, giving the claim. ∎

Remark: The above arguments also give an $O(\log^2 n)$-approximation algorithm for the *minimum latency group Steiner* problem which is Min-LPGST when $h = g$. Here the objective is to minimize the sum of weighted arrival times at all groups (every group has to be visited). The algorithm for this problem just runs Step 3 until *all* groups are covered, instead of stopping after l iterations. A proof identical to that in Claim 40 implies an $O(\log^2 n)$ approximation. Furthermore, as shown in Theorem 55, an α-approximation to latency group Steiner implies an $O(\alpha \cdot \log g)$-approximation to group Steiner tree. Hence an improvement over the $O(\log^2 n)$ ratio for latency group Steiner would also improve the best known bound for the usual group Steiner tree problem.

Claim 41. *The tour σ covers at least $\frac{h}{4}$ groups and has length $O(\rho) \cdot \mathsf{OPT}^*$.*

Proof: Directly from the guarantee on GSO algorithm $\mathcal{A}$ (Theorem 39) invoked on the instance in Step 5: here the optimal profit is at least h, as witnessed by the optimal solution to the Min-LPGST instance (which has length at most 2^l). ∎

Theorem 42. *Tour π covers at least $\frac{h}{4}$ groups and has latency $O(\rho) \cdot \mathsf{Lat}^*$. Hence there is an $(O(\log^2 n), 4)$ bicriteria approximation algorithm for* Min-LPGST*, where n is the number of vertices in the metric.*

Proof: Since π visits all the vertices in σ, Claim 41 implies that π covers at least $\frac{h}{4}$ groups. For each group $i \in [g]$, let α_i denote its arrival time under tour τ (note that $\alpha_i = d(\tau)$ for any group i that is not covered by τ). Claim 40 implies that the latency of tour τ, $\sum_{i=1}^{g} w_i \cdot \alpha_i = O(\rho) \cdot \mathsf{Lat}^*$. Observe that for each group i that is covered in τ, its arrival time under tour π is also α_i. For any group j *not* covered in τ, its arrival time under π is at most $d(\pi)$ (the tour length) whereas $\alpha_j = d(\tau)$. Note that the final tour length $d(\pi) = d(\tau) + d(\sigma) = \Theta(\rho) \cdot \mathsf{OPT}^*$ and $d(\tau) \geq \rho \cdot \mathsf{OPT}^*$ (by Step 4). Hence, the arrival time under π of each group $i \in [g]$ is $O(1) \cdot \alpha_i$. Using Claim 40, we obtain the theorem. ∎

4.3.2 Algorithm for the Metric Isolation Problem

Recall that an instance of Metric Isolation is specified by metric (V, d), root $r \in V$, and m scenarios $\{\overline{q}^i\}_{i=1}^m$ with respective probabilities $\{p_i\}_{i=1}^m$. We give a recursive algorithm for solving Metric Isolation. Given an input instance as above, we will deal with sub-instances given by some subset $M \subseteq [m]$ of scenarios with probabilities $\{s_i\}_{i \in M}$ where $\sum_{i \in M} s_i = 1$; we refer to such an instance as $\langle M, \{s_i\}_{i \in M} \rangle$. Below we describe a recursive algorithm IsoAlg that represents an adaptive strategy of isolating the realized scenario amongst M. In the following, $[Q] := \{0, 1, \cdots, Q\}$.

The algorithm IsoAlg proceeds in several phases. In each phase, it maintains a candidate set M of scenarios such that the realized scenario lies in M. Upon observing demands along the tour produced by algorithm Partition (in Step 2), a new set $M' \subseteq M$ containing the realized scenario is identified such that the number of candidate scenarios reduces by a constant factor (i.e. $|M'| \leq \frac{3}{4} \cdot |M|$); then IsoAlg recurses on scenarios M'. After $O(\log m)$ such phases the realized scenario would be correctly identified.

Algorithm IsoAlg$\langle M, \{s_i\}_{i \in M} \rangle$**:**

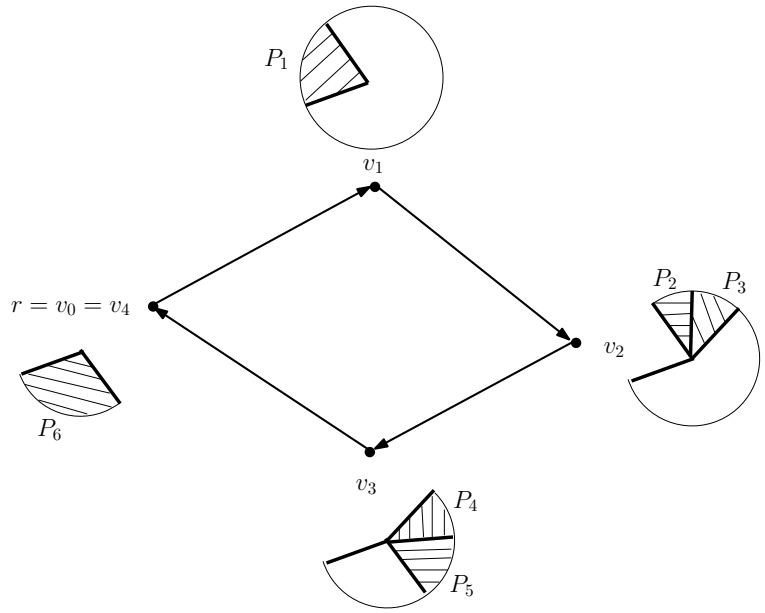
1. If $|M| = 1$, return this unique scenario as realized.
2. Apply algorithm Partition$\langle M, \{s_i\}_{i \in M} \rangle$ to obtain r-tour τ given by $\langle r = v_0, v_1, v_2, \cdots, v_l, v_{l+1} = r \rangle$ and an associated partition $\{P_k\}_{k=1}^t$ of M.

3. For each $k \in [t]$, let $s'_k := \sum_{i \in P_k} s_i$.
4. Traverse tour τ and return directly to r after the first vertex that reveals the realized scenario to be in some part P_k (for $k \in [t]$).
5. Run IsoAlg$\langle P_k, \{\frac{s_i}{s'_k}\}_{i \in P_k}\rangle$ to determine the realized scenario.

The next algorithm Partition seeks to find an r-tour τ such that after observing demands on τ, the candidate number of scenarios is reduced by a constant factor. Note that each vertex v corresponds to a partition $\mathcal{P}_v = \{F_{v,y}\}_{y=0}^{Q}$ of M depending on the observed demand at v (defined formally in Step 1 below). If the largest part in $\mathcal{P}_v$ (hence any other part) was at most a constant fraction of M, then visiting vertex v suffices to always reduce the number of candidate scenarios by this constant factor. However the largest part in $\mathcal{P}_v$ may be almost all of M, in which case the tour has to visit more vertices before substantially reducing the number of candidate scenarios. This is the basic idea behind the group Steiner instance constructed in this algorithm.

Algorithm Partition$\langle M, \{s_i\}_{i \in M}\rangle$**:**

1. For each $v \in V$ and $y \in [Q]$, define $F_{v,y} := \{i \in M \mid \overline{q}_v^i = y\}$, and
$$y(v) \leftarrow \arg\max\{|F_{v,y}| : 0 \le y \le Q\}; \qquad D_v \leftarrow M \setminus F_{v,y(v)}$$
2. For each $i \in M$, set $X_i \leftarrow \{v \in V \mid i \in D_v\}$.
3. Run the algorithm (Section 4.3.1) for the Min-LPGST instance on metric (V, d) with root r, groups $\{X_i\}_{i \in M}$ with weights $\{s_i\}_{i \in M}$, and target $|M| - 1$. Let $\tau := \langle r = v_0, v_1, v_2, \cdots, v_l, v_{l+1} = r\rangle$ be the r-tour obtained.
4. Obtain partition $\{P_k\}_{k=1}^{t}$ of M as follows (Figure 4.1 gives an example):
   (a) Initialize $k \leftarrow 0$, $N \leftarrow M$.
   (b) For $j = 1, \cdots, l$ do:
      - Define $R_x^j := N \cap F_{v_j,x}$ for all $x \in [Q] \setminus \{y(v_j)\}$, partitioning $N \cap D(v_j)$.
      - Let $R_{(1)}^j, \cdots, R_{(e)}^j$ be non-empty parts in $\{R_x^j \mid x \in [Q] \setminus \{y(v_j)\}\}$.
      - For each $f = 1, \cdots, e$, set $P_{k+f} \leftarrow R_{(f)}^j$ and $\nu(k+f) \leftarrow j$.
      - Set $k \leftarrow k + e$, and $N \leftarrow N \setminus D(v_j)$.
   (c) If $N \neq \emptyset$ then, set $P_{k+1} \leftarrow N$, $\nu(k+1) \leftarrow l+1$ and $k \leftarrow k+1$.



Tour τ is $\langle r = v_0, v_1, v_2, v_3, v_4 = r \rangle$.

The shaded regions denote the parts P_ks defined along the tour τ.

Figure 4.1: An illustration of Step 4 of Partition.

(d) Set $t \leftarrow k$.

5. Return tour τ along with the partition $\{P_k\}_{k=1}^{t}$.

Constructing decision tree from IsoAlg**.** Note that the adaptive strategy IsoAlg implicitly defines a strategy tree as described in Definition 37. We create a path $(r, v_1, v_2, \cdots, v_l, v_{l+1} = r)$, and for each $j \in [l+1]$ hang from node v_j, all the subtrees created in recursive calls to the following instances:

$$\langle P_k, \{\frac{s_i}{s'_k}\}\rangle, \qquad \text{for all } k \in [t] \text{ with } \nu(k) = j.$$

Analyzing algorithms IsoAlg **and** Partition

We first prove the following property that captures 'subadditivity' of the Metric Isolation objective function. This explains why the recursive approach in IsoAlg

works.

Claim 43. *For any partition $\{P_k\}_{k=1}^t$ of M and $s'_k = \sum_{i \in P_k} s_i$ (for $k \in [t]$), we have $\sum_{k=1}^t s'_k \cdot \mathsf{OPT}(\langle P_k, \{\frac{s_i}{s'_k}\}_{i \in P_k}\rangle) \leq \mathsf{OPT}(\langle M, \{s_i\}_{i \in M}\rangle)$.*

Proof: Let T denote the decision tree representing the optimal strategy for the Metric Isolation instance $\mathcal{I}_0 := \langle M, \{s_i\}_{i \in M}\rangle$. Recall the definition of leaf-node l_i and path π_i (for each scenario $i \in M$) from the beginning of Section 4.3. For each $k \in [t]$, let $\mathcal{I}_k := \langle P_k, \{\frac{s_i}{s'_k}\}_{i \in P_k}\rangle$. Consider a feasible decision tree for $\mathcal{I}_k$ given by T induced on the leaf-nodes $\{l_i \mid i \in P_k\}$ (note that it is indeed feasible since T isolates all scenarios $\cup_{k=1}^t P_k$). Observe that the expected cost of this strategy is $\sum_{i \in P_k} \frac{s_i}{s'_k} \cdot d(\pi_i)$; hence $\mathsf{OPT}(\mathcal{I}_k) \leq \sum_{i \in P_k} \frac{s_i}{s'_k} \cdot d(\pi_i)$. Summing over all parts $k \in [t]$,

$$\sum_{k=1}^{t} s'_k \cdot \mathsf{OPT}(\mathcal{I}_k) \leq \sum_{k=1}^{t} s'_k \cdot \sum_{i \in P_k} \frac{s_i}{s'_k} \cdot d(\pi_i) = \sum_{i \in M} s_i \cdot d(\pi_i) = \mathsf{OPT}(\mathcal{I}_0)$$

The second last equality uses the fact that $\{P_k\}_{k=1}^t$ is a partition of M. ∎

We now prove that the algorithm is well-defined and establish its performance guarantee. For any instance $\mathcal{J}$ of Metric Isolation, let $\mathsf{OPT}(\mathcal{J})$ denote its optimal value. Let the original Metric Isolation instance be $\mathcal{I} = \langle [m], \{p_i\}_{i \in [m]}\rangle$.

Claim 44. *The optimal value of the* Min-LPGST *instance in Step 3 of algorithm* Partition$\langle M, \{s_i\}_{i \in M}\rangle$ *is at most* $\mathsf{OPT}(\langle M, \{s_i\}_{i \in M}\rangle)$.

Proof: Let $\mathcal{I}_0 = \langle M, \{s_i\}_{i \in M}\rangle$ denote the given Metric Isolation instance, and T an optimal decision tree corresponding to it. Note that by definition of the sets $\{D_v\}_{v \in V}$, any internal node in T labeled vertex v has one child (namely $c_{y(v)}$) that corresponds to the realized scenario being in $F_{v,y(v)} = M \setminus D_v$. Consider a path σ traced from the root of T that always moves from node v to the child corresponding to $M \setminus D_v$, until it reaches a leaf-node; let nodes in σ be labeled $r, u_1, u_2, \cdots, u_l, r$ (this defines an r-tour in the metric). Since T is a feasible decision tree for the Metric Isolation instance, there is a unique scenario $a \in M$ such that when T is run under demands $\overline{q}_a$, it traces the root-leaf path σ. In other words, every scenario $b \in M \setminus \{a\}$ gives rise to a root-leaf path (the path π_b in Definition 37) that diverges from σ. However from the construction of σ, it follows that the scenarios diverging from it are precisely $\cup_{j=1}^l D_{u_j} = M \setminus \{a\}$.

Now consider the r-tour $r, u_1, u_2, \cdots, u_j, r$ in metric (V, d) (corresponding to σ) as a solution to the Min-LPGST instance. The number of groups covered is

$|\cup_{j=1}^{l} D_{u_j}| = |M| - 1$, so it is indeed feasible. Furthermore, it is easy to see from the definition of the isolation cost (4.2) that the latency of this solution is at most $\mathsf{IsolateTime}(T)$, i.e. at most $\mathsf{OPT}(\mathcal{I}_0)$. ∎

Consider the tour $\tau := \langle r = v_0, v_1, \cdots, v_l, r = v_{l+1}\rangle$ obtained in Step 3 of Partition. For every $i \in M$, let α_i denote the arrival time for group i in tour τ (see Section 4.3.1 for the definition). Also let $\mathsf{Lat}(\tau) := \sum_{i\in M} s_i \cdot \alpha_i$ denote the latency of tour τ. Theorem 42 and Claim 44 imply that τ covers $|\cup_{j=1}^{l} D_{v_j}| \geq \frac{|M|}{4}$ groups, and:

$$\mathsf{Lat}(\tau) \leq \rho \cdot \mathsf{OPT}(\langle M, \{s_i\}_{i\in M}\rangle), \qquad \text{where } \rho := O(\log^2 n) \tag{4.4}$$

We now show some properties of the partition $\{P_k\}_{k=1}^{t}$ constructed in Step 4.

Claim 45. *After Step 4 of* Partition*, we have $|P_k| \leq \frac{3}{4}|M|$ for each $k \in [t]$.*

Proof: We first show that $|P_k| \leq \frac{1}{2}|M|$ for every part P_k produced in the iterations of Step 4b. This is immediate from the observation that for all vertices v and $x \in [Q] \setminus \{y(v)\}$, $|F_{v,x}| \leq \frac{|M|}{2}$ (since $F_{v,y(v)}$ is the largest sized part among $\{F_{v,0}, F_{v,1}, \cdots, F_{v,Q}\}$).

There may be at most one part that is produced after Step 4b, i.e. the last P_t in Step 4c. From the construction, it follows that this part is precisely $M \setminus \cup_{j=1}^{l} D(v_j)$. In other words, these are all the uncovered groups in the solution τ for the Min-LPGST instance. Theorem 42 implies that τ covers at least $\frac{1}{4}(|M|-1)$ groups; hence $|M \setminus \cup_{j=1}^{l} D(v_j)| \leq \frac{3}{4}|M|$. So this extra part has size at most $\frac{3}{4}|M|$. ∎

Claim 46. *After Step 4 of* Partition*, for all $k \in [t]$ and $i \in P_k$, the arrival time of group i in tour τ is $\alpha_i = \sum_{j=1}^{\nu(k)} d(v_{j-1}, v_j)$.*

Proof: This is immediate from the construction of partition $\{P_k\}_{k=1}^{t}$. For all parts P_k (with $k \in [t]$) that are produced in Step 4b of Partition, it is clear that $\nu(k)$equals the index of the first vertex in τ that contains groups of P_k. Hence the claim follows for all groups in these parts.

For the (possibly absent) single part with ν-value $l+1$ produced in Step 4c of Partition, the groups in this part are *not* covered by τ. So their arrival time equals the tour length $\sum_{j=1}^{l+1} d(v_{j-1}, v_j)$, again as required. ∎

Claim 47. *At the end of Step 4 of* IsoAlg$\langle M, \{s_i\}_{i\in M}\rangle$ *the realized scenario lies in P_k. The expected length traversed in this step is at most $2 \cdot \mathsf{Lat}(\tau) \leq 2\rho \cdot \mathsf{OPT}(\langle M, \{s_i\}_{i\in M}\rangle)$.*

Proof: Visiting any vertex v reveals which part among $\{F_{v,x} \mid x \in [Q]\}$ the realized scenario lies in. Now consider the construction of partition $\{P_k\}_{k=1}^{t}$ in Step 4 of Partition. Note that for any $j \in [l]$, $\{P_k \mid \nu(k) = j\}$ are all the parts added in the j-th iteration of Step 4b; also let $\{P_k \mid \nu(k) = l+1\}$ (possibly empty) denote the single extra part added in Step 4c. Clearly $\sqcup_{j=1}^{l+1}\{P_k \mid \nu(k) = j\} = \{P_k\}_{k=1}^{t}$.

This implies that when vertex v_j (any $j \in [l+1]$) is visited along τ, it is determined which part (if any) among $\{P_k \mid \nu(k) = j\}$ the realized scenario lies in. Thus the traversal of τ in Step 4 of IsoAlg returns to r from vertex v_j (for $j \in [l+1]$) iff the realized scenario lies in one of $\{P_k \mid \nu(k) = j\}$. Hence P_k correctly contains the realized scenario after Step 4 of IsoAlg$\langle M, \{s_i\}_{i\in M}\rangle$.

From the above arguments, it follows that for any $j \in [l+1]$, the probability of returning to r from vertex v_j (in Step 4 of IsoAlg) is precisely $\sum_{k\in[t]:\nu(k)=j} \sum_{i\in P_k} s_i$.

So the expected length traversed is:

$$\begin{aligned}
&\leq 2 \cdot \sum_{j=1}^{l+1} \left[\left(\sum_{h=1}^{j} d(v_{h-1}, v_h) \right) \cdot \left(\sum_{k\in[t]:\nu(k)=j} \sum_{i\in P_k} s_i \right) \right] && (4.5)\\
&= 2 \cdot \sum_{k=1}^{t} \left[\left(\sum_{h=1}^{\nu(k)} d(v_{h-1}, v_h) \right) \cdot \sum_{i\in P_k} s_i \right] && (4.6)\\
&= 2 \cdot \sum_{k=1}^{t} \sum_{i\in P_k} \alpha_i \cdot s_i && (4.7)\\
&= 2 \cdot \sum_{i\in M} \alpha_i \cdot s_i && (4.8)
\end{aligned}$$

Inequality (4.5) follows since the tour-length when returning from v_j is at most $2\left(\sum_{h=1}^{j} d(v_{h-1}, v_h)\right)$. Equality (4.6) is just an interchange of summation, equality (4.7) is from Claim 46, and equality (4.8) uses the fact that $\{P_k\}_{k=1}^{t}$ partitions M. The last expression is Lat(τ), and from Equation (4.4), we have the claim. ∎

Lemma 48. *The expected length of the strategy given by* IsoAlg$\langle M, \{s_i\}_{i\in M}\rangle$ *is at most* $2\rho \cdot (\log_{4/3} |M|) \cdot \mathsf{OPT}(\langle M, \{s_i\}_{i\in M}\rangle)$.

Proof: We prove this by induction on $|M|$. The base case of $|M| = 1$ is trivial: zero length is traversed in this case. In the following assume $|M| \geq 2$. Let $\mathcal{I}_0 :=$

$\langle M, \{s_i\}_{i\in M}\rangle$, and for each $k \in [t]$, let $\mathcal{I}_k := \langle P_k, \{\frac{s_i}{s'_k}\}_{i\in P_k}\rangle$. From Claim 47, the expected length spent in Step 4 of IsoAlg$(\mathcal{I}_0)$ is at most $2\rho \cdot \mathsf{OPT}(\mathcal{I}_0)$. For any $k \in [t]$, since $|P_k| \leq \frac{3}{4}|M|$ (by Claim 45), the inductive hypothesis implies that the expected length of IsoAlg$(\mathcal{I}_k)$ is at most:

$$2\rho \cdot \log_{4/3}|P_k| \cdot \mathsf{OPT}(\mathcal{I}_k) \leq 2\rho(\log_{4/3}|M| - 1) \cdot \mathsf{OPT}(\mathcal{I}_k).$$

The probability of recursing on $\mathcal{I}_k$ is exactly s'_k for each $k \in [t]$. So the expected length of IsoAlg$(\mathcal{I}_0)$ is at most:

$$\begin{aligned} & 2\rho \cdot \mathsf{OPT}(\mathcal{I}_0) + \sum_{k=1}^{t} s'_k \cdot 2\rho \cdot (\log_{4/3}|M| - 1) \cdot \mathsf{OPT}(\mathcal{I}_k) \\ \leq\ & 2\rho \cdot \mathsf{OPT}(\mathcal{I}_0) + 2\rho \cdot (\log_{4/3}|M| - 1) \cdot \mathsf{OPT}(\mathcal{I}_0) \\ =\ & 2\rho \cdot \log_{4/3}|M| \cdot \mathsf{OPT}(\mathcal{I}_0) \end{aligned}$$

where the inequality uses Claim 43. ∎

Thus for any Metric Isolation instance given by metric (V, d), root r, and scenarios $\{\overline{q}^i, p_i\}_{i=1}^m$ (as above), the strategy IsoAlg$\langle [m], \{s_i\}_{i\in M}\rangle$ is $O(\log^2 n \cdot \log m)$-approximately optimal. This clearly runs in time polynomial in n and m. As mentioned earlier, it is also easy to modify the description of algorithm IsoAlg so as to output a decision-tree representation of the resulting strategy (again in polynomial time). Hence we obtain:

Theorem 49. *There is an $O(\log^2 n \cdot \log m)$ approximation algorithm for* Metric Isolation*, where n is number of vertices and m is number of scenarios.*

4.3.3 Optimal Split Tree Problem

In the optimal split tree problem [99], we are given a set of m items with associated non-negative weights $\{p_i\}_{i=1}^m$ (that sum to 1) and a collection $\{T_j\}_{j=1}^n$ of n binary tests with non-negative costs $\{c_j\}_{j=1}^n$. Each test $T_j \subseteq [m]$ (for $j \in [n]$) is a subset of the items that correspond to passing the test; so performing test j distinguishes between items T_j and $[m] \setminus T_j$.

Definition 50. *A* split tree S *is a binary tree where each internal node is labeled by a test, and each leaf node is labeled by an item such that:*

- *For each item $i \in [m]$ define a path π_i in S from the root node to some leaf as follows. At any internal node, if i passes the test then π_i follows the right branch; if it fails the test then π_i follows the left branch.*
- *For each $i \in [m]$, the path π_i ends at a leaf labeled item i.*

The cost *L_i of an item $i \in [m]$ is the sum of test-costs along path π_i; and the cost of the split tree S is $\sum_{i=1}^{m} p_i \cdot L_i$.*

The objective in the optimal split tree problem is to compute a split tree of minimum cost. As mentioned earlier, $O(\log m)$-approximation algorithms are known for the optimal split tree problem in the following special cases: equal item-weights [99] or equal test-costs [2]. We show here that our algorithm for Metric Isolation implies an $O(\log m)$ approximation for the optimal split tree problem (under arbitrary item-weights and test-costs). As mentioned earlier, there is an $\Omega(\log m)$ hardness of approximation for this problem [35].

We first observe that the optimal split tree problem is a special case of Metric Isolation. Given an instance of optimal split tree (as above), consider a metric (V, d) induced by a weighted star with center r and n leaves corresponding to the tests. For each $j \in [n]$, we set $d(r, j) = \frac{c_j}{2}$. The demand scenarios are as follows: for each $i \in [m]$, scenario i consists of a unit demand at each of $\{j \in [n] \mid i \in T_j\}$ and zero demand at all other vertices. It is easy to see that this Metric Isolation instance corresponds exactly to the optimal split tree instance.

We now show an $O(\log m)$-approximation algorithm for Metric Isolation instances on weighted star metrics, which implies the same bound for the optimal split tree problem. This improvement comes from the following strengthening of Theorem 39.

Corollary 51. *There is a $(1 - \frac{1}{e})$-approximation algorithm for group Steiner orienteering on weighted star metrics.*

Proof: We assume (without loss of generality) that the root r is the star-center. Consider an instance of group Steiner orienteering on a weighted star-metric (V, d) with center r and leaves $[n]$, g groups $\{X_i \subseteq [n]\}_{i=1}^{g}$ with profits $\{v_i\}_{i=1}^{g}$, and length bound B. For each $j \in [n]$, define set $S_j := \{i \in [g] \mid j \in X_i\}$ of cost $c_j := \frac{d(r,j)}{2}$. The group Steiner orienteering instance is equivalent to computing a collection $K \subseteq [n]$ of the sets with $\sum_{j \in K} c_j \leq B/2$ that maximizes $f(K) := \sum\{v_i \mid i \in \cup_{j \in K} S_j\}$. Observe that the latter problem is an instance of maximizing a monotone submodular function (namely $f : 2^{[n]} \to \mathbb{R}$) over a knapsack constraint (i.e. $\sum_{j \in K} c_j \leq B/2$), for which a $1 - \frac{1}{e}$ approximation algorithm is known [147]. ■

Based on the framework in Section 4.3.2, the above constant approximation for GSO implies the following.

Theorem 52. *There is an $O(\log m)$-approximation algorithm for the optimal split tree problem, where m is the number of items.*

4.3.4 Issue of Observing Demands

In the stochastic vehicle routing problem as defined, the exact demand at a vertex is observed when the vehicle visits that vertex. In this section, we consider a variant where the demand at a vertex is not determined by merely visiting it, but only when it is actually *served*. This setting only applies to the split-delivery version. We refer to this variant of SVRP as $\mathsf{SVRP}_{\mathsf{obs}}$. It is clear that for any instance of the problem, the optimal value under SVRP is at most that under $\mathsf{SVRP}_{\mathsf{obs}}$ (since the latter is more constrained). The algorithm for SVRP does not directly extend to $\mathsf{SVRP}_{\mathsf{obs}}$ since Lemma 38 does not hold here (see remark after Lemma 38). However the following theorem shows that a suitable modification still permits a reduction from $\mathsf{SVRP}_{\mathsf{obs}}$ to Metric Isolation.

Theorem 53. *A ρ-approximation algorithm for* Metric Isolation *implies a $(2\rho + \frac{7}{2})$-approximation for $\mathsf{SVRP}_{\mathsf{obs}}$. Hence there is an $O(\log^2 n \cdot \log m)$-approximation algorithm for $\mathsf{SVRP}_{\mathsf{obs}}$.*

Proof: Consider any instance $\mathcal{I}$ of $\mathsf{SVRP}_{\mathsf{obs}}$ given by metric (V, d) with root $r \in V$, and m scenarios given by demand-vectors $\{\overline{q}^i\}_{i=1}^m$ and probabilities $\{p_i\}_{i=1}^m$. Let OPT denote the optimal value of $\mathcal{I}$ considered as an SVRP instance; as noted above, the optimal value of $\mathcal{I}$ under $\mathsf{SVRP}_{\mathsf{obs}}$ is at least OPT. As argued in Lemma 38, the optimal value of the Metric Isolation instance corresponding to $\mathcal{I}$ is at most OPT. The algorithm for $\mathsf{SVRP}_{\mathsf{obs}}$ is:

1. Obtain a ρ-approximately optimal strategy σ for the Metric Isolation instance of $\mathcal{I}$ (given by the algorithm in the assumption).
2. Return the following strategy for $\mathsf{SVRP}_{\mathsf{obs}}(\mathcal{I})$:
   (a) Visit vertices (using the capacity Q vehicle) according to strategy σ *while servicing* each vertex as it is visited (so the demands are determined even in the $\mathsf{SVRP}_{\mathsf{obs}}$ setting); whenever the vehicle runs out of items, it performs a *refill trip* to-and-from r.

(b) Let k denote the realized scenario, that is identified at the end of σ.

(c) Service all remaining demands in $\overline{q}^k$ using the $\frac{5}{2}$-approximate tour for CVRP [73].

For the analysis, we *condition* on a fixed scenario $j \in [m]$ (as in the proofs for SVRP with independent demands). Let τ_j denote the r-tour followed by σ under scenario j when demands are only observed (not serviced). Note that when the strategy for $\mathsf{SVRP}_{\mathsf{obs}}(\mathcal{I})$ executes step 2a, the resulting r-tour τ'_j is τ_j along with refill-trips from some vertices on τ_j (where the vehicle runs out of items). We refer to any vertex on τ_j from which a refill-trip is made as a *breakpoint*. Let U_j denote the set of all breakpoints on τ_j; we always include the root r in U_j. Then the length of τ'_j is exactly $d(\tau_j) + 2\sum_{w \in U_j} d(r,w)$. Using Claim 36 (Section 4.2) on tour τ_j, the demands served in it, and breakpoints U_j, we obtain:

$$2 \cdot \sum_{w \in U_j} d(r,w) \leq d(\tau_j) + \frac{2}{Q} \sum_{v \in V} q_v^j \cdot d(r,v), \qquad \forall j \in [m] \tag{4.9}$$

We now obtain that $d(\tau'_j) \leq 2 \cdot d(\tau_j) + \frac{2}{Q}\sum_{v \in V} q_v^j \cdot d(r,v)$ for each $j \in [m]$. In other words, the expected length of Step 2a is:

$$\sum_{j=1}^{m} p_j \cdot d(\tau'_j) \leq 2\sum_{j=1}^{m} p_j \cdot d(\tau_j) + \sum_{j=1}^{m} p_j \cdot \left(\frac{2}{Q} \sum_{v \in V} q_v^j \cdot d(r,v) \right) \leq (2\rho + 1) \cdot \mathsf{OPT}$$

where the last inequality uses the observations: (1) $\sum_{j=1}^{m} p_j \cdot d(\tau_j)$ equals $\mathsf{IsolateTime}(\sigma)$, which is at most $\rho \cdot \mathsf{OPT}$; and (2) $\sum_{j=1}^{m} p_j \cdot \left(\frac{2}{Q} \sum_{v \in V} q_v^j \cdot d(r,v) \right)$ is a lower bound on $\mathsf{SVRP}(\mathcal{I})$.

Finally as argued in Lemma 38, since we know the realized scenario in Step 2c, the expected length here is at most $\frac{5}{2} \cdot \mathsf{OPT}$. Thus we have the theorem. ∎

4.3.5 Hardness of Approximation

In this subsection, we provide lower bounds on the approximability of the metric isolation problem, and SVRP under explicit demands. We first observe that Metric Isolation is at least as hard to approximate as group Steiner tree. This is almost identical to the reduction [35] from Set-cover to the optimal split tree problem; we give a proof for completeness. Combined with the result in [77], the following theorem implies that Metric Isolation is $\Omega(\log^{2-\epsilon} n)$ hard to approximate.

Theorem 54. *An α-approximation algorithm for* Metric Isolation *implies an $\alpha + o(1)$ approximation algorithm for group Steiner tree.*

Proof: Consider an arbitrary instance of group Steiner tree on metric (V, d) with root r and groups $X_1, \cdots, X_g \subseteq V$; let OPT denote its optimal value. Assume without loss of generality that $X_i \neq X_j$ for all $i \neq j$. We construct an instance of Metric Isolation as follows. Let $V' = V \cup \{s\}$ where s is a new vertex, and define metric d' on V':

$$d'(u,v) := \begin{cases} d(u,v) & \text{for } u, v \in V \\ d(u,r) + L & \text{for } u \in V,\ v = s \end{cases}, \qquad \forall (u,v) \in \binom{V'}{2}$$

Above $L \gg \alpha n \cdot \max_{u,v} d(u,v)$ is some large value. There are $g+1$ scenarios in the Metric Isolation instance: $X_1, \cdots, X_g$ and $X_{g+1} := \{s\}$, with probabilities

$$p_i := \begin{cases} \frac{1}{gL} & \text{if } 1 \leq i \leq g \\ 1 - \frac{1}{L} & \text{if } i = g+1 \end{cases},$$

The root in the Metric Isolation instance remains r. Let OPT' denote the optimal isolation time of this instance. We will show that $(1 - o(1)) \cdot \mathsf{OPT} \leq \mathsf{OPT}' \leq \mathsf{OPT} + 1$ which would prove the proposition.

(A) $(1 - o(1)) \cdot \mathsf{OPT} \leq \mathsf{OPT}'$**.** Consider the optimal strategy for the Metric Isolation instance; let σ denote the r-tour traversed by this strategy under scenario X_{g+1}. We now argue that σ is a feasible solution to group Steiner tree. Observe that the optimal isolation time is at most $n \cdot \max_{u,v} d(u,v) \ll \frac{1}{\alpha} \cdot L$: visiting all vertices in V along any r-tour is a feasible strategy. Since $p_{g+1} = 1 - o(1)$, the r-tour σ does not visit vertex s (otherwise the objective value would be at least $L - 1$). Hence σ is an r-tour in metric (V, d) as well. Furthermore σ must visit at least one vertex from each $\{X_i\}_{i=1}^{g}$: otherwise scenario X_{g+1} is not isolated. Thus σ is a feasible solution to the group Steiner instance. Finally, $\mathsf{OPT}' \geq (1 - \frac{1}{L}) \cdot d(\sigma) \geq (1 - \frac{1}{L}) \cdot \mathsf{OPT}$.

(B) $\mathsf{OPT}' \leq \mathsf{OPT} + 1$**.** Let τ denote an optimal r-tour for the given GST instance, so $d(\tau) = \mathsf{OPT}$. Consider the following strategy for Metric Isolation:

1. Traverse r-tour τ to determine whether or not X_{g+1} is the realized scenario.
2. If X_{g+1} is realized, stop.
3. If X_{g+1} is not realized, visit all vertices in V along an arbitrary r-tour to determine the realized scenario.

For any $i \in [g+1]$, let π_i denote the r-tour traversed under scenario X_i in the above strategy. It is clear that $d(\pi_{g+1}) \leq d(\tau) \leq \mathsf{OPT}$, and for all $i \in [g]$, $d(\pi_i) \leq \frac{L}{\alpha}$. Thus the resulting isolation time is at most:

$$(1 - \frac{1}{L}) \cdot \mathsf{OPT} + g \cdot \frac{1}{gL} \cdot \frac{L}{\alpha} \leq \mathsf{OPT} + 1$$

Thus we have the desired reduction. ∎

In the latency group Steiner problem (LGST), we are given metric (V, d) with root r, groups $\{X_i \subseteq V\}_{i=1}^{g}$ of vertices, and the goal is to compute an r-tour covering all groups that minimizes the *average arrival time* at a group. As observed in Subsection 4.3.1, this is a special case of partial latency group Steiner. We now obtain the following hardness result for SVRP under explicit demands. Combined with the result in [77], this implies that SVRP is $\Omega(\log^{1-\epsilon} n)$ hard to approximate.

Theorem 55. *An α-approximation algorithm for SVRP under explicit demands implies an $O(\alpha)$-approximation algorithm for* LGST*, which in turn implies an $O(\alpha \cdot \log g)$-approximation algorithm for* GST.

Proof: We first give the reduction from LGST to SVRP with explicit demands, and then the reduction from group Steiner tree to LGST.

(I) Reducing LGST **to SVRP.** Consider an arbitrary instance $\mathcal{I}_0$ of LGST with metric (V, d), root r, and groups $\{X_i \subseteq V\}_{i=1}^{g}$; let OPT_0 denote its optimal value. By standard scaling arguments we can assume that every edge in the metric has length in the range $[1, n^3]$. For each $i \in [g]$, let Tsp_i denote the length of the minimum TSP tour on vertices X_i, and $L := \max_{i=1}^{g} \mathsf{Tsp}_i$. Also set $t := \lceil n \cdot L \rceil$; it is clear that $t \leq n^5$. We construct a new metric (U, l) as follows: take t disjoint copies of metric (V, d) and contract all copies of vertex r to a new root s. So we have:

$$l(u,v) := \begin{cases} d(u,v) & \text{for } u, v \text{ in the same copy of } V \\ d(u,r) + d(v,r) & \text{for } u, v \text{ in distinct copies of } V \end{cases}$$

For this reduction, it suffices to consider the stochastic TSP problem (which is SVRP with capacity $Q = \infty$); hence each scenario is just a subset of vertices. In fact we consider a further special case, where upon visiting any vertex v, it is not only known whether or not there is demand at v, but if there is demand at v it is also known what the realized scenario is. (Eg. this can be ensured by making copies of

every vertex v (at zero distance from each other) where each copy of v corresponds to exactly one of the scenarios that occur at v.)

We now define the stochastic TSP instance $\mathcal{I}$ on metric (U, l) with root s. There are $m := t \cdot g$ scenarios (each with equal probability), for each $j \in [t]$ and $i \in [g]$, the X_i-vertices in the j-th copy of (V, d) constitute a scenario. This completes the description of the stochastic TSP instance $\mathcal{I}$, which is essentially t copies of $\mathcal{I}_0$. Let OPT denote the optimal stochastic TSP objective value of $\mathcal{I}$, and OPT_I the optimal Metric Isolation objective value of $\mathcal{I}$. Note that by construction, the minimum TSP length for each scenario is at most L; so by Lemma 38, $\mathsf{OPT}_I \leq \mathsf{OPT} \leq \mathsf{OPT}_I + L$. Note also that $\mathsf{OPT}_I \geq \Omega(t) \gg L$. Hence we have,

$$\mathsf{OPT}_I \leq \mathsf{OPT} \leq (1 + o(1)) \cdot \mathsf{OPT}_I \tag{4.10}$$

This is main purpose of considering the scaled up instance $\mathcal{I}$. The above relation between the SVRP and Metric Isolation objectives does not hold in arbitrary instances such as $\mathcal{I}_0$. We now establish a sequence of claims that imply $\mathsf{OPT} = \Theta(t) \cdot \mathsf{OPT}_0$, which suffices to prove the theorem.

Claim 56. *The optimal* LGST *value corresponding to $\mathcal{I}$ is $(1 + o(1)) \cdot \mathsf{OPT}_I$.*

Proof: Observe that for instance $\mathcal{I}$, the Metric Isolation objective is precisely Min-LPGST where weights are $\frac{1}{m}$ each and target $h = m-1$. Furthermore since $\mathcal{I}$ consists of t copies of the original instance $\mathcal{I}_0$, the optimal LGST value corresponding to $\mathcal{I}$ (i.e. Min-LPGST with target m) is $(1 + o(1))$ times the Metric Isolation value. ∎

Claim 57. *The optimal* LGST *value corresponding to $\mathcal{I}$ is $O(t) \cdot \mathsf{OPT}_0$.*

Proof: Fix an optimal solution ζ to $\mathcal{I}_0$; and for each integer $i \geq 1$, let N_i denote the fraction of groups having arrival times between 2^{i-1} and 2^i. So $\mathsf{OPT}_0 \geq \sum_{i \geq 1} N_i \cdot 2^{i-1}$. We construct a feasible solution to $\mathcal{I}$ as follows. For each integer $i \geq 1$ do: for every copy of (V, d) in (U, l), starting from s traverse the 2^i-length prefix of ζ and return to s. It is easy to see that the latency of the resulting solution to LGST on $\mathcal{I}$ is at most $\sum_{i \geq 1} N_i \cdot (t \cdot 2^{i+1}) = O(t) \cdot \mathsf{OPT}_0$. ∎

Claim 58. *The optimal* LGST *value corresponding to $\mathcal{I}$ is $\Omega(t) \cdot \mathsf{OPT}_0$.*

Proof: Fix an optimal solution τ to LGST on $\mathcal{I}$; note that $\mathcal{I}$ has a total weight of one distributed uniformly over all t copies $\mathcal{I}_0$. For each integer $i \geq 0$, let L_i be the

length of the shortest prefix of τ that contains weight $1 - \frac{1}{2^i}$ and let τ_i denote this prefix. The latency of τ is:

$$\mathsf{Lat}(\tau) \geq \sum_{i \geq 1} (L_i - L_{i-1}) \cdot \frac{1}{2^i} = \sum_{i \geq 1} L_i \cdot \left(\frac{1}{2^i} - \frac{1}{2^{i+1}} \right) = \frac{1}{2} \sum_{i \geq 1} L_i \cdot \frac{1}{2^i}$$

We first claim that for every integer $i \geq 1$, there is an r-tour σ_i in metric (V, d) of length at most $\frac{2}{t} \cdot L_i$ that covers at least $1 - \frac{1}{2^{i-1}}$ fraction of $\mathcal{I}_0$-groups. For each $j \in [t]$, let $f_j \in [0, 1]$ denote the fraction of groups in the j-th copy of $\mathcal{I}_0$ that are *not covered* by τ_i. By the definition of τ_i, we have $\frac{1}{t} \sum_{j=1}^{t} f_j \leq \frac{1}{2^i}$. Thus (by Markov inequality) there is a subset $S \subseteq [t]$ of $|S| \geq \frac{t}{2}$ copies with $f_j \leq \frac{2}{2^i}$ for all $j \in S$. Now there is some $k \in S$ such that the portion of τ_i in the k-th copy of $\mathcal{I}_0$ has length at most $\frac{L_i}{|S|} \leq 2 \cdot \frac{L_i}{t}$. Thus the portion of τ_i in this k-th copy corresponds to an r-tour σ_i in metric (V, d) of length $d(\sigma_i) \leq 4 \cdot \frac{L_i}{t}$ that covers at least $1 - \frac{1}{2^{i-1}}$ fraction of $\mathcal{I}_0$-groups.

Consider the solution to $\mathcal{I}_0$ which is the concatenation $\sigma_1 \cdot \sigma_2 \cdots$. The latency of this solution is at most:

$$d(\sigma_1) + \sum_{i \geq 2} d(\sigma_i) \cdot \frac{1}{2^{i-2}} \leq 4 \cdot \frac{L_1}{t} + \frac{16}{t} \cdot \sum_{i \geq 2} L_i \cdot \frac{1}{2^i} \leq \frac{32}{t} \cdot \mathsf{Lat}(\tau)$$

Thus we have the claim. ∎

Combining Claims 56, 57 and 58 we obtain $\mathsf{OPT}_I = \Theta(t) \cdot \mathsf{OPT}_0$. Together with Equation (4.10), this implies $\mathsf{OPT} = \Theta(t) \cdot \mathsf{OPT}_0$, proving the first part of the theorem.

(II) Reducing GST **to** LGST**.** This is a simple set-covering based reduction. Recall that any instance $\mathcal{I}$ of the group Steiner tree problem consists of metric (V, d) with root $r \in V$ and groups $\{X_i \subseteq V\}_{i=1}^{g}$. The goal is to compute a minimum length r-tour covering all groups. Let $\mathcal{A}$ be any ρ-approximation algorithm for LGST; we now describe how this may be used to obtain an $O(\rho \cdot \log g)$-approximation for GST.

1. Set current r-tour $S \leftarrow \emptyset$, uncovered groups $U \leftarrow [g]$.
2. While $(U \neq \emptyset)$ do:
   (a) Run algorithm $\mathcal{A}$ on the LGST instance with metric (V, d), root r and groups $\{X_i\}_{i \in U}$. Let τ be the r-tour obtained, and Lat the average arrival time of groups in U.

(b) Let τ' be the truncated tour obtained by returning to root r after traversing length $2 \cdot \mathsf{Lat}$ along τ.

(c) Augment $S \leftarrow S \cdot \tau'$.

(d) Remove from U all groups covered in τ'.

3. Output S as solution to GST.

Let OPTdenote the optimal value of the given instance $\mathcal{I}$. It is clear that upon termination, the final solution S is a feasible GST solution. It only remains to bound the length of S.

Claim 59. *The solution S obtained above has length $d(S) \leq O(\rho \cdot \log g) \cdot \mathsf{OPT}$.*

Proof: Consider an arbitrary iteration where $U \subseteq [g]$ denotes the set of uncovered groups. Observe that the optimal value of LGST instance restricted to U is at most OPT: the arrival time of *every* group in an optimal GST solution to $\mathcal{I}$ is at most OPT. Thus the average arrival time in τ is $\mathsf{Lat} \leq \rho \cdot \mathsf{OPT}$. So the increase in the length of S in any iteration is at most $4\rho \cdot \mathsf{OPT}$. Also, by Markov inequality, at least $|U|/2$ groups have arrival time at most $2 \cdot \mathsf{Lat}$ in tour τ; i.e. τ' covers at least $|U|/2$ groups from U. Hence the number of uncovered groups (i.e. $|U|$) decreases by a factor of two in each iteration; this implies that the number of iterations is at most $\lceil \log_2 g \rceil + 1$. The claim now follows. ∎

Thus we have an $O(\rho \cdot \log g)$-approximation algorithm for GST assuming a ρ-approximation for LGST. This proves the second part of the theorem. ∎

4.4 SVRP under Black-box Distribution

In this section, we study the most general model for SVRP where the demand distribution $\mathcal{D}$ is given by means of a black-box (or an oracle) from which an algorithm can draw samples. In this setting, every sample from this black-box is an independent draw from the actual demand distribution $\mathcal{D}$. We provide a strong information-theoretic lower bound on the approximability of SVRP under black-box distributions. In fact this lower bound holds even in the special case of stochastic TSP where the capacity is ∞, so each scenario is just a subset of vertices to visit. We show that any algorithm that samples from the distribution $\mathcal{D}$ at most n^c times (for any $c = o(n/\log n)$) achieves only an $\Omega(n/c)$ approximation ratio; recall that

n denotes the number of vertices in the underlying metric. In particular, there is no $o(n)$-approximation algorithm for the black-box model of SVRP, that makes polynomially many samples.

Let $\mathcal{A}$ be any algorithm for SVRP under black-box distributions that samples from the distribution at most $N := n^c$ times (for some $c = o(\frac{n}{\log n})$). We make *no* assumption on the running time of $\mathcal{A}$. Given any instance of SVRP, $\mathcal{A}$ outputs a decision tree corresponding to an adaptive strategy of visiting vertices (Definition 37); note that the size of this decision tree may be exponentially large, since we make no assumption on the running time of $\mathcal{A}$. We will show that there exist instances for which the solution produced by $\mathcal{A}$ has cost $\Omega(\frac{n}{c})$ times the optimal.

Consider a weighted-star metric (V, d) with center r and $2t$ leaves $A := \{a_i\}_{i=1}^t$ and $B := \{b_i\}_{i=1}^t$. The edges (r, a_i) (for all $i \in [t]$) have length zero, and edges (r, b_i) (for all $i \in [t]$) have length one. The number of vertices in the metric is $n = 2t + 1$. Fix a parameter $k := 2c$, and let $C := \binom{[t]}{k}$ be the collection of all k-subsets of $[t]$. For any $M \in C$, we define $A(M) = \{a_j \mid j \in M\}$ and similarly $B(M) = \{b_l \mid l \in M\}$. Let $\mathcal{F}$ denote the set of all functions $f : C \to C$. For every function $f \in \mathcal{F}$, define demand-distribution $\mathcal{D}_f$ as containing the following scenarios uniformly at random (each scenario in $\mathcal{D}_f$ is just a subset of vertices to be visited, specified by a k-subset of $[t]$):

$$\mathcal{D}_f \equiv \{A(S) \cup B(f(S)), \text{ for all } S \in C\}$$

For each function $f \in \mathcal{F}$, define instance $\mathcal{I}_f$ of SVRP on metric (V, d) with demand distribution $\mathcal{D}_f$, and infinite vehicle capacity. We will consider the outcome of algorithm $\mathcal{A}$ on these SVRP instances, and show that it performs poorly on at least one of them.

Claim 60. *The optimal value of each instance $\mathcal{I}_f$ is at most $2k$.*

Proof: This strategy first visits all vertices in A (at zero length) to observe demands at $A(S)$ for some $S \in C$, and then visits precisely the vertices in $B(f(S))$. The resulting tour always has length $2k$ (since $|f(S)| = k$). ∎

Note that this strategy relies crucially on the fact that it knows the exact function f. On the other hand, when algorithm $\mathcal{A}$ is run on some instance $\mathcal{I}_f$, the only information about function f that it obtains is through a set of $N \ll \binom{t}{k}$ random samples from the black-box $\mathcal{D}_f$. In the following we show that this information is insufficient to compute a good solution. Below, u.a.r. stands for uniformly at random.

By the definition of functions f that we consider, each sample from the black-box is of the form $A(A') \cup B(B')$ where $A', B' \in C$. Note that for a given set of samples from the black-box distribution, algorithm $\mathcal{A}$ outputs a unique decision tree. Conditioned on a set $Q = \{A_i \cup B_i\}_{i=1}^N$ of samples from the black-box, let $T(Q)$ denote the decision-tree output by $\mathcal{A}$. For any set $Q = \{A_i \cup B_i\}_{i=1}^N$ of demand-samples, any function $f \in \mathcal{F}$ that agrees with Q (i.e. $f(A_i) = B_i$ for all $1 \leq i \leq N$), and any scenario $S \in C$, let $L(f, S, Q)$ denote the number of B-vertices visited under decision-tree $T(Q)$ when the realized demand is $A(S) \cup B(f(S))$.

Claim 61. *For any set $Q = \{A_i \cup B_i\}_{i=1}^N$ of demand-samples and $S \in C \setminus \{A_i\}_{i=1}^N$, $E_f[L(f, S, Q)] = \Omega(t)$ where the expectation is taken over u.a.r. function $f \in \mathcal{F}$ that agrees with Q.*

Proof: Note that conditioned on Q, algorithm $\mathcal{A}$ outputs the unique decision-tree $T(Q)$. We claim that for any $S' \in C$, when $T(Q)$ is run with demands $A(S) \cup B(S')$, it visits all the vertices $B(S')$. This follows since for any $S' \in C$ there is some function $f \in \mathcal{F}$ that agrees with Q *and* has $f(S) = S'$; and $T(Q)$ must be a feasible solution to this instance $\mathcal{I}_f$. Hence $T(Q)$ visits all vertices in $A(S) \cup B(S')$ under this scenario in $\mathcal{D}_f$. Let $\tilde{L}(S', S, Q)$ be the number of B-vertices visited when $T(Q)$ is run with demands $A(S) \cup B(S')$. Observe that for every $f \in F$ that agrees with Q and has $f(S) = S'$, we have $L(f, S, Q) = \tilde{L}(S', S, Q)$. When $f \in \mathcal{F}$ u.a.r. over functions that agree with Q, it is clear that $E_f[L(f, S, Q)] = E_{S'}[\tilde{L}(S', S, Q)]$ (where $S' \in C$ u.a.r.).

We assume (without loss of generality) that decision-tree $T(Q)$ first visits all vertices in A, since this has zero length. We focus on the decision tree $T(Q, S)$ defined as $T(Q)$ *conditioned* on observing demands at $A(S)$; note that $T(Q, S)$ visits only B-vertices. As observed above, for any $S' \in C$, when $T(Q, S)$ is run with demands $B(S')$ it must visit all the vertices $B(S')$, and $\tilde{L}(S', S, Q)$ is the number of B-vertices it visits. Let $M(S', S, Q) \leq \tilde{L}(S', S, Q)$ denote the number of B-vertices visited by $T(Q, S)$ under demands $B(S')$ until the point it visits all of $B(S')$. We now show a lower bound $E_{S'}[M(S', S, Q)] \geq \Omega(t)$ that proves the claim.

Consider running decision tree $T(Q, S)$ under u.a.r. $S' \in C$ representing demands on B-vertices. Let us condition on traversing some path O in $T(Q, S)$ from the root that visits at most $\frac{t}{2}$ B-vertices, such that some vertex v is the next to be visited (after O). The outcomes along path O do *not* include vertex v, which is the first unexplored vertex to be visited after O. We have $Pr_{S'}[v \in S' \mid S' \text{ consistent with } O] \leq \frac{2k}{t}$, since irrespective of the exact outcomes along O there are at least $\frac{t}{2}$ unexplored B-vertices and at most k are in S'. In the traversal of $T(Q, S)$ under u.a.r. $S' \in C$,

whenever a vertex $v \in S'$ is visited, we term it *success*. Note that $E_{S'}[M(S', S, Q)]$ is precisely the expected number of vertex-visits until k successes. Consider a truncated process $\mathcal{P}$ that traverses $T(Q, S)$ under u.a.r. $S' \in C$, and stops at the earlier of k successes or $\frac{t}{2}$ vertex-visits. The expected number of vertices visited in $\mathcal{P}$ is clearly at most $E_{S'}[M(S', S, Q)]$. Note that at any point in $\mathcal{P}$, the conditional probability of success is at most $\frac{2k}{t}$ (from above). Now define random process $\mathcal{P}'$ that performs independent tosses of a coin with heads probability $\frac{2k}{t}$: $\mathcal{P}'$ keeps tossing the coin until either the number of tosses exceeds $\frac{t}{2}$ *or* at least k heads are observed. Clearly, the expected number of vertices visited in $\mathcal{P}$ is at least the expected number of tosses in $\mathcal{P}'$. Finally, it is easy to see that the expected number of tosses in $\mathcal{P}'$ is $\Omega(t)$. ∎

The next claim is straightforward from the definitions.

Claim 62. *For any function $f \in \mathcal{F}$, the expected cost of the decision-tree produced by algorithm $\mathcal{A}$ is $E_{A_1,\cdots,A_N} E_S[L(f, S, \{A_i, f(A_i)\}_{i=1}^N)]$ where the expectation is taken over $A_1, \cdots, A_N, S \in C$ chosen independently and u.a.r.*

Theorem 63. *For any algorithm $\mathcal{A}$ that makes at most $N = n^c$ samples from the black-box distribution (where $c = o(n/\log n)$), there is a function $f \in \mathcal{F}$ such that algorithm $\mathcal{A}$ achieves only an $\Omega(\frac{n}{c})$ approximation on SVRP instance $\mathcal{I}_f$.*

Proof: For a fixed demand-sample $Q = \{A_i, B_i\}_{i=1}^N$, let $\mathcal{F}(Q) := \{f \in \mathcal{F} \mid f(A_i) = B_i,\ \forall i \in [N]\}$ denote all functions in $\mathcal{F}$ that agree with Q. Consider the following two ways of generating random tuples of the form $\langle f, Q = \{A_i, B_i\}_{i=1}^N \rangle$, where $f \in \mathcal{F}$ and $A_1, \cdots, A_N, B_1, \cdots, B_N \in C$.

D1 Pick $A_1, \cdots, A_N, B_1, \cdots, B_N \in C$ u.a.r. independently to set $Q = \{A_i, B_i\}_{i=1}^N$; then pick $f \in \mathcal{F}(Q)$ independently u.a.r.

D2 Pick u.a.r. $f \in \mathcal{F}$; then pick $A_1, \cdots, A_N \in C$ independently u.a.r., and set $B_i = f(A_i)$ for all $i \in [N]$ (again $Q = \{A_i, B_i\}_{i=1}^N$).

It is easy to see that the distributions of tuples $\langle f, Q \rangle$ resulting from these two procedures is identical (since $\mathcal{F}$ includes all functions mapping C to C).

For any fixed $Q = \{A_i, B_i\}_{i=1}^N$, taking expectation over $S \in C$ chosen u.a.r. and independently in Claim 61 implies that $E_S E_{f \in \mathcal{F}(Q)}[L(f, S, Q)] = \Omega(t)$, since $Pr_S[S \notin \{A_1, \cdots, A_N\}] \geq 1 - \frac{N}{|C|} \geq \frac{1}{2}$. Since S and f are independent, we have

$E_{f\in\mathcal{F}(Q)}E_S[L(f,S,Q)] = E_S E_{f\in\mathcal{F}(Q)}[L(f,S,Q)] = \Omega(t)$. Now taking an outer expectation over $Q = \{A_i, B_i\}_{i=1}^N$,

$$E_{A_1,\cdots,A_N,B_1,\cdots,B_N} E_{f\in\mathcal{F}(Q)} E_S[L(f,S,Q)] = \Omega(t)$$

Taking an outer expectation over $f \in \mathcal{F}$ u.a.r. in Claim 62, the average solution cost produced by $\mathcal{A}$ on instances $\mathcal{I}_f$ is:

$$E_f E_{A_1,\cdots,A_N} E_S[L(f,S,\{A_i, f(A_i)\}_{i=1}^N)] = E_Q E_{f\in\mathcal{F}(Q)} E_S[L(f,S,Q)] = \Omega(t)$$

where the first equality uses the equivalence of the two distributions (D1) and (D2) mentioned above. Since the average solution cost on instances corresponding to $\mathcal{F}$ is $\Omega(t)$, there is some $f \in \mathcal{F}$ with solution cost $\Omega(t) = \Omega(n)$.

Furthermore, Claim 60 implies that the optimal value of every $\mathcal{I}_f$ is at most $2k = 4c$. Thus we obtain the theorem. ■

Credits: The results in this chapter are based on joint work with Anupam Gupta, Ravishankar Krishnaswamy and R. Ravi.

Chapter 5

VRPs on Asymmetric Metrics

5.1 Introduction

In this chapter, we consider some vehicle routing problems on *asymmetric metrics*. An asymmetric metric is given by tuple (V, d) where V denotes the vertex-set and $d : V \times V \to \mathbb{R}_+$ is a distance function that satisfies the triangle inequality, i.e. $d(u, v) + d(v, w) \geq d(u, w)$ for all $u, v, w \in V$. We emphasize that the distances need not satisfy symmetry, as in the metrics considered in previous chapters; hence for any $u, v \in V$, $d(u, v) \neq d(v, u)$ in general.

The most basic VRP in this setting is the *Asymmetric Traveling Salesman Problem* (ATSP). The best known approximation ratio for ATSP is $O(\log n)$ [60] (as opposed to $\frac{3}{2}$ [34] in the symmetric version). Improving this bound is an important open question in approximation algorithms. The problems considered in this chapter are variants of ATSP that are also directed counterparts of well-studied VRPs on symmetric metrics. Network design problems on directed graphs are often much harder to approximate than their undirected counterparts– the traveling salesman and Steiner tree problems are well known examples. The currently best known approximation ratio for the *directed Steiner tree* problem is $O(n^\epsilon)$ [27] for any fixed $\epsilon > 0$, whereas there is a 1.55-approximation algorithm on symmetric metrics [130].

5.1.1 Problem Definition and Preliminaries

We now define the directed VRPs considered in this chapter. All these problems are defined over an asymmetric metric (V, d) with $|V| = n$ vertices and a specified root $r \in V$. Any tour (resp. path) starting at r is called an r-tour (resp. r-path).

Directed k-TSP. Given a target $k \leq n$, the goal is to compute a minimum length r-tour that contains at least k other vertices. This is a generalization of the ATSP, which is obtained by setting $k = n$.

Minimum Ratio ATSP. This involves finding an r-tour that minimizes the ratio of the length of the tour to the number of vertices in it (not including the root r). Observe that if the requirement that the tour contain the root is dropped, this ratio problem becomes the *minimum mean weight cycle problem*, which is exactly solvable in polynomial time [93]. However, the rooted version which we are interested in is NP-complete.

Directed Orienteering. Here we are given a length bound D, and the goal is to find an r-path of length at most D, that visits the maximum number of vertices. The orienteering problem can also be extended to the setting where there is some non-negative profit at each vertex, and the goal is to maximize total profit.

Directed Minimum Latency. For a directed path (or tour) π and vertices $u, v \in V$, let $d^{\pi}(u, v)$ denote the distance from u to v along π; if v is not reachable from u along π, then $d^{\pi}(u, v) = \infty$. The directed minimum latency problem involves computing a spanning r-path π that minimizes $\sum_{v \in V} d^{\pi}(r, v)$; the quantity $d^{\pi}(r, v)$ is the *latency* of vertex v in path π. Another possible definition of this problem requires an *r-tour* covering all vertices, where the latency of the root r is set to be the distance required to return to r (i.e. the total tour length); note that in the previous definition, the latency of r is zero. As we show later in this chapter, the approximability of both these versions of directed latency are within a constant factor of each other. We work with the *path version* of directed latency.

Bicriteria approximations. Consider any problem $\mathcal{P}$ having the form:

$$\min\{C(z) : z \in \mathcal{S}, N(z) \geq k\}$$

where $\mathcal{S}$ is the feasible region, $C : \mathcal{S} \to \mathbb{R}_+$ is the cost function, $N : \mathcal{S} \to \mathbb{R}_+$ is a

coverage function, and k is the target. Eg., in the directed k-TSP problem, $\mathcal{S}$ is the set of all r-tours; for any $z \in \mathcal{S}$, $C(z)$ is the length of tour z, and $N(z)$ is the number of vertices covered in tour z. An algorithm $\mathcal{A}$ for problem $\mathcal{P}$ is said to be an (α, β) *bi-criteria approximation*, if on each problem instance $\mathcal{A}$ obtains a solution $y \in \mathcal{S}$ satisfying $C(y) \leq \alpha \cdot \mathsf{OPT}$ and $N(y) \geq \frac{k}{\beta}$, where $\mathsf{OPT} = \min\{C(z) : z \in \mathcal{S}, N(z) \geq k\}$ is the optimal value of this instance.

For a directed multigraph $G = (V, E)$ and any $S \subseteq V$, we denote by $\delta^+(S) = \{(u, v) \in E \mid u \in S,\ v \notin S\}$ the edges leaving vertex-set S, and $\delta^-(S) = \{(u, v) \in E \mid u \notin S,\ v \in S\}$ the edges entering S. When dealing with asymmetric metrics with vertex set V, the edge set E is assumed to be $V \times V$, unless mentioned otherwise. For any vector $x : E \to \mathbb{R}_+$ and $F \subseteq E$, denote $x(F) := \sum_{e \in F} x_e$.

Multigraph $G = (V, E)$ is called *Eulerian* if in-degree equals out-degree at each vertex, i.e. $|\delta^+(v)| = |\delta^-(v)|$ for each $v \in V$. Given $G = (V, E)$ and vertices $u, v \in V$, the *directed connectivity* from u to v equals the maximum number of edge-disjoint paths from u to v (or equivalently, the minimum number of edges whose removal disconnects v from u). Given a directed multigraph $G = (V, E)$ and edges $(u, v),\ (v, w) \in E$, the operation of replacing edges (u, v) and (v, w) by a new edge (u, w) (i.e. $E \leftarrow (E \setminus \{(u, v), (v, w)\}) \cup \{(u, w)\}$) is called *splitting-off*. Some proofs in this chapter make use of the following 'splitting-off' theorems for digraphs.

Theorem 64 (Mader [107]). *Let $G = (U \cup \{v\}, E)$ be a directed multigraph such that the indegree equals outdegree at v, and the directed connectivity between any pair of vertices in U is at least k. Then for every edge $(v, w) \in E$ there exists an edge $(u, v) \in E$ so that after splitting-off edges (u, v) and (v, w), the directed connectivity between every pair of vertices in U remains at least k.*

Theorem 65 (Frank [57] (Theorem 4.3) and Jackson [90]). *Let $G = (U \cup \{v\}, E)$ be an Eulerian directed multigraph. For each edge $(v, w) \in E$ there exists an edge $(u, v) \in E$ so that after splitting-off edges (u, v) and (v, w), the directed connectivity between every pair of vertices in U remains the same.*

Remark: Consider any vector $x : E \to \mathbb{R}_+$ of rational edge-capacities that is Eulerian, namely $x(\delta^-(v)) = x(\delta^+(v))$ at all vertices $v \in V$. Let $M \in \mathbb{N}$ be large enough that $M \cdot x_e \in \mathbb{N}$ for all $e \in E$. Then $M \cdot x$ corresponds to an Eulerian multi-graph, for which the above splitting-off theorems apply. Based on this correspondence, we use splitting-off theorems directly on rational edge-capacities that are Eulerian.

5.1.2 Results

We present a polynomial time $O(\log^2 n \cdot \log k)$-approximation algorithm for the directed k-TSP problem. This is based on an $O(\log^2 n)$-approximation algorithm for minimum ratio ATSP. To the best of our knowledge, this ratio problem has not been studied earlier. The main ingredient in the algorithm is a splitting-off theorem on directed Eulerian graphs. This algorithm appears in Section 5.2. We then use the approximation algorithm for minimum ratio ATSP, to obtain an $O(\log^2 n)$ approximation guarantee for directed orienteering (Section 5.3). This answers in the affirmative, the question of poly-logarithmic approximability of directed orienteering [17].

For the directed latency problem, we present an LP-based reduction to the *Asymmetric Traveling Salesman Path* problem (ATSP-path) [101, 32]. We give an $n^{O(1/\epsilon)}$ time algorithm for the directed latency problem that achieves an approximation ratio of $O(\rho \cdot \frac{n^\epsilon}{\epsilon^3})$ (for any $\frac{1}{\log n} < \epsilon < 1$), where ρ is the integrality gap of an LP relaxation for the ATSP-path problem (details in Section 5.4). We obtain an upper bound $\rho = O(\sqrt{n})$ in Section 5.5; however we conjecture that $\rho = O(\log n)$. In particular, our result implies a polynomial time $O(n^{1/2+\epsilon})$-approximation algorithm for directed latency (with any fixed $\epsilon > 0$), which is the first non-trivial approximation guarantee for this problem.

5.1.3 Related Work

The best known approximation guarantee for ATSP is $O(\log n)$; the first approximation ratio [60] was $\lceil \log_2 n \rceil$, and since then several papers improved the constant term, leading to the currently best $\frac{2}{3} \cdot \log_2 n$ bound [53].

All problems considered in this chapter are well-studied in the undirected setting. Blum et al. [17] obtained the first constant factor approximation algorithm for the orienteering problem (on symmetric metrics). This was improved to a factor of 3 in Bansal et al. [13], and later to $2 + \epsilon$ (for any fixed $\epsilon > 0$) in Chekuri et al. [30]. Bansal et al. [13] also used orienteering as a subroutine to also obtain poly-logarithmic approximation algorithms for some generalizations of orienteering: *deadline TSP* and *vehicle routing problem with time windows*. The first constant-factor approximation algorithm for undirected minimum latency was given by Blum et al. [18]; following a line of improvements, the currently best known approximation ratio is 3.59 due to Chaudhuri et al. [29].

Chekuri and Pal [31] obtained a general approximation algorithm for a class of VRPs on asymmetric metrics, that runs in *quasi-polynomial time*. In particular, their result implies a quasi-polynomial time $O(\log n)$-approximation algorithm for the directed orienteering problem. Their result also holds for the generalization when the profit is any monotone submodular set function on the vertices.

The directed latency problem turns out to be closely related to *Asymmetric Traveling Salesman Path* (ATSP-path) [101]. ATSP-path is a generalization of ATSP, where the goal is to compute the minimum length spanning path between specified end-points. Lam and Newmann [101] were the first to consider this problem, and they gave an $O(\sqrt{n})$ approximation based on the Frieze et al. [60] algorithm for ATSP. This was improved to an $O(\log n)$ ratio in Chekuri and Pal [32], that extended the Kleinberg and Williamson [98] algorithm for ATSP. Subsequently Feige and Singh [53] showed that the approximability of ATSP and ATSP-path are within a constant factor of each other.

5.2 Directed k-TSP

In this section, we consider the directed k-TSP problem and obtain an $O(\log^2 n \cdot \log k)$-approximation algorithm for this problem. We first obtain an $O(\log^2 n)$-approximation algorithm for the related minimum ratio ATSP (Theorem 69), and then show how this implies the result for directed k-TSP (Theorem 70). Our algorithm for minimum ratio ATSP is based on upper bounding the integrality gap of a suitable LP relaxation for ATSP (Theorem 66), which we study next.

5.2.1 A linear relaxation for ATSP

Consider the following LP relaxation for ATSP on metric (V, d).

$$
\begin{array}{lcl}
 & \min \ \sum_e d_e \cdot z_e & \\
s.t. & & \\
 & z(\delta^+(v)) = z(\delta^-(v)) & \forall v \in V \\
(ALP) & z(\delta^+(S)) \geq 1 & \forall \emptyset \neq S \neq V \\
 & z_e \geq 0 & \forall \text{ edges e}
\end{array}
$$

This relaxation was also studied in Vempala and Yannakakis [151], where the authors proved a structural property about basic solutions to (ALP). We are not

aware of any previous result bounding the integrality gap of (ALP). However, the following stronger LP relaxation (with additional degree equals 1 constraints) was shown to have an integrality gap of at most $\lceil \log n \rceil$ in Williamson [153].

$$
\begin{array}{rcl}
 & \min \ \sum_e d_e \cdot z_e & \\
s.t. & & \\
 & z(\delta^+(v)) = 1 & \forall v \in V \\
(ALP') & z(\delta^-(v)) = 1 & \forall v \in V \\
 & z(\delta^+(S)) \geq 1 & \forall \emptyset \neq S \neq V \\
 & z_e \geq 0 & \forall \text{ edges e}
\end{array}
$$

We give an independent proof using the directed splitting-off Theorem 64 that gives a $\lceil \log n \rceil$ upper bound on the integrality gap of (ALP) (Theorem 66). Then we observe that for any asymmetric metric (V, d), the optimal values of (ALP) and (ALP') coincide (Theorem 68).

Theorem 66. *The integrality gap of* (ALP) *is at most* $\lceil \log n \rceil$.

Proof: This proof has the same outline as the proof for the stronger LP relaxation (ALP') in Williamson [153]. We use the $\lceil \log n \rceil$ approximation algorithm for ATSP due to Frieze et al. [60], which works by computing cycle covers repeatedly (in at most $\lceil \log n \rceil$ iterations). In this algorithm, if $U \subseteq V$ is the set of representative vertices in some iteration, the cost incurred in this iteration equals the minimum cycle cover on U. Let $ALP(U)$ denote the LP relaxation ALP restricted to a subset U of the original vertices (and edges induced on U), and $\mathsf{OPT}(ALP(U))$ its optimal value. Then we have:

Claim 67. *For any subset* $U \subseteq V$*, the minimum cycle cover on* U *has cost at most* $\mathsf{OPT}(ALP(U))$.

Proof: Consider the following linear relaxation for cycle cover.

$$
\begin{array}{rcl}
 & \min \ \sum_e d_e \cdot x_e & \\
s.t. & & \\
 & x(\delta^+(v)) - x(\delta^-(v)) = 0 & \forall v \in U \\
(CLP) & x(\delta^+(v)) \geq 1 & \forall v \in U \\
 & x_e \geq 0 & \forall \text{ edges e}
\end{array}
$$

These constraints are equivalent to a circulation problem on network N which contains two vertices v_{in} and v_{out} for each vertex $v \in U$. The edges in N are:

$\{(u_{out}, v_{in}) : \forall\ u, v \in U,\ u \neq v\}$, and $\{(v_{in}, v_{out}) : \forall v \in U\}$. The cost of each (u_{out}, v_{in}) edge is $d(u, v)$, and each (v_{in}, v_{out}) edge costs 0. It is easy to see that the minimum cost circulation on N that places at least one unit of flow on each edge in $\{(v_{in}, v_{out}) : \forall v \in U\}$ is exactly the optimal solution to (CLP). But the linear program for minimum cost circulation is integral (network matrices are totally unimodular, c.f. [117]), and so is (CLP).

Any integral solution to (CLP) defines an Eulerian subgraph H with each vertex in U having degree at least 1. Each connected component C of H is Eulerian and can be shortcut to get a cycle on the vertices of C. Since triangle inequality holds, the cost of each such cycle is at most that of the original component. So this gives a cycle cover of U of cost at most $\mathsf{OPT}(CLP(U))$, the optimal value of (CLP). But the linear program $ALP(U)$ is more constrained than $CLP(U)$; so the minimum cycle cover on U costs at most $\mathsf{OPT}(ALP(U))$. ∎

We now establish the *monotonicity property* of ALP, namely:

$$\mathsf{OPT}(ALP(U)) \leq \mathsf{OPT}(ALP(V)) \qquad \forall U \subseteq V$$

Consider any subset $U \subseteq V$, vertex $v \in U$, and $U' = U - v$; we will show that $\mathsf{OPT}(ALP(U')) \leq \mathsf{OPT}(ALP(U))$. Let z be any fractional solution to $ALP(U)$. Applying splitting-off (Theorem 64) repeatedly on vertex $v \in U$ (until its degree is zero), we obtain another fractional solution z' to $ALP(U)$ where $z'(\delta^+(v)) = z'(\delta^-(v)) = 0$. Furthermore, due to the triangle inequality, $d \cdot z' \leq d \cdot z$. Now observe that z' is a feasible solution to $ALP(U')$ (since it is induced on U'); hence $\mathsf{OPT}(ALP(U')) \leq \mathsf{OPT}(ALP(U))$, and using this inductively we have monotonicity for ALP.

This suffices to prove the theorem, as the cost incurred in each iteration of the Frieze et al. [60] algorithm can be bounded by $\mathsf{OPT}(ALP(V))$, and there are at most $\lceil \log n \rceil$ iterations. ∎

We note that using splitting-off, one can prove a stronger statement than Theorem 66, which relates the optimal values of (ALP) and (ALP'). It was shown in [153] that the optimal value of (ALP') equals the Held-Karp lower bound [84]; so the next theorem shows that for any ATSP instance, the values of the Help-Karp bound, (ALP') and (ALP) are all equal. A similar result for the symmetric case was proved in Goemans and Bertsimas [67], which was also based on splitting-off (for undirected graphs).

Theorem 68. *The optimal values of (ALP) and (ALP') are equal.*

Proof: Clearly the optimal value of (ALP') is at most that of (ALP). We will show that any fractional solution z to (ALP) can be modified to a fractional solution z' to (ALP'), such that $\sum_e d_e \cdot z'_e \leq \sum_e d_e \cdot z_e$, which would prove the theorem. Let $L \in \mathbb{N}$ be large enough so that $L \cdot z$ is integral, and let H denote a directed multigraph with $L \cdot z_{u,v}$ edges from u to v, for all $u, v \in V$. From the feasibility of z in (ALP), we know that H is Eulerian and has directed connectivity at least L between every pair of vertices.

If some $v \in V$ has degree strictly greater than L, we reduce its degree by one as follows. Let v' be any vertex in $V \setminus v$, and $\mathcal{P}_{v,v'}$ denote a minimal set of edges that constitutes exactly L edge-disjoint paths from v to v'. Due to minimality, the number of edges in $\mathcal{P}_{v,v'}$ incident to v is exactly L and they are all edges leaving v. Since the degree of v is at least $L+1$, there is an edge $(v,w) \in H \setminus \mathcal{P}_{v,v'}$. Applying Theorem 64 to edge (v,w), we obtain edge $(u,v) \in H \setminus \mathcal{P}_{v,v'}$ such that the directed connectivity between vertices of $V \setminus v$ in $H' = (H \setminus \{(u,v),(v,w)\}) \cup (u,w)$ remains at least L. Furthermore, by the choice of (v,w), $\mathcal{P}_{v,v'} \subseteq H'$; so the directed connectivity from v to v' in H' is at least L. Since H' is Eulerian, it now follows that the directed connectivity between all vertex-pairs in H' is also at least L. Thus we obtain a multigraph H' from H which maintains connectivity and decreases the degree of vertex v by 1. Repeating this procedure for all vertices in V having degree greater than L, we obtain (an Eulerian) multigraph G having directed connectivity L (between all pairs of vertices) such that the degree of each vertex equals L.

Note that in the degree reducing procedure above, the only operation we used was splitting-off. Since d satisfies triangle inequality, the total cost of edges in G (under length d) is at most that of H. Finally, scaling down G by L, we obtain the claimed fractional solution z' to (ALP'). ∎

5.2.2 Minimum ratio ATSP

We now describe the $O(\log^2 n)$-approximation algorithm for minimum ratio ATSP, which uses Theorem 66 and the stronger splitting-off Theorem 65 for Eulerian digraphs.

Theorem 69. *There is a polynomial time $O(\log^2 n)$-approximation algorithm for the minimum ratio ATSP problem.*

Proof: The approximation algorithm for minimum ratio ATSP is based on the

following LP relaxation for this problem.

$$
\begin{array}{rcl}
 & \min \ \sum_e d_e \cdot x_e & \\
s.t. & & \\
 & x(\delta^+(v)) = x(\delta^-(v)) & \forall v \in V \\
 & x(\delta^+(S)) \geq y_v & \forall S \subseteq V - \{r\} \ \forall v \in S \\
(RLP) & \sum_{v \neq r} y_v \geq 1 & \\
 & x_e \geq 0 & \forall \text{ edges e} \\
 & 0 \leq y_v \leq 1 & \forall v \in V - \{r\}
\end{array}
$$

To see that this is indeed a relaxation, consider the optimal integral r-tour C^* that covers l vertices (excluding r). We construct a solution to (RLP) by setting $y_v = \frac{1}{l}$ for all vertices $v \in C^*$, and $x_e = \frac{1}{l}$ for all edges $e \in C^*$. It is easy to see that this solution is feasible and has cost $\frac{d(C^*)}{l}$ which is the optimal ratio. The linear program (RLP) can be solved in polynomial time using the Ellipsoid algorithm. The algorithm is as follows:

1. Let (x, y) denote an optimal solution to (RLP).
2. Discard all vertices $v \in V \setminus r$ with $y_v \leq \frac{1}{2n}$; all remaining vertices have y-values in the interval $[\frac{1}{2n}, 1]$.
3. Define $g = \lceil \log_2 n + 1 \rceil$ groups of vertices where group G_i (for $i = 1, \cdots, g$) consists of all vertices v having $y_v \in (\frac{1}{2^i}, \frac{1}{2^{i-1}}]$.
4. Run the Frieze et al. [60] algorithm on each of $G_i \cup \{r\}$ and output the r-tour with the smallest ratio.

Note that the total y-value of vertices remaining after step 2 is at least $1/2$. For any edge capacities $z : V \times V \to \mathbb{R}_+$ and vertices $u, v \in V$, let $\lambda(u, v; z)$ denote the directed connectivity from u to v. Consider any group G_i, and edge-capacities given by $x'_i := 2^i \cdot x_i$. It is clear (from feasibility of x in RLP) that x'_i is Eulerian, and for all $v \in G_i$, $\lambda(r, v; x'_i) = \lambda(v, r; x'_i) \geq 2^i \cdot y_v \geq 1$. Now we split-off vertices in $V \setminus (G_i \cup \{r\})$ one by one, using Theorem 65, which preserves the directed connectivity between vertices of $G_i \cup \{r\}$. This results in (Eulerian) edge-capacities z^i induced on vertices $G_i \cup r$ satisfying $\lambda(r, v; z^i), \lambda(v, r; z^i) \geq 1$ for all $v \in G_i$. Hence z^i is a feasible fractional solution to $ALP(G_i \cup \{r\})$. Furthermore, by triangle inequality, $d \cdot z^i \leq d \cdot x'_i = 2^i(d \cdot x)$. Now Theorem 66 implies that there exists an r-tour on G_i of cost at most $\beta = \lceil \log n \rceil$ times $d \cdot z^i$. In fact, the Frieze et al. [60]

algorithm applied on $G_i + r$ produces such a tour. We now claim that one of the r-tours found in step 4 (over all $i = 1, \cdots g$) has a small ratio:

$$\min_{i=1}^{g} \frac{\beta(d \cdot z^i)}{|G_i|} \leq \min_{i=1}^{g} \frac{2^i \beta(d \cdot x)}{|G_i|} \leq \frac{\beta \sum_{i=1}^{g} d \cdot x}{\sum_{i=1}^{g} |G_i|/2^i} \leq 4g\beta \cdot (d \cdot x)$$

The last inequality follows from the fact that there is a total y-weight of at least $1/2$ after step 2, so $\frac{1}{2} \leq \sum_{v \neq r} y_v \leq \sum_{i=1}^{g} \frac{1}{2^{i-1}} |G_i| = 2 \sum_{i=1}^{g} \frac{|G_i|}{2^i}$. Thus we have a $4g\beta = O(\log^2 n)$ approximation algorithm for minimum ratio ATSP. ∎

5.2.3 Algorithm for Directed k-TSP

We now describe how minimum ratio ATSP can be used to obtain an approximation algorithm for the directed k-TSP problem.

Theorem 70. *There is a polynomial time $O(\log^2 n \cdot \log k)$ approximation algorithm for the directed k-TSP problem.*

Proof: We use the $\alpha = O(\log^2 n)$-approximation algorithm for the related minimum ratio ATSP problem. Let OPT denote the optimal value of the directed k-TSP instance. By performing binary search, we may assume that we know the value of OPT within a factor 2. We only consider vertices $v \in V$ satisfying $d(r, v), d(v, r) \leq$ OPT; this does not affect the optimal solution. Then we invoke the minimum ratio ATSP algorithm repeatedly (each time restricted to the currently uncovered vertices) until the total number of covered vertices $t \geq \frac{k}{2}$. Note that for every instance of the ratio problem that we solve, there is a feasible solution of ratio $\leq \frac{2 \cdot \mathsf{OPT}}{k}$ (namely, the optimal k-TSP tour covering at least $k/2$ residual vertices). Thus we obtain an r-tour on $t \geq \frac{k}{2}$ vertices having ratio $\leq \frac{2\alpha \cdot \mathsf{OPT}}{k}$; so the length of this r-tour is at most $\frac{2\alpha t \cdot \mathsf{OPT}}{k}$. Now, we split this r-tour into $l = \lceil \frac{2t}{k} \rceil$ di-paths, each containing at least $\frac{t}{l} \geq \frac{k}{4}$ vertices (this can be done in a greedy fashion). By averaging, the minimum length di-path in this collection has length at most $\frac{2\alpha t \mathsf{OPT}/k}{l} \leq \alpha \cdot \mathsf{OPT}$. Joining the first and last vertices in this di-path to r, we obtain an r-tour containing at least $\frac{k}{4}$ vertices, of length at most $(\alpha + 2) \cdot \mathsf{OPT}$. So we get an $(O(\alpha), 4)$ bi-criteria approximation for directed k-TSP. This algorithm can now be used as follows. Until at least k vertices are covered, do:

- Let k' denote the number of vertices covered so far; run the bi-criteria approximation algorithm with a target of $k - k'$, restricted to currently uncovered vertices.

A standard set cover based analysis implies that this is an $O(\alpha \cdot \log k)$-approximation algorithm for directed k-TSP. ■

5.3 Directed Orienteering

In this section, we consider the orienteering problem in asymmetric metrics. We consider a slightly more general version than the definition in Section 5.1. In the directed orienteering problem, we are given metric (V, d), length bound D, *origin* $s \in V$ and *destination* $t \in V$; the goal is to compute an $s - t$ directed path that has length at most D and maximizes the number of vertices visited. A closely related problem is *directed k-path*: given metric (V, d), origin s, destination t, and a target k, find an s-t directed path of minimum length that visits at least k other vertices. Note that directed k-path reduces to directed k-TSP when $s = t$.

The algorithm for directed orienteering is based on the following sequence of reductions: directed orienteering to 'directed minimum excess' (Theorem 73), directed minimum excess to directed k-path (Theorem 72), and directed k-path to minimum ratio ATSP (Theorem 71). The first two reductions above are identical to the corresponding reductions for undirected orienteering in Blum et al. [17] and Bansal et al. [13]. We prove the following bi-criteria approximation guarantee for directed k-path.

Theorem 71. *A ρ-approximation algorithm for minimum ratio ATSP implies a $(3, 4\rho)$ bi-criteria approximation algorithm for the directed k-path problem.*

Proof: We assume (by performing a binary search) that we know the optimal value OPT of the directed k-path instance within a constant factor, and let G denote the directed graph corresponding to metric (V, d) (which has an edge of length $d(u, v)$ from u to v for every pair of vertices $u, v \in V$). We modify graph G to obtain graph H as follows: **(a)** discard all vertices v such that $d(s, v) > \mathsf{OPT}$ or $d(v, t) > \mathsf{OPT}$; and **(b)** add an extra edge from t to s of length OPT. In the rest of this proof, we refer to the shortest path metric induced by H as (V, l). Note that each tour in metric l corresponds to a tour in graph H (using shortest paths in H for each metric edge); below, any tour in metric l will refer to the corresponding tour in graph H. Since there is an s-t path of length OPT (in metric d) covering k vertices, appending the (t, s) edge, we have an s-tour σ^* of length at most $2 \cdot \mathsf{OPT}$ (in metric l) covering $k + 1$ vertices.

Now, we run the minimum ratio ATSP algorithm with root s in metric l repeatedly until either **(1)** $\frac{k}{2}$ vertices are covered and the extra (t,s) edge is never used in the current tour (in graph H); or **(2)** the extra (t,s) edge is used for the first time in the current tour (in H). Let σ be the s-tour obtained (in graph H) at the end of this iteration, and h the number of vertices covered. Note that each s-tour added in a single call to minimum ratio ATSP, may use the extra (t,s) edge at most once (by an averaging argument). So in case (1), the (t,s) edge is absent in σ, and in case (2), the (t,s) edge is used exactly once and it is the last edge in σ. Note also that during each call of minimum ratio ATSP, there is a feasible solution of ratio $\frac{2\mathsf{OPT}}{k}$ (σ^* restricted to the remaining vertices); so the ratio of the s-tour σ, $\frac{l(\sigma)}{h} \leq \rho \cdot \frac{2\mathsf{OPT}}{k}$. From σ we now obtain a feasible s-t path τ in metric d as follows. In case (1), add a direct (s,t) edge: $\tau = \sigma \cdot (s,t)$; in case (2), remove the only copy of the extra (t,s) edge (occurring at the end of σ): $\tau = \sigma \setminus \{(t,s)\}$. In either case, s-t path τ contains h vertices and has length $d(\tau) \leq \frac{2\rho h}{k}\mathsf{OPT} + \mathsf{OPT}$. Note that in case (1), $h \geq \frac{k}{2}$; and in case (2), since the extra (t,s) edge is used, $\frac{\mathsf{OPT}}{h} \leq \frac{l(\sigma)}{h} \leq 2\rho\frac{\mathsf{OPT}}{k}$, so $h \geq \frac{k}{2\rho}$. Hence in either case, τ contains $h \geq \frac{k}{2\rho}$ vertices and $d(\tau) \leq \frac{4\rho h}{k}\mathsf{OPT}$. We now greedily split τ into maximal paths, each of which has length at most OPT; the number of subpaths obtained is at most $\frac{d(\tau)}{\mathsf{OPT}} \leq \frac{4\rho h}{k}$. So one of these paths contains at least $h/(\frac{4\rho h}{k}) = \frac{k}{4\rho}$ vertices. Adding direct edges from s to the first vertex on this path and from the last vertex on this path to t, we obtain an s-t path of length at most $3 \cdot \mathsf{OPT}$ containing at least $\frac{k}{4\rho}$ vertices. ∎

As in Blum et al. [17], we define the *excess* of an s-t di-path as the difference of the path length and the shortest path distance from s to t. The *directed minimum excess* problem is defined as follows: given an asymmetric metric (V,d), origin (s) and destination (t) vertices, and a target k, find an s-t di-path of minimum excess that visits at least k other vertices. The next two theorems together reduce the directed orienteering problem to the directed k-path problem, for which we just obtained an approximation algorithm.

Theorem 72 (Blum et al. [17])**.** *An (α,β) bi-criteria approximation algorithm for the directed k-path problem implies a $(2\alpha-1,\beta)$ bi-criteria approximation algorithm for the directed minimum excess problem.*

Theorem 73 (Bansal et al. [13])**.** *An (α,β) bi-criteria approximation algorithm for the directed minimum excess problem implies an $\lceil\alpha\rceil \cdot \beta$ approximation algorithm for the directed orienteering problem.*

The proofs of Theorems 72 and 73 are identical to the corresponding proofs in the undirected setting, and are not repeated here. The only difference from the

undirected case is that we consider bi-criteria guarantees for the directed k-path and minimum excess problems. We now obtain a result that relates the directed orienteering problem and minimum ratio ATSP.

Corollary 74. *A ρ-approximation algorithm for the minimum ratio ATSP problem implies an $O(\rho)$-approximation algorithm for the directed orienteering problem. Conversely, a ρ-approximation algorithm for directed orienteering implies an $O(\rho)$-approximation algorithm for minimum ratio ATSP.*

Proof: The first direction follows directly from Theorems 71,72 and 73. For the other direction, we are given a ρ-approximation algorithm for directed orienteering. Let D denote the length of some minimum ratio tour σ^*, t the last vertex visited by σ^* (before returning to the root r), and h the number of vertices it covers; so the optimal ratio is $\frac{D}{h}$. The algorithm for minimum ratio ATSP first guesses a value D' such that $D' \leq D \leq 2 \cdot D'$, and the last vertex t. Note that we can guess powers of two for the value of D', which gives $O(\log_2(n \cdot d_{max}))$ possibilities for D' (where d_{max} is the length of the longest edge). Also, the number of possibilities for t is at most n; so the algorithm only makes a polynomial number of guesses. The algorithm then runs the directed orienteering algorithm with r and t as the start/end vertices and a length bound of $2D' - d(t,r) \geq D - d(t,r)$. Note that removing the last (t,r) edge from σ^* gives a feasible solution to this orienteering instance that covers h vertices. Hence the ρ-approximation algorithm is guaranteed to find an r-t di-path covering at least $\frac{h}{\rho}$ vertices, having length at most $2D' - d(t,r)$. Now, adding the (t,r) edge to this path gives an r-tour of ratio at most $2D'/(\frac{h}{\rho}) \leq 2\rho\frac{D}{h}$. ∎

Corollary 74 and Theorem 69 imply an $O(\log^2 n)$-approximation algorithm for directed orienteering. Furthermore, any improvement in minimum ratio ATSP implies a corresponding improvement for directed orienteering.

5.4 Directed Minimum Latency

In this section, we present our algorithm for the directed latency problem. First we establish the equivalence (up to a constant factor) of the 'tour' and 'path' versions of the latency problem. Recall that *Path latency* involves computing a spanning r-*path* π that minimizes $\sum_{v \in V} d^{\pi}(r,v)$ (here $d^{\pi}(r,r) = 0$); and *Tour latency* involves finding a spanning r-*tour* τ that minimizes $\sum_{v \in V} d^{\tau}(r,v)$ (here $d^{\tau}(r,r) = d(\tau)$).

Theorem 75. *The approximability of the path and tour versions of directed latency are within a factor* 4 *of each other.*

Proof: Reducing Tour-latency to Path-latency. We first show that any approximation algorithm for path-latency implies the same guarantee for tour-latency. Modify the given metric (V, d) (for tour-latency) by adding a new vertex r' with distances to vertices V as follows: $d(v, r') = d(v, r)$ and $d(r', v) = \infty$ for all $v \in V$. It is clear that any spanning r-path in the modified metric corresponds to a spanning r-tour in the original metric followed by edge (r, r') (which has length 0). Hence there is a correspondence between solutions to tour-latency and path-latency, that maintains the objective value.

Reducing Path-latency to Tour-latency. Let $\mathcal{A}$ be any α-approximation algorithm for tour-latency (we assume $\alpha \leq n$). Set $\beta = \lceil 4\alpha \rceil + 1$ and define a new metric $(\tilde{V}, \tilde{d})$ which is obtained from (V, d) by making β copies of each vertex in $V \setminus r$ (for each $v \in V \setminus r$, all its copies in $\tilde{V}$ are at distance 0 from each other). Let $\tilde{\mathsf{pLat}}$ (resp. pLat) denote the optimal value of path-latency on $\tilde{d}$ (resp. d); note that $\tilde{\mathsf{pLat}} = \beta \cdot \mathsf{pLat}$. We assume (by scaling) that the smallest non-zero distance in d is 1, and that there is a finite length spanning r-path in metric d (otherwise the latency problem is trivial); so $\mathsf{pLat} \leq n^2 d_{max}$ where d_{max} is the maximum finite distance in d. We further modify metric $\tilde{d}$ by adding *dummy edges* of length L (exact value to be set later) from each vertex in $\tilde{V} \setminus r$ to r, and let l denote the shortest path metric on this modified graph. Note that the optimal value of tour-latency on metric l is $\mathsf{tLat} \leq 2 \cdot \tilde{\mathsf{pLat}} + L$. The algorithm for path-latency on d is as follows.

- For each $0 \leq i \leq \lg(\beta n^2 d_{max})$ do:
  1. Set $L \leftarrow 2^i$ and l to be the metric as defined above.
  2. Run algorithm $\mathcal{A}$ for tour-latency on l to get tour τ. Let π denote the portion of τ until the first usage of a dummy edge.
  3. If π visits at least one copy of each vertex in V, consider this as a feasible solution to path-latency on d; otherwise skip this iteration.
- Output the best path-latency solution encountered in Step 3 above.

Note that since $1 \leq \tilde{\mathsf{pLat}} \leq \beta n^2 d_{max}$, there is an iteration where $\tilde{\mathsf{pLat}} \leq L = 2^i \leq 2 \cdot \tilde{\mathsf{pLat}}$. We now argue that in this iteration i, the above algorithm gets a feasible solution in Step 3 with path-latency value (on metric d) at most $4\alpha \cdot \mathsf{pLat}$.

The tour τ found by algorithm $\mathcal{A}$ has latency (in metric l) $\mathsf{Lat}(\tau) \leq \alpha \cdot \mathsf{tLat} \leq \alpha(2 \cdot \mathsf{pL\tilde{a}t} + L) \leq 4\alpha \cdot \mathsf{pL\tilde{a}t}$. If x denotes the number of unvisited vertices of $\tilde{V}$ in π (i.e. when a dummy edge is used for the first time in τ) then $\mathsf{Lat}(\tau) \geq L \cdot x$; hence $x \leq \mathsf{Lat}(\tau)/L \leq 4\alpha \cdot \mathsf{pL\tilde{a}t}/L \leq 4\alpha < \beta$. Since $\tilde{V}$ has β copies of each vertex of V, π visits at least one copy of each V-vertex: so Step 3 records π as a potential solution. We modify τ so that it visits all copies of each V-vertex the first time some copy is visited: this only reduces the latency of τ. After this modification, note that dummy edges are used in τ only after *all* vertices $\tilde{V}$ are visited: in other words, τ induces a path-latency solution on metric $(\tilde{V}, \tilde{d})$ of latency value $\mathsf{Lat}(\tau) \leq 4\alpha \cdot \mathsf{pL\tilde{a}t}$, that does not use any dummy edge. Hence the corresponding solution π in metric (V, d) has latency $\mathsf{Lat}(\pi) = \frac{1}{\beta}\mathsf{Lat}(\tau) \leq \frac{1}{\beta}4\alpha \cdot \mathsf{pL\tilde{a}t} = 4\alpha \cdot \mathsf{pLat}$. Thus the best solution to path-latency on d obtained over all iterations has value at most $4\alpha \cdot \mathsf{pLat}$, and we have a 4α approximation algorithm. ∎

The next proposition shows that the directed latency problem is at least as hard to approximate as the Asymmetric Traveling Salesman Problem.

Proposition 76. *An α-approximation for directed latency (path version) implies a 4α-approximation for ATSP. This reduction also holds in the special case of metrics induced by unweighted digraphs.*

Proof: We only present the reduction for unweighted metrics (the general case is identical). Let $\mathcal{A}$ be an α-approximation algorithm for the (path version) directed latency on unweighted metrics, and $G = (V, E)$ any n-vertex digraph that induces a shortest-path metric. We show how $\mathcal{A}$ can be used to obtain a 4α approximation for ATSP on G. Pick some root $r \in V$ and modify G by adding to it a directed path of n new vertices U originating at r: let H be the graph so obtained. A candidate solution for latency on H first visits the vertices V along the optimal ATSP tour on G and then visits U along the new path: this implies that the optimal latency on H, $\mathsf{Lat}^*(H) \leq 2n(\mathsf{Tsp}^*(G) + n)$ where $\mathsf{Tsp}^*(G)$ is the optimal value of ATSP on G. Note that $\mathsf{Tsp}^* \geq n$ (G has n vertices), and so $\mathsf{Lat}^*(H) \leq 4n\mathsf{Tsp}^*(G)$. Note also that any spanning r-path in H must first visit all the vertices V in some r-tour (that lies in graph G) and then the vertices U along the new path. Since there are n vertices (those in U) that appear after the spanning tour on G, we have $\mathsf{Lat}^*(H) \geq n \cdot \mathsf{Tsp}^*(G)$. Thus we have $\frac{1}{4n}\mathsf{Lat}^*(H) \leq \mathsf{Tsp}^*(G) \leq \frac{1}{n}\mathsf{Lat}^*(H)$, which implies the proposition. ∎

Before presenting the algorithm for directed latency, we define and describe an LP relaxation for the Asymmetric Traveling Salesman Path (ATSP-path) problem.

This LP is used in the algorithm for directed latency.

ATSP-path. Here we are given a directed metric (V, d) and specified start and end vertices $s, t \in V$. The goal is to compute the minimum length $s-t$ path that visits all the vertices. As mentioned earlier, it is known that the approximability of ATSP-path and ATSP are equal (up to a constant factor) [53]. We are concerned with the following LP relaxation of the ATSP-path problem.

$$
\begin{array}{rll}
 & \min \ \sum_e d_e \cdot x_e & \\
s.t. & & \\
 & x(\delta^+(u)) = x(\delta^-(u)) & \forall u \in V - \{s,t\} \\
 & x(\delta^+(s)) = x(\delta^-(t)) = 1 & \\
(ATSP - path) & x(\delta^-(s)) = x(\delta^+(t)) = 0 & \\
 & x(\delta^-(S)) \geq \frac{2}{3} & \forall \{u\} \subseteq S \subseteq V \setminus \{s\}, \quad \forall u \in V \\
 & x_e \geq 0 & \forall \text{ arcs e}
\end{array}
$$

The most natural LP relaxation for ATSP-path would have a one in the right-hand-side of the cut constraints, instead of $\frac{2}{3}$ as above. The above LP further relaxes the cut-constraints, and is still a valid relaxation of the problem. The precise value in the right-hand-side of the cut constraints is not important for our application: we only require it to be some constant strictly between $\frac{1}{2}$ and 1. Note that if the right-hand-side of the cut constraints is $\frac{1}{2}$ then the LP has infinite integrality gap, as shown by an instance consisting of two vertex disjoint paths from s to t. In the rest of this section, we let ρ denote the integrality gap of this LP. We study this LP further in Section 5.5, where we obtain an upper bound $\rho = O(\sqrt{n})$.

5.4.1 Algorithm for directed latency

We assume (by scaling) that the smallest non-zero distance in the given metric (V, d) is one, and let d_{max} denote the largest finite distance. The algorithm for directed latency is also given a parameter $\Omega(\frac{1}{\log n}) \leq \epsilon < 1$: it produces an $O(\rho \cdot \frac{n^\epsilon}{\epsilon^3})$ approximation in $n^{O(1/\epsilon)}$ time.

For the given instance of directed latency on metric (V, d) with root r, let π denote an optimal latency path, $L = d(\pi)$ its length, and OPT its total latency. Note that $1 \leq L \leq n \cdot d_{max}$. For any two vertices $u, v \in V$, recall that $d^\pi(u, v)$ denotes the length along path π from u to v; note that $d^\pi(u, v)$ is finite only if u appears before v on path π. The algorithm first guesses the length L (within factor 2) and the

following $l = \lceil \frac{1}{\epsilon} \rceil$ vertices on the optimal path: for each $i = 1, \cdots, l$, v_i is the last vertex on π with $d^{\pi}(r, v_i) \leq n^{i\epsilon}\frac{L}{n}$. We set $v_0 = r$ and note that v_l is the last vertex visited by π. Let $F := \{v_0, v_1, \cdots, v_l\}$ denote these *breakpoints*. Consider the linear program (MLP) in Figure 5.1

$$\min \sum_{i=0}^{l-1} n^{(i+1)\epsilon} \cdot \frac{L}{n} \cdot \left(\sum_{u \notin F} y_u^i \right) \tag{5.1}$$

$$z^i\left(\delta^+(u)\right) = z^i\left(\delta^-(u)\right) \qquad \forall u \in V \setminus \{v_i, v_{i+1}\}, \forall i = 0, \cdots, l-1 \tag{5.2}$$

$$z^i\left(\delta^+(v_i)\right) = z^i\left(\delta^-(v_{i+1})\right) = 1 \qquad \forall i = 0, \cdots, l-1 \tag{5.3}$$

$$z^i\left(\delta^-(v_i)\right) = z^i\left(\delta^+(v_{i+1})\right) = 0 \qquad \forall i = 0, \cdots, l-1 \tag{5.4}$$

$$z^i\left(\delta^-(S)\right) \geq y_u^i \qquad \forall \{u\} \subseteq S \subseteq V \setminus \{v_i\}, \quad \forall u \in V \setminus F, \quad \forall i = 0, \cdots, l-1 \tag{5.5}$$

$$\sum_{i=0}^{l-1} y_u^i \geq 1 \qquad \forall u \in V \setminus F \tag{5.6}$$

$$\sum_e d_e \cdot z^i(e) \leq n^{(i+1)\epsilon} \cdot \frac{L}{n} \qquad \forall i = 0, \cdots, l-1 \tag{5.7}$$

$$z^i(e) \geq 0 \qquad \forall \text{ edges } e, \quad \forall i = 0, \cdots, l-1 \tag{5.8}$$

$$y_u^i \geq 0 \qquad \forall u \in V \setminus F, \quad \forall i = 0, \cdots, l-1 \tag{5.9}$$

Figure 5.1: Linear program (MLP) for directed latency.

The linear program (MLP) requires a unit flow z^i to be sent from v_i to v_{i+1} (for all $0 \leq i \leq l-1$) such that the total extent to which each vertex u is covered (over all these flows) is at least 1. The constraints (5.2)-(5.4) ensure that z^i represents a fractional path from v_i to v_{i+1}. The constraint (5.5) measures the extent y_u^i to which vertex u is covered in flow z^i, and constraint (5.6) ensures that each vertex is covered to extent at least one. In addition, constraint (5.7) enforces the length of the i-th flow (under the length function d) to be at most $n^{(i+1)\epsilon} \cdot \frac{L}{n}$. It is easy to see that this LP can be solved in polynomial time for any guess $\{v_i\}_{i=1}^{l}$. Furthermore the number of possible guesses are: $\log_2(n \cdot d_{max})$ for the length L, and n^l for the breakpoints F. Hence we can obtain the optimal solution of (MLP) over all guesses, in $n^{O(1/\epsilon)}$ time.

Claim 77. *The minimum value of (MLP) over all possible guesses of $\{v_i\}_{i=0}^{l}$ is at most $2n^{\epsilon} \cdot \mathsf{OPT}$.*

Proof: This claim is straightforward, based on the correct guesses from an optimal path. Recall that π is an optimal latency path for the given instance. One of the guesses of the vertices $\{v_i\}_{i=0}^{l}$ satisfies the following condition: each v_i (for $i = 1, \cdots, l$) is the last vertex on π with $d^\pi(s, v_i) \leq n^{i\epsilon}\frac{L}{n}$. For each $i = 0, \cdots, l-1$, define O_i to be the set of vertices that are visited between v_i and v_{i+1} in path π. Let z^i denote the (integral) edge values corresponding to path π restricted to the vertices $O_i \cup \{v_i, v_{i+1}\}$; note that the cost of this flow $d \cdot z^i \leq d^\pi(r, v_{i+1}) \leq n^{(i+1)\epsilon}\frac{L}{n}$. Also set $y^i_u = 1$ for $u \in O_i$ and 0 otherwise, for all $i = 0, \cdots, l-1$. Note that each vertex in $V \setminus \{v_i\}_{i=0}^{l}$ appears in some set O_i, and each z^i supports unit flow from v_i to all vertices in O_i; hence this integral solution $\{z^i, y^i\}_{i=0}^{l-1}$ is feasible for (MLP). The cost of this solution is $\sum_{i=0}^{l-1} n^{(i+1)\epsilon}\frac{L}{n} \cdot |O_i| \leq n^\epsilon L + n^\epsilon \sum_{i=1}^{l-1} n^{i\epsilon}\frac{L}{n} \cdot |O_i| \leq 2n^\epsilon \cdot \mathsf{OPT}$, since $|O_0| \leq n$, $L \leq \mathsf{OPT}$, and each vertex $u \in O_i$ (for $i = 1, \cdots, l-1$) has $d^\pi(r, u) > n^{i\epsilon}\frac{L}{n}$. ■

We now assume that we have an optimal fractional solution $\{z^i, y^i\}_{i=0}^{l-1}$ to (MLP) over all guesses (with objective value as in Claim 77), and show how to round it to obtain $v_i - v_{i+1}$ paths for each $i = 0, \cdots, l-1$, which when concatenated give rise to *one* r-path having a small latency objective. We say that a vertex u is *well-covered* by flow z^i if $y^i_u \geq \frac{1}{4l}$. We partition the vertices $V \setminus F$ into two parts: V_1 consists of those vertices that are well-covered for *at least two* values of $i \in \{0, 1, \cdots, l\}$, and V_2 consists of all other vertices (i.e. well-covered for only one $i \in \{0, 1, \cdots, l\}$. Note that each vertex in V_2 is in fact covered by some flow z^i to the extent at least $\frac{3}{4}$. See also Figure7.1.

In the next two subsections, we give algorithms to service each of V_1 and V_2 separately using local paths, and then in Section 5.4.4 we show to stitch all the local paths into a single spanning r-path.

5.4.2 Servicing vertices V_1

We partition V_1 into l parts as follows: U_i (for $i = 0, \cdots, l-1$) consists of those vertices of V_1 that are well-covered by z^i but *not* well-covered by any flow z^j for $j > i$. Each set U_i is serviced separately by means of a suitable ATSP solution on $U_i \cup \{v_i\}$ (see Lemma 79): this step requires a bound on the length of back-arcs from U_i-vertices to v_i, which is ensured by the next claim.

Claim 78. *For each vertex $w \in U_i$, $d(w, v_i) \leq 8l \cdot n^{i\epsilon}\frac{L}{n}$.*

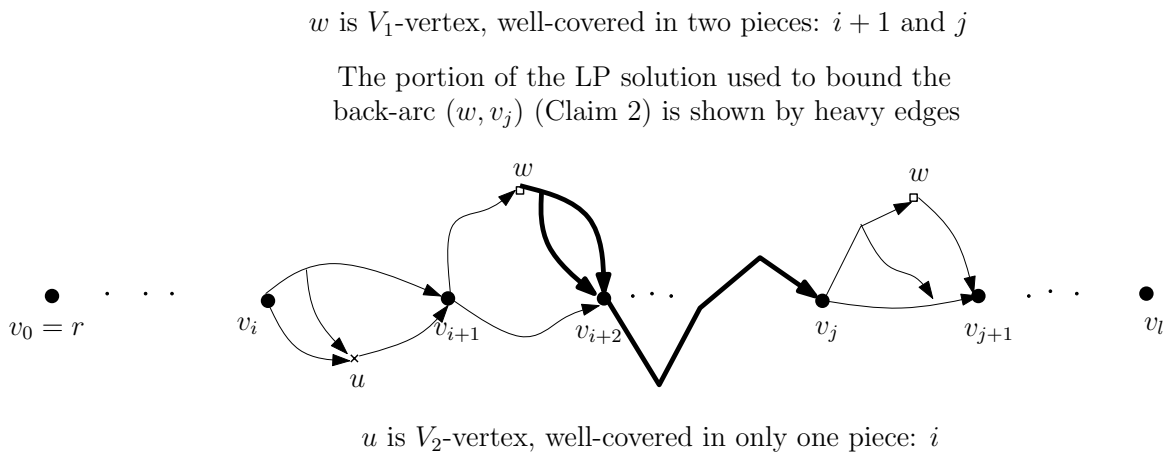


Figure 5.2: Guessed breakpoints $v_0, \cdots, v_l$ and flows sent by the LP between consecutive breakpoints.

Proof: Let $j \leq i-1$ be such that $y^j_w \geq \frac{1}{4l}$; such an index exists by the definition of V_1 and U_i. In other words, edge-capacities z^j support at least $\frac{1}{4l}$ flow from w to v_{j+1}; so $4l \cdot z^j$ supports a unit flow from w to v_{j+1}. Thus $d(w, v_{j+1}) \leq 4l(d \cdot z^j) \leq 4l \cdot n^{(j+1)\epsilon}\frac{L}{n}$. Note that for any $0 \leq k \leq l$, z^k supports a unit flow from v_k to v_{k+1}; hence $d(v_k, v_{k+1}) \leq d \cdot z^k \leq n^{(k+1)\epsilon}\frac{L}{n}$. Now, $d(w, v_i) \leq d(w, v_{j+1}) + \sum_{k=j+1}^{i-1} d(v_k, v_{k+1}) \leq 4l\frac{L}{n}\sum_{k=j}^{i-1} n^{(k+1)\epsilon} \leq 8l \cdot n^{i\epsilon}\frac{L}{n}$. ∎

We now show how all vertices in U_i can be covered by a v_i-tour.

Lemma 79. *For each $i = 0, \cdots, l-1$, there is a poly-time computable v_i-tour covering vertices U_i, of length $O(\frac{1}{\epsilon^2} n^{(i+1)\epsilon} \log n \cdot \frac{L}{n})$.*

Proof: Fix an $i \in \{0, \cdots, l-1\}$; note that the edge capacities z^i are Eulerian at all vertices except v_i and v_{i+1}. Although applying splitting-off (Theorem 65) requires an Eulerian graph, we can apply it to z^i after adding a dummy (v_{i+1}, v_i) edge of capacity 1, and observing that flows from v_i or flows into v_{i+1} do not use the dummy edge. So using Theorem 65 on vertices $V \setminus (U_i \cup \{v_i, v_{i+1}\})$ and triangle inequality, we obtain edge capacities α on the edges induced by $U_i \cup \{v_i, v_{i+1}\}$ such that: $d \cdot \alpha \leq d \cdot z^i \leq n^{(i+1)\epsilon} \cdot \frac{L}{n}$ and α supports $y^i_u \geq \frac{1}{4l}$ flow from v_i to u and from u to v_{i+1}, for every $u \in U_i$. Below we use B to denote the quantity $n^{(i+1)\epsilon} \cdot \frac{L}{n}$. Consider adding a dummy edge from v_{i+1} to v_i of length B in the induced metric on $U_i \cup \{v_i, v_{i+1}\}$, and set the edge capacity $\alpha(v_{i+1}, v_i)$ on this edge to be 1. Note that

α is Eulerian, has total cost at most $2B$, and every non-trivial cut has value at least $\min\{y^i_u : u \in U_i\} \geq \frac{1}{4l}$. So scaling α uniformly by $4l$, we obtain a fractional feasible solution to ATSP on the vertices $U_i \cup \{v_i, v_{i+1}\}$ (in the modified metric), having cost at most $8l \cdot B$. Since the Frieze et al. [60] algorithm computes an integral tour of length at most $O(\log n)$ times any fractional feasible solution (see Williamson [153] or Theorem 68), we obtain a v_i-tour τ on the modified metric of length at most $O(l \log n) \cdot B$. Since the dummy (v_{i+1}, v_i) edge has length B, it may be used at most $O(l \log n)$ times in τ. So removing all occurrences of this dummy edge gives a set of $O(l \log n)$ paths from v_i to v_{i+1} in the original metric, that together cover U_i. Ignoring vertex v_{i+1} and inserting the direct edge to v_i from the last U_i vertex in each of these paths gives us a v_i-tour covering U_i. Finally note that each of the edges to v_i inserted above has length $O(l \cdot n^{i\epsilon})\frac{L}{n} = O(l) \cdot B$ (from Claim 78), and the number of edges inserted is $O(l \log n)$. So the length of this v_i-tour is at most $O(l \log n) \cdot B + O(l^2 \log n) \cdot B = O(\frac{1}{\epsilon^2} n^{(i+1)\epsilon} \log n \cdot \frac{L}{n})$. ∎

5.4.3 Servicing vertices V_2

We partition vertices in V_2 into $W_0, \cdots, W_{l-1}$, where each W_i contains the vertices in V_2 that are well-covered by z^i. As noted earlier, each vertex $u \in W_i$ in fact has $y^i_u \geq \frac{3}{4} > \frac{2}{3}$. We now consider any particular W_i and obtain a $v_i - v_{i+1}$ path covering the vertices of W_i. Vertices in W_i are covered by a fractional $v_i - v_{i+1}$ path as follows. Splitting off vertices $V \setminus (W_i \cup \{v_i, v_{i+1}\})$ in the fractional solution z^i gives us edge capacities β in the metric induced on $W_i \cup \{v_i, v_{i+1}\}$, such that: β supports at least $\frac{2}{3}$ flow from v_i to u and from u to v_{i+1} for all $u \in W_i$, and $d \cdot \beta \leq d \cdot z^i$ (this is similar to how edge-capacities α were obtained in Lemma 5.4.2). Note that β is a fractional feasible solution to the LP relaxation $(ATSP - path)$ for the ATSP-path instance on the metric induced by $W_i \cup \{v_i, v_{i+1}\}$ with start-vertex v_i and end-vertex v_{i+1}. So if ρ denotes the (constructive) integrality gap of $(ATSP - path)$, we can obtain an integral v_i-v_{i+1} path that spans W_i, having length at most $\rho(d \cdot \beta) \leq \rho(d \cdot z^i) \leq \rho n^{(i+1)\epsilon}\frac{L}{n}$. This requires a polynomial time algorithm that computes an integral path of length at most ρ times the LP value; However even a non-constructive proof of integrality gap ρ' implies a constructive integrality gap $\rho = O(\rho' \log n)$, using the algorithm in Chekuri and Pal [32]. So we obtain:

Lemma 80. *For each $i = 0, \cdots, l-1$, there is a poly-time computable v_i-v_{i+1} path covering W_i of length at most $\rho \cdot n^{(i+1)\epsilon}\frac{L}{n}$.*

5.4.4 Stitching the local paths

We now stitch the v_i-tours that service V_1 (Lemma 79) and the $v_i - v_{i+1}$ paths that service V_2 (Lemma 80), to obtain a single r-path that covers all vertices. For each $i = 0, \cdots, l-1$, let π_i denote the v_i-tour servicing U_i, and let τ_i denote the $v_i - v_{i+1}$ path servicing W_i. The final r-path that the algorithm outputs is the concatenation $\tau^* = \pi_0 \cdot \tau_0 \cdot \pi_1 \cdots \pi_{l-1} \cdot \tau_{l-1}$. From Lemmas 79 and 80, it follows that for all $0 \le i \le l-1$, $d(\pi_i) \le O(\frac{1}{\epsilon^2}\log n) \cdot n^{(i+1)\epsilon}\frac{L}{n}$ and $d(\tau_i) \le O(\rho) \cdot n^{(i+1)\epsilon}\frac{L}{n}$. So the length of τ^* from r until all vertices of $U_i \cup W_i$ are covered is at most $O(\rho + \frac{1}{\epsilon^2}\log n) \cdot n^{(i+1)\epsilon}\frac{L}{n}$ (as $\epsilon \ge \Omega(\frac{1}{\log n})$). This implies that the total latency of vertices in $U_i \cup W_i$ along path τ^* is at most $O(\rho + \frac{1}{\epsilon^2}\log n) \cdot n^{(i+1)\epsilon}\frac{L}{n} \cdot (|W_i| + |U_i|)$.

Moreover, the contribution of each vertex in U_i (resp., W_i) to the LP objective is at least $\frac{1}{4l} \cdot n^{(i+1)\epsilon}\frac{L}{n}$ (resp., $\frac{3}{4} \cdot n^{(i+1)\epsilon}\frac{L}{n}$). Thus the contribution of $U_i \cup W_i$ to the LP objective is at least $\frac{1}{4l} \cdot n^{(i+1)\epsilon}\frac{L}{n} \cdot (|W_i| + |U_i|)$. Using the upper bound on the latency along τ^* for $U_i \cup W_i$, and summing over all i, we obtain that the total latency along τ^* is at most $O(\frac{1}{\epsilon}\rho + \frac{1}{\epsilon^3}\log n)$ times the optimal value of (MLP). From Claim 77, it now follows that the latency of τ^* is $O(\frac{1}{\epsilon}\rho + \frac{1}{\epsilon^3}\log n)n^\epsilon \cdot \mathsf{OPT}$.

Theorem 81. *For any $\Omega(\frac{1}{\log n}) < \epsilon < 1$, there is an $O(\frac{\rho+\log n}{\epsilon^3} \cdot n^\epsilon)$-approximation algorithm for directed latency, that runs in time $n^{O(1/\epsilon)}$, where ρ is the integrality gap of the LP $(ATSP - path)$. Using $\rho = O(\sqrt{n})$, we have a polynomial time $O(n^{\frac{1}{2}+\epsilon})$ approximation algorithm for any fixed $\epsilon > 0$.*

We prove the bound $\rho = O(\sqrt{n})$ in the next section. A bound of $\rho = O(\log n)$ on the integrality gap of $(ATSP - path)$ would imply that this algorithm is a quasi-polynomial time $O(\log^4 n)$ approximation for directed latency.

5.5 Integrality Gap of ATSP-path

In this section we prove an upper bound of $\rho = O(\sqrt{n})$ on the integrality gap of the linear relaxation $(ATSP - path)$ (c.f. Section 5.4). Even for the seemingly stronger LP with one in the right-hand-side of the cut constraints, the best bound on the integrality gap we can obtain is $O(\sqrt{n})$: this follows from the cycle-cover based algorithm of Lam and Newmann [101]. As mentioned in Chekuri and Pal [32], it is unclear whether their $O(\log n)$-approximation can be used to bound the integrality gap of such a linear program. In this section, we present a rounding algorithm for the weaker LP $(ATSP - path)$, which shows $\rho = O(\sqrt{n})$. Our algorithm is similar

to the ATSP-path algorithm of Lam and Newmann [101] and the ATSP algorithm of Frieze et al. [60]; but it needs some more work as we compare the algorithm's solution against a fractional solution to $(ATSP - path)$. We note that the proof holds for any fixed constant $\frac{1}{2} < c < 1$ in the right-hand-side of the cut-constraints of $(ATSP - path)$. Furthermore the reduction from directed latency (Theorem 81) also holds for any constant $\frac{1}{2} < c < 1$. However if $c \leq \frac{1}{2}$, the integrality gap of the LP $(ATSP - path)$ is infinite (eg., a graph consisting of two vertex disjoint paths from s to t).

Let x be any feasible solution to $(ATSP - path)$. We now show how x can be rounded to obtain an integral path spanning all vertices, of total length $O(\sqrt{n})(d \cdot x)$. Let N denote a network (corresponding to the directed metric) on vertices V with edges between all pairs of vertices, where the *cost* of each edge equals its metric length and the *capacity* of each edge is ∞. An extra (t, s) edge of zero cost and capacity three is added to N. The rounding algorithm for x is as follows.

1. Initialize the set of representatives $R \leftarrow V \setminus \{s, t\}$, and the current integral solution $\sigma = \emptyset$.

2. While $R \neq \emptyset$, do:

   (a) Compute a minimum cost circulation $\mathcal{C}$ in $N[R \cup \{s, t\}]$ (the network N induced on $R \cup \{s, t\}$) that sends at least 2 units of flow through each vertex in R. Note that any circulation (in particular $\mathcal{C}$) can be expressed as an edge-disjoint sum of cycles.

   (b) Repeatedly extract from $\mathcal{C}$ all cycles that do not use the extra edge (t, s), to obtain circulation $\mathcal{A} \subseteq \mathcal{C}$. Let $R' \subseteq R$ be the set of R-vertices that have degree at least 1 in $\mathcal{A}$.

   (c) Let $\mathcal{B} = \mathcal{C} \setminus \mathcal{A}$; note that $\mathcal{B}$ is also a circulation and each cycle in it uses edge (t, s).

   (d) If $|R'| \geq \sqrt{n}$, do:

      i. Set $\sigma \leftarrow \sigma \cup \mathcal{A}$.

      ii. *Modify* R by dropping all but one R'-vertex from each strong component of $\mathcal{A}$.

   (e) If $|R'| < \sqrt{n}$, do:

      i. Take an Euler tour on $\mathcal{B}$ and remove all (at most 3) occurrences of edge (t, s) to obtain s-t paths P_1, P_2, P_3.

ii. Restrict each path P_1, P_2, P_3 to vertices in $R \setminus R'$ by short-cutting over R'-vertices, to obtain paths $\tilde{P}_1, \tilde{P}_2, \tilde{P}_3$.

iii. Take a topological ordering $s = w_1, w_2, \cdots, w_h = t$ of vertices $(R \setminus R') \cup \{s, t\}$ relative to the edge-set $\tilde{P}_1 \cup \tilde{P}_2 \cup \tilde{P}_3$.

iv. Set $\sigma \leftarrow \sigma \cup \{(w_j, w_{j+1}) : 1 \leq j \leq h-1\}$.

v. Repeat for each vertex $u \in R'$: find an edge $(w, w') \in \sigma$ such that x supports $\frac{1}{6}$ flow from w to u and from u to w', and modify $\sigma \leftarrow (\sigma \setminus (w, w')) \cup \{(w, u), (u, w')\}$.

vi. Set $R \leftarrow \emptyset$.

3. Output any spanning s-t walk in σ.

We now show the correctness and performance guarantee of the rounding algorithm. We first bound the cost of the circulation obtained in Step 2a during any iteration.

Claim 82. *For any $R \subseteq V \setminus \{s, t\}$, the minimum cost circulation $\mathcal{C}$ computed in step 2a has cost at most $3(d \cdot x)$.*

Proof: The edge values x define a fractional $s - t$ path in network N. Extend x to be a (fractional) circulation by setting $x(t, s) = 1$. We can now apply splitting-off (Theorem 65) on each vertex in $V \setminus R \setminus \{s, t\}$, to obtain capacities x' in network $N[R \cup \{s, t\}]$, such that every pairwise connectivity is preserved and (by triangle inequality) $d \cdot x' \leq d \cdot x$. Note that since neither s nor t is split-off, the extra (t, s) edge is not modified in the splitting-off steps. So x' supports $\frac{2}{3}$ flow from s to each vertex in R; this implies that $3x'$ is a feasible fractional solution to the circulation instance solved in step 2a (note that $x'(t, s)$ remains 1, so solution $3x'$ satisfies the capacity of edge (t, s)). Finally, note that the linear relaxation for circulation is integral (c.f. Nemhauser and Wolsey [117]). So the minimum cost (integral) circulation computed in step 2a has cost at most $3d \cdot x' \leq 3d \cdot x$. ∎

Note that each time step 2d is executed, $|R|$ decreases by at least $\sqrt{n}/2$ (each strong component in $\mathcal{A}$ has at least 2 vertices); so there are at most $O(\sqrt{n})$ such iterations and the cost of σ due to additions in this step is $O(\sqrt{n})(d \cdot x)$ (using Claim 82). Step 2e is executed at most once (at the end); the next claim shows that this step is well defined and bounds the cost incurred.

Claim 83. *In step 2(e)iii, there exists a topological ordering $w_1, \cdots, w_h$ of $(R \setminus R') \cup \{s, t\}$ w.r.t. edges $\tilde{P}_1 \cup \tilde{P}_2 \cup \tilde{P}_3$. Furthermore, $\{(w_j, w_{j+1}) : 1 \leq j \leq h-1\} \subseteq \tilde{P}_1 \cup \tilde{P}_2 \cup \tilde{P}_3$.*

Proof: Note that any cycle in $P_1 \cup P_2 \cup P_3$ is a cycle in $\mathcal{B}$ that does not use edge (t,s), which is not possible by the construction of $\mathcal{B}$ (every cycle in $\mathcal{B}$ uses edge (t,s)); so $P_1 \cup P_2 \cup P_3$ is acyclic. It is clear that if $\tilde{P}_1 \cup \tilde{P}_2 \cup \tilde{P}_3$ contains a cycle, so does $P_1 \cup P_2 \cup P_3$ (each path $\tilde{P}_i$ is obtained by short-cutting the corresponding path P_i). Hence $\tilde{P}_1 \cup \tilde{P}_2 \cup \tilde{P}_3$ is also acyclic, and there is a topological ordering (say $w_1, \cdots, w_h$) of $(R \setminus R') \cup \{s,t\}$ relative to edges $\tilde{P}_1 \cup \tilde{P}_2 \cup \tilde{P}_3$. We now prove the second part of the claim. In circulation $\mathcal{C}$, each vertex of R has at least 2 units of flow through it; but vertices $R \setminus R'$ are not covered (even to an extent 1) in the circulation $\mathcal{A}$. So each vertex of $R \setminus R'$ is covered to extent at least 2 in circulation $\mathcal{B}$, and hence in $P_1 \cup P_2 \cup P_3$. In other words, each vertex of $R \setminus R'$ appears on at least two of the three $s-t$ paths P_1, P_2, P_3. This also implies that (after the short-cutting) each $R \setminus R'$ vertex appears on at least two of the three $s-t$ paths $\tilde{P}_1, \tilde{P}_2, \tilde{P}_3$. Now observe that for each consecutive pair (w_j, w_{j+1}) $(1 \leq j \leq h-1)$ in the topological order, there is a common path $\tilde{P}_k$ (for some $k = 1,2,3$) that contains both w_j and w_{j+1}. Furthermore, in $\tilde{P}_k$, w_j and w_{j+1} are consecutive in that order (otherwise, the topological order would contain a back edge!). Thus each edge (w_j, w_{j+1}) (for $1 \leq j \leq h-1$) is present in $\tilde{P}_1 \cup \tilde{P}_2 \cup \tilde{P}_3$, and we obtain the claim. ∎

We also need the following claim to bound the cost of insertions in step 2(e)v.

Claim 84. *For any two vertices $u', u'' \in V$, if $\lambda(u', u''; x)$ (resp. $\lambda(u'', u'; x)$) denotes the directed connectivity under x from u' to u'' (resp. from u'' to u'), then $\lambda(u', u''; x) + \lambda(u'', u'; x) \geq \frac{1}{3}$.*

Proof: If either u' or u'' is in $\{s,t\}$, the claim is obvious since for every vertex v, x supports $\frac{2}{3}$ flow from s to v and from v to t. Otherwise $\{s,t,u',u''\}$ are distinct, and define capacities $\hat{x}$ as:

$$\hat{x}(v_1,v_2) = \begin{cases} x(v_1,v_2) & \text{for edges } (v_1,v_2) \neq (t,s) \\ 1 & \text{for edge } (v_1,v_2) = (t,s) \end{cases}$$

Observe that $\hat{x}$ is Eulerian; now apply Theorem 65 to $\hat{x}$ and split-off all vertices of V except $T = \{s,t,u',u''\}$, and obtain capacities y on the edges induced on T. We have $\lambda(t_1,t_2;y) = \lambda(t_1,t_2;\hat{x})$ for all $t_1, t_2 \in T$. Note that since neither t nor s is split-off, their degrees in y are unchanged from $\hat{x}$, and also $y(t,s) \geq \hat{x}(t,s) = 1$. Since the out-degree of t in $\hat{x}$ (hence in y) is 1 and $y_{t,s} \geq 1$, we have $y(t,u') = y(t,u'') = 0$ and $y(t,s) = 1$. The capacities y support at least $\frac{2}{3}$ flow from s to u'; so $y(s,u') + y(u'',u') \geq \frac{2}{3}$. Similarly for u'', we have $y(s,u'') + y(u',u'') \geq \frac{2}{3}$, and adding these two inequalities we get $y(u',u'') + y(u'',u') + (y(s,u') + y(s,u'')) \geq \frac{4}{3}$. Note that

$y(s,u') + y(s,u'') \leq y(\delta^+(s)) = \hat{x}(\delta^+(s)) = 1$ (the degree of s is unchanged in the splitting-off). So $y(u',u'') + y(u'',u') \geq \frac{1}{3}$. Since y is obtained from $\hat{x}$ by a sequence of splitting-off operations, it follows that $\hat{x}$ supports flows corresponding to all edges in y *simultaneously*. In particular, the following flows are supported disjointly in $\hat{x}$: $\mathcal{F}_1$ that sends $y(u',u'')$ units from u' to u'', $\mathcal{F}_2$ that sends $y(u'',u')$ units from u'' to u', and $\mathcal{F}_3$ that sends $y(t,s) = 1$ unit from t to s. Hence the flows $\mathcal{F}_1$ and $\mathcal{F}_2$ are each supported by $\hat{x}$ and *do not* use the extra (t,s) edge (since $\hat{x}(\delta^+(t)) = \hat{x}(t,s) = 1$). This implies that the flows $\mathcal{F}_1$ and $\mathcal{F}_2$ are both supported by the original capacities x (where $x(t,s) = 0$). Hence $\lambda(u',u'';x) + \lambda(u'',u';x) \geq y(u',u'') + y(u'',u') \geq \frac{1}{3}$. ■

From Claim 83, we obtain that the cost addition in step 2e(iv) is at most $d(\tilde{P}_1) + d(\tilde{P}_2) + d(\tilde{P}_3) \leq d(P_1) + d(P_2) + d(P_3) \leq 3(d \cdot x)$ (from Claim 82). We now consider the cost addition to σ in step 2(e)v. Claim 84 implies that for any pair of vertices $u', u'' \in V$, x supports $\frac{1}{6}$ flow either from u' to u'' or from u'' to u'. Also for every vertex u, x supports $\frac{2}{3}$ flow from s to u and from u to t. Since σ always contains an $s-t$ path in step 2(e)v, there is always some position along this $s-t$ path to insert any vertex $u \in R'$ as required in step 2(e)v. Furthermore, the cost increase in any such insertion step is at most $12(d \cdot x)$. Hence the total cost for inserting all the vertices R' into σ is at most $12|R'|(d \cdot x) = O(\sqrt{n})(d \cdot x)$. Thus the total cost of σ at the end of the algorithm is $O(\sqrt{n})(d \cdot x)$. Finally note that σ is connected (in the undirected sense), Eulerian at all vertices in $V \setminus \{s,t\}$ and has outdegree 1 at s. This implies that σ corresponds to a spanning $s-t$ walk. This completes the proof of the following.

Theorem 85. *The integrality gap of* $(ATSP - path)$ *is at most* $O(\sqrt{n})$.

Credits: The results in this chapter are from *"Poly-logarithmic Approximation Algorithms for Directed Vehicle Routing Problems"* [111] and *"The Directed Minimum Latency Problem"* [112], obtained jointly with R. Ravi. In independent work, Chekuri et al. [30] also obtained the results on directed orienteering reported here, although via different techniques.

Chapter 6

Permutation Flowshop Scheduling

6.1 Introduction

In the *flow shop* problem, there are m machines located in a fixed order (say, 1 through m), and n jobs each of which consists of a sequence of operations on machines (in the order 1 through m). For any job $j \in \{1, \ldots, n\}$ and machine $i \in \{1, \ldots, m\}$ the length of the operation of job j on machine i is called its *processing time* p_{ij}. A schedule for the jobs is feasible if (i) each machine processes at most one job at any time; (ii) for each job, its operations on the machines are processed in the fixed order 1 through m; and (iii) each operation (of a job on a machine) is processed without interruption. The flow shop problem is a special case of *acyclic job shop* scheduling [40, 48], which in turn is a special case of the general *job shop* scheduling [33, 102].

In this chapter, we study the *permutation flow shop* problem, which is the flow shop problem with the additional constraint that each machine processes all the jobs in the *same* order. So any feasible schedule to the permutation flow shop problem corresponds to a permutation of the n jobs. It is well-known [38] that there exists an optimal schedule for the ordinary flow shop problem having the same processing order (of jobs) on the first two machines and the same processing order on the last two machines. So the permutation flow shop problem is equivalent to the ordinary flow shop problem for $m \leq 3$ machines. However, it is easy to construct an instance with $m = 4$ machines where no permutation schedule is optimal for the ordinary flow shop problem.

Two natural objective functions that are typically considered for scheduling

problems are *makespan* and *weighted completion time*. The makespan of a schedule is the completion time of its last job, i.e. maximum completion time among all jobs. For the weighted completion time objective, each job j comes with a weight $w_j \geq 0$, and the weighted completion time of a schedule is the weighted sum of completion times over all jobs.

6.1.1 Preliminaries

An instance of the permutation flow shop problem with m machines and n jobs is given by an $m \times n$ matrix $P = \{p_{i,j} \mid i = 1, \cdots, m,\ j = 1, \cdots, n\}$ of processing times, where $p_{i,j}$ is the processing time of job j on machine i. We often denote the set $\{1, \cdots, n\}$ of all jobs by $[n]$, and the set $\{1, \cdots, m\}$ of all machines by $[m]$. Any feasible schedule for permutation flow shop corresponds to a permutation of the n jobs. Given a permutation $\pi : [n] \to [n]$ of jobs, the complete schedule of job-operations on machines can be obtained by running jobs on machines in the order π while maintaining the minimum possible wait-time between operations. It is convenient to think of π as a mapping from the set of n possible positions to the set of n jobs. Therefore, $\pi(p)$ denotes the job located at position p. For any permutation π of the n jobs and a job $j \in [n]$, we use C_j^π to denote the completion time of job j under the schedule π; we also denote the makespan of schedule π by $C_{max}^\pi = \max_{j=1}^n C_j^\pi$. Given non-negative weights $\{w_j\}_{j=1}^n$ for the jobs, the weighted completion time objective of the schedule corresponding to permutation π is $\sum_{j=1}^n w_j C_j^\pi$.

The following are two obvious lower bounds for the permutation flow shop scheduling problem with the makespan objective: maximum job length (denoted $l := \max_{j=1}^n \{\sum_{i=1}^m p_{i,j}\}$), and maximum machine load ($L := \max_{i=1}^m \{\sum_{j=1}^n p_{i,j}\}$). We refer to $\max\{l, L\}$ as the trivial lower bound. Our algorithms are based on just these lower bounds.

A *monotone path* (or *critical path*) in an $m \times n$ matrix is defined to be a sequence $\langle (x_1, y_1), \cdots, (x_t, y_t) \rangle$ of matrix positions such that $(x_1, y_1) = (1, 1)$, $(x_t, y_t) = (m, n)$, and for each $1 \leq i < t$ either $x_{i+1} = x_i + 1$ and $y_{i+1} = y_i$ or $x_{i+1} = x_i$ and $y_{i+1} = y_i + 1$. In particular, this definition implies that each monotone path in an $m \times n$ matrix consists of exactly $t = m + n - 1$ positions. We denote the set of all monotone paths in an $m \times n$ matrix by $\mathcal{M}_{m,n}$.

A map $\tau : [n] \to X \cup \{\emptyset\}$ where $X \subseteq [m]$ is called a *partial permutation* if there is a subset $Y \subseteq [n]$ with $|Y| = |X|$ such that (i) $\tau(Y) = X$ (τ is a one-to-one map

from Y to X); and (ii) $\tau(z) = \emptyset$ for all $z \in [n] \setminus Y$. For such a partial permutation τ, we refer to the set X as its *image*, denoted $\mathsf{Image}(\tau)$. A 0-1 $m \times n$ matrix Π is called a *permutation matrix* if every row and column has at most one entry that is 1 (all other entries are 0s). Note that there is an obvious correspondence between partial permutations and permutation matrices. In the rest of the paper we will use partial permutations that map a subset of jobs into a set of machines.

6.1.2 Our Results

We give a simple randomized algorithm (Section 6.2) for minimizing makespan in the permutation flowshop problem, that achieves an approximation guarantee of $2\sqrt{\min\{m,n\}}$. This guarantee is relative to the trivial lower bounds. The analysis is based on a connection between the permutation flow shop scheduling problem and the longest increasing subsequence problem. Hence we answer an open question from Potts et al. [124], by matching algorithmically the $\Omega(\sqrt{\min\{m,n\}})$ gap (between optimal makespan and the trivial lower bound) shown in [124].

We also show how this algorithm can be derandomized to obtain a deterministic approximation guarantee of $3\sqrt{\min\{m,n\}}$. This algorithm uses the derandomization technique of pessimistic estimators due to Raghavan [128] and certain ideas from the concentration of increasing subsequences in random permutations [59]. The details are non-trivial and appear in Section 6.3. We note that among algorithms that are based on the trivial lower bounds, our algorithm is the best possible (up to a $2\sqrt{2}$ factor).

We then consider the weighted completion time objective (Section 6.4) and use our algorithm for minimizing makespan to obtain an $O(\sqrt{\min\{m,n\}})$ approximation algorithm for this problem. This algorithm uses a natural LP relaxation of the problem. The LP rounding algorithm is similar to the approach used in Queranne and Sviridenko [127] (and many other papers on scheduling with the weighted completion time objective [75, 23, 76]). We also show a matching $\Omega(\sqrt{\min\{m,n\}})$ lower bound on the integrality gap of our LP relaxation.

6.1.3 Related Work

Permutation flow shop scheduling has been extensively studied in the operations research literature and there is a significant body of work devoted to the design of heuristics for this problem. The survey paper [56] establishes a common framework

for these heuristics, and describes main classes of algorithms and research directions.

When the number of machines m is a fixed constant, a PTAS is known for the job-shop scheduling problem with the makespan objective due to Jansen et al. [96] and the total weighted completion time objective due to Fishkin et al. [55]. It seems likely that similar techniques yield PTASes for the permutation flow shop scheduling problem under fixed number of machines. For the ordinary flow shop problem with the makespan objective, the best known approximation guarantee is $O(\log m(\log\log m)^{1+\epsilon})$, where $\epsilon > 0$ is any constant, due to Czumaj and Scheideler [40]; in fact this result holds for the more general class of acyclic-shop scheduling. Using the general result from [127] one can derive an approximation algorithm with the same performance guarantee for the flow shop scheduling problem under the weighted completion time objective.

Potts et al. [124] showed a family of instances for permutation flowshop scheduling where the optimal makespan is $\Omega(\sqrt{\min\{m,n\}})$ times the trivial lower bound ($\max\{L,l\}$). It was an open question in [124] to determine whether this bound is tight. The previously best known approximation guarantee for the makespan problem is $O(\sqrt{m\log m})$ due to Sviridenko [146]; this guarantee is also relative to the trivial lower bound. Prior to this, a number of algorithms [131, 119, 120] were shown to have a (tight) worst case guarantee of $\lceil\frac{m}{2}\rceil$. There are also some papers dealing with additive guarantees for the permutation flowshop problem [139, 146]. Sevastianov [139] gave an algorithm that always produces a schedule of length at most $L + O(m^2)\max_{i,j} p_{i,j}$. Sviridenko [146] obtained a similar guarantee of $(1+\delta)L + K_\delta(m\log m)\max_{i,j} p_{i,j}$ for any $\delta > 0$ (here K_δ is a function depending on δ alone). The best multiplicative approximation guarantee obtainable from these results is $O(\sqrt{m\log m})$ [146].

Smutnicki [142] gave worst case analyses of several algorithms for the permutation flow shop problem with the weighted completion time objective. Assuming a ρ_k approximation guarantee for the problem on k machines, [142] gave an $\frac{m}{k}\rho_k$ approximation algorithm for the problem on m machines. Using the $(2+\delta)$-approximation algorithm [140] for the permutation flow shop scheduling problem with the weighted completion time objective and fixed number of machines, one could obtain an ϵm guarantee for minimizing weighted completion time (for any constant $\epsilon > 0$). To the best of our knowledge this is the previously best known result for weighted completion time.

6.2 Randomized Algorithm for Minimizing Makespan

In this section, we give a randomized $\Theta(\sqrt{\min\{m,n\}})$ approximation guarantee for minimizing makespan in the permutation flow shop problem. From the results of Potts et al. [124], it follows that this result is the best possible using the known lower bounds for this problem (namely, machine load and job length). Our algorithm is extremely simple: always output a permutation chosen uniformly at random. The rest of this section proves that this algorithm achieves a guarantee of $2\sqrt{\min\{m,n\}}$.

Given any instance of permutation flow shop, consider the $m \times n$ matrix P of processing times. By standard rounding arguments, we can assume (at the loss of a $1 + o(1)$ factor) that all entries in P are rational. For the purpose of analysis, we scale P appropriately and assume all entries are integral. We first show how P can be decomposed into a collection of smaller matrices having certain properties.

Lemma 86. *Given any matrix $P \in \mathbb{N}_{m\times n}$, there exist $h = \max\{l, L\}$ permutation matrices $\{\Pi_k\}_{k=1}^{h}$ such that $P = \sum_{k=1}^{h} \Pi_k$, where $l = \max_{j=1}^{n}\{\sum_{i=1}^{m} p_{i,j}\}$ and $L = \max_{i=1}^{m}\{\sum_{j=1}^{n} p_{i,j}\}$.*

Proof: Define a bipartite multi-graph graph G corresponding to P as follows. G has vertex bipartition $[m]$ and $[n]$. For every $i \in [m]$ and $j \in [n]$, G contains $p_{i,j}$ parallel edges between i and j. Note that the maximum degree of G is exactly $h = \max\{l, L\}$. By the König edge-coloring theorem for bipartite graphs there is a valid coloring of the edges of G (no two adjacent edges receive the same color) with h colors. Let $E_1, \cdots, E_h$ denote the edges in each color class of such a valid coloring. For each $1 \leq k \leq h$, let Π_k denote the $m \times n$ 0-1 matrix that contains 1s in the positions corresponding to edges of E_k, and 0s everywhere else. Since we have a valid coloring, each E_k is a matching in G; in other words, the matrices $\{\Pi_k\}_{k=1}^{h}$ are all permutation matrices. Further, since each edge of G is assigned some color, we have $P = \sum_{k=1}^{h} \Pi_k$. ∎

Recall that $\mathcal{M}_{m,n}$ denotes the set of all monotone paths in an $m \times n$ matrix. In this section, m and n are fixed; so we abbreviate $\mathcal{M} = \mathcal{M}_{m,n}$. For any permutation σ of the n jobs, monotone paths can be used to characterize the makespan of the schedule resulting from σ as follows:

$$C^{\sigma}_{max} = \max_{\alpha \in \mathcal{M}} \sum_{(i,q)\in\alpha} p_{i,\sigma(q)}.$$

This well-known characterization follows from the fact that the makespan C^{σ}_{max} is lower bounded by the total length of any monotone path since every two consecutive

operations in such a path are either consecutive operations of the same job or consecutive operations on the same machine. It is also easy to build a monotone path that attains the equality. Starting with the operation that finishes last (on the last machine) include all operations preceding to that operation on the same machine till the last idle time. Let J_j be the job whose m-th operation starts on the last machine after that idle time. Include the $m-1$st operation of job J_j into the monotone path. Include all operations preceding that operation on machine $m-1$ till the last idle time before that operation and so on.

Consider any permutation matrix Π_k ($k = 1, \cdots, h$) in the decomposition of Lemma 86. Let the 1-entries of Π_k be in positions $\{(x_1^k, y_1^k), \cdots, (x_r^k, y_r^k)\}$, where $1 \leq x_1^k < \cdots < x_r^k \leq m$ and $y_1^k, \cdots, y_r^k \in [n]$ are distinct elements. Denote $X_k = \{x_1^k, \cdots, x_r^k\}$ and $Y_k = \{y_1^k, \cdots, y_r^k\}$; clearly, $|X_k| = |Y_k| = r \leq \min\{m, n\}$. Define the map $\tau_k : [n] \rightarrow X_k \cup \{\emptyset\}$ where $\tau_k(y_g^k) = x_g^k$ (for all $1 \leq g \leq r$) and $\tau_k(z) = \emptyset$ for $z \notin Y_k$. Since each Π_k is a permutation matrix, it follows that the τ_k is a partial permutation for $k = 1, \ldots, h$.

Finally, for any sequence S of elements from $[m] \cup \emptyset$, define $I(S)$ to be the length of the longest increasing subsequence of numbers in S (ignoring all occurrences of the null element $\emptyset$).

Lemma 87. *For any permutation σ on jobs and any monotone path $\alpha \in \mathcal{M}$,*

$$\sum_{(i,q)\in\alpha} p_{i,\sigma(q)} \leq \sum_{k=1}^{h} I(\tau_k \circ \sigma[n])$$

Proof: Clearly we have:

$$\sum_{(i,q)\in\alpha} p_{i,\sigma(q)} = \sum_{(i,q)\in\alpha} [\sum_{k=1}^{h} \Pi_k(i, \sigma(q))] = \sum_{k=1}^{h} \sum_{(i,q)\in\alpha} \Pi_k(i, \sigma(q))$$

Now consider a particular permutation matrix Π_k (for $k = 1, \cdots, h$) and the sum $\sum_{(i,q)\in\alpha} \Pi_k(i, \sigma(q))$.

Let $S_k = \{(i,q) \in \alpha \mid \Pi_k(i, \sigma(q)) = 1\}$; then the sum $\sum_{(i,q)\in\alpha} \Pi_k(i, \sigma(q)) = |S_k|$. Since Π_k has at most one non-zero entry in each row and column (given by the partial permutation τ_k) and α is a monotone path, we obtain that $S_k = \{(i_1, q_1), \cdots, (i_t, q_t)\}$ (where $t = |S_k|$), with the following properties:

1. $1 \leq i_1 < \cdots < i_t \leq m$ and $\{i_1, \cdots, i_t\} \subseteq X_k$.

2. $1 \leq q_1 < \cdots < q_t \leq n$.

3. $\tau_k(\sigma(q_g)) = i_g$ for all $1 \leq g \leq t$.

From the above, we have that $i_1 < i_2 < \cdots < i_t$ is an increasing subsequence of length t in the sequence $\tau_k \circ \sigma[n] = \langle \tau_k \circ \sigma(1), \cdots, \tau_k \circ \sigma(n) \rangle$; namely given by the positions $q_1 < q_2 < \cdots < q_t$. Thus the longest increasing subsequence in $\tau_k \circ \sigma[n]$ has length at least $|S_k|$. In other words, $\sum_{(i,q)\in\alpha} \Pi_k(i, \sigma(q)) \leq I(\tau_k \circ \sigma[n])$. Summing this expression over all permutation matrices Π_k for $k = 1, \cdots, h$, we obtain the statement of the Lemma. ∎

Note that the right hand side in the inequality in Lemma 87 does not depend on the monotone path α; hence using $C^{\sigma}_{max} = \max_{\alpha\in\mathcal{M}} \sum_{(i,q)\in\alpha} p_{i,\sigma(q)}$, we obtain:

$$C^{\sigma}_{max} \leq \max_{\alpha\in\mathcal{M}} \sum_{k=1}^{h} I(\tau_k \circ \sigma[n]) = \sum_{k=1}^{h} I(\tau_k \circ \sigma[n]). \tag{6.1}$$

We will also need the following:

Lemma 88 (Logan and Shepp [104]; Vershik and Kerov [152])**.** *The expected length of the longest increasing subsequence of a uniformly random permutation on r elements is $(2 + o(1))\sqrt{r}$.*

We are now ready for the main theorem of this section.

Theorem 89. *$E_\sigma[C^{\sigma}_{max}] \leq (2 + o(1))h \cdot \sqrt{\min\{m, n\}}$. Hence there is a randomized polynomial time ($2\sqrt{\min\{m, n\}}$)-approximation algorithm for the permutation flow shop problem.*

Proof: From linearity of expectation applied to inequality (6.1), it suffices to bound $E_\sigma[I(\tau_k \circ \sigma[n])]$ for each $1 \leq k \leq h$. Fix a $1 \leq k \leq h$: since σ is chosen uniformly at random over all permutations, the jobs from Y_k are ordered uniformly at random. Thus $\tau_k \circ \sigma[n]$ is a uniformly random ordering of the elements X_k (ignoring occurrences of $\emptyset$). Applying Lemma 88, we immediately obtain the following which proves the theorem.

$$E_\sigma[I(\tau_k \circ \sigma[n])] \leq (2 + o(1))\sqrt{|X_k|} \leq (2 + o(1))\sqrt{\min\{m, n\}}.$$

∎

Thus we have a very simple randomized $\Theta(\sqrt{\min\{m, n\}})$-approximation algorithm for the permutation flow shop problem, based on the trivial lower bound.

Potts et al. [124] gave a family of examples where the optimal permutation schedule has length at least $\frac{1}{\sqrt{2}}\sqrt{\min\{m,n\}}$ times the lower bound. Hence our result is the best possible guarantee (within a factor of $2\sqrt{2}$) using these lower bound. We note that Theorem 89 also implies that for any instance of flow shop scheduling, there is a permutation schedule of length at most $2\sqrt{\min\{m,n\}}$ times the length of an optimal *non-permutation* schedule; hence this resolves positively the open question in Potts et al. [124] regarding the gap between permutation and non-permutation schedules.

Tight Example. The following simple example shows that the performance guarantee of this randomized algorithm is tight. There are n jobs and $m = 2n$ machines. Each job j (for $1 \leq j \leq n$) has processing time 1 on machines j and $n + j$, and 0 elsewhere. The optimal permutation of jobs is $n, n-1, \cdots, 1$ which results in a makespan of 2. However, it follows from Lemma 88 that a random permutation has expected makespan at least $2\sqrt{n}$.

6.3 A Deterministic Algorithm

We apply the technique of *pessimistic estimators* due to Raghavan [128] to derandomize the algorithm of the previous section, and obtain a deterministic $\Theta(\sqrt{\min\{m,n\}})$-approximation guarantee. We first apply the decomposition of Lemma 86 to obtain h permutation-matrices $\Pi_1, \cdots, \Pi_h$ corresponding to P. By assigning weights $w_1, \cdots, w_h \in \mathbb{N}$ to each of these permutations, we can ensure that $P = \sum_{k=1}^{h} w_k \cdot \Pi_k$ and $h \leq mn$; here $\sum_{k=1}^{h} w_k$ is the trivial lower-bound for the flowhop instance. This computation can be done easily in polynomial time by iteratively using any bipartite matching algorithm. There are many more efficient algorithms for computing an edge-coloring in bipartite multigraphs (See the table in Section 20.9b [136] for running times and references for various edge-coloring algorithms). Furthermore, Lemma 87 implies that for any permutation $\sigma : [n] \to [n]$ of the jobs, the resulting makespan $C^{\sigma}_{max} \leq C^*(\sigma) := \sum_{k=1}^{h} w_k \cdot I(\tau_k \circ \sigma)$, where τ_ks are the partial permutations corresponding to the permutation-matrices Π_ks. From the previous section, we have that $E_\sigma[C^*(\sigma)] \leq 2\sqrt{\min\{m,n\}} \cdot \sum_{k=1}^{h} w_k$. In this section, we give a deterministic algorithm that obtains a permutation σ satisfying $C^*(\sigma) \leq 3\sqrt{\min\{m,n\}} \cdot \sum_{k=1}^{h} w_k$.

In particular, we show that given any collection of h partial permutations $\tau_1, \cdots, \tau_h : [n] \to [m] \cup \{\emptyset\}$, each having a non-empty value on at most r elements, and associated weights $\{w_k\}_{k=1}^{h}$, there is a polynomial time deterministic

algorithm that computes a single permutation $\sigma : [n] \to [n]$ satisfying $C^*(\sigma) = \sum_{k=1}^{h} w_k \cdot I(\tau_k \circ \sigma[n]) \leq 3\sqrt{r} \cdot \sum_{k=1}^{h} w_k$. This immediately implies the desired deterministic approximation guarantee for the permutation flow shop problem since each partial permutation has an image of size at most $r \leq \min\{m, n\}$. In the following, we refer to a permutation that is chosen uniformly at random as u.a.r. permutation.

The algorithm first computes the partial permutations τ_k and weights w_k for $k = 1, \ldots, h$, and then builds the solution σ incrementally. In each step $i = 1, \ldots, n$ we suitably fix the value of $\sigma(i)$ that results in a prefix of the permutation $\langle \sigma(1), \ldots, \sigma(i) \rangle$, i.e. we fix jobs located in the first i positions. The choices for $\sigma(i)$ in each step i are made in such a way that finally, $C^*(\sigma) \leq 3\sqrt{r} \cdot \sum_{k=1}^{h} w_k$. Define the following quantities for any partial permutation τ_k ($1 \leq k \leq h$), step $0 \leq i \leq n$, and elements $a_1, \cdots, a_i \in \mathsf{Image}(\tau_k) \cup \{\emptyset\}$:

$$E_i^k(a_1, \cdots, a_i) := \begin{array}{l} \text{expected value of the longest increasing subsequence} \\ \text{in } \langle a_1, \cdots, a_i, \tau \rangle, \text{ where } \tau \text{ is a permutation} \\ \text{on } \mathsf{Image}(\tau_k) \setminus \{a_1, \cdots, a_i\} \text{ picked u.a.r.} \end{array}$$

$$U_i^k(a_1, \cdots, a_i) := \begin{array}{l} \text{an efficiently computable upper bound on } E_i^k(a_1, \cdots, a_i) \\ \text{(exact definition later).} \end{array}$$

In the above definitions, the elements $a_1, \cdots, a_i$ represent the values $\tau_k \circ \sigma(1), \cdots, \tau_k \circ \sigma(i)$ respectively, obtained from the first i positions of permutation σ, that have been fixed thus far. We also define the expected value of function $C^*(\sigma)$ in step i as functions of the first i positions of permutation σ (that have been fixed):

$$E_i(\sigma(1), \cdots, \sigma(i)) : \begin{array}{l} \text{expected value } E_{\sigma(i+1)\cdots\sigma(n)}[C^*(\sigma)], \text{ with } \langle \sigma(i+1) \cdots \sigma(n) \rangle \\ \text{being u.a.r. permutation on } [n] \setminus \{\sigma(1), \cdots, \sigma(i)\} \end{array}$$

Note that for any $1 \leq k \leq h$, since $\langle \sigma(i+1), \cdots, \sigma(n) \rangle$ is u.a.r. permutation on $[n] \setminus \{\sigma(1), \cdots, \sigma(i)\}$, we obtain that $\langle \tau_k \circ \sigma(i+1), \cdots, \tau_k \circ \sigma(n) \rangle$ is u.a.r. permutation on $\mathsf{Image}(\tau_k) \setminus \{\tau_k \circ \sigma(j) : 1 \leq j \leq i\}$. Thus we can rewrite E_i as:

$$E_i(\sigma(1), \cdots, \sigma(i)) := \sum_{k=1}^{h} w_k \cdot E_i^k(\tau_k \circ \sigma(1), \cdots, \tau_k \circ \sigma(i))$$

We also define the efficiently computable upper bound on E_i as:

$$U_i(\sigma(1), \cdots, \sigma(i)) := \sum_{k=1}^{h} w_k \cdot U_i^k(\tau_k \circ \sigma(1), \cdots, \tau_k \circ \sigma(i))$$

The precise definition of the upper bound functions U_i^k for $k = 1, \ldots, h$ and $i = 0, \ldots, n$ appears in the next subsection. In Lemmas 90 and 91, we prove some important properties of the functions U_i, which allow us to derandomize the algorithm of the previous section to obtain Theorem 92.

6.3.1 Properties of the pessimistic estimator

Recall the definition of $E_i^k(a_1, \cdots, a_i)$; we now construct an upper bound $U_i^k(a_1, \cdots, a_i)$ for this expected value. Fix a parameter $t = 3\sqrt{r}$, where $r = \max_{k=1}^{h} |\mathsf{Image}(\tau_k)|$ is an upper bound on the length of each partial permutation. Define $S_i^k(a_1, \cdots, a_i)$ to be the *expected number* of t-length increasing subsequences in $\langle a_1, \cdots, a_i, \tau \rangle$, when τ is u.a.r. permutation on $\mathsf{Image}(\tau_k) \setminus \{a_1, \cdots, a_i\}$. We can now upper bound $E_i^k(a_1, \cdots, a_i)$, the expected length of the longest increasing subsequence in $\langle a_1, \cdots, a_i, \tau \rangle$, as follows:

$$\begin{aligned} E_i^k(a_1, \cdots, a_i) &\leq t \cdot Pr_\tau[\langle a_1, \cdots, a_i, \tau \rangle \text{ has no } t\text{-length increasing subsequence}] \\ &\quad + r \cdot Pr_\tau[\langle a_1, \cdots, a_i, \tau \rangle \text{ contains a } t\text{-length increasing subsequence}] \\ &\leq t + r \cdot Pr_\tau[\langle a_1, \cdots, a_i, \tau \rangle \text{ contains a } t\text{-length increasing subsequence}] \\ &\leq t + r \cdot S_i^k(a_1, \cdots, a_i) \end{aligned}$$

Define the upper bound U_i^k on the expected value E_i^k as:

$$U_i^k(a_1, \cdots, a_i) := t + r \cdot S_i^k(a_1, \cdots, a_i) \qquad \forall 1 \leq k \leq h$$

Let $N_i^k = \mathsf{Image}(\tau_k) \setminus \{a_1, \cdots, a_i\} \subseteq [m]$; and for any set T, let $\mathcal{P}(T)$ denote the set of all permutations of the elements of T. We first show that each U_i^k can be efficiently computed, which implies the same for the functions $\{U_i\}_{i=0}^n$.

Lemma 90. *For any $1 \leq k \leq h$, $i \in \{0, \cdots, n\}$ and $a_1, \cdots, a_i \in \mathsf{Image}(\tau_k) \cup \{\emptyset\}$, the value $U_i^k(a_1, \cdots, a_i)$ can be computed exactly in polynomial time.*

Proof: Fix any values of $1 \leq k \leq h$, $0 \leq i \leq n$ and $a_1, \cdots, a_i \in \mathsf{Image}(\tau_k) \cup \{\emptyset\}$. Clearly it suffices to show that $S_i^k(a_1, \cdots, a_i)$ can be computed in polynomial time. We say that a t-length increasing subsequence s is *feasible* if there is some permutation $\tau \in \mathcal{P}(N_i^k)$ such that s is a subsequence in $\langle a_1, \cdots, a_i, \tau \rangle$. Let $\mathcal{I}$ denote the set of all such feasible t-length increasing subsequences. Then we can partition $\mathcal{I}$ as $\left(\sqcup \{\mathcal{I}_{j,q} \mid 1 \leq j \leq i \text{ and } 1 \leq q \leq t\} \right) \sqcup \mathcal{I}_{0,0}$ where:

$$\mathcal{I}_{0,0} = \{\tau^0 \mid \tau^0 \text{ is a } t\text{-length increasing sequence of numbers from } N_i^k\}$$

$$\mathcal{I}_{j,q} = \left\{ \langle \tau', \tau'' \rangle \;\middle|\; \begin{array}{l} \tau' \text{ is a } q \text{ length increasing subsequence in } \langle a_1, \cdots, a_j \rangle \\ \text{ending at } a_j \neq \emptyset, \text{ and } \tau'' \text{ is a } t-q \text{ length increasing} \\ \text{sequence of numbers from } \{e \in N_i^k : e > a_j\} \end{array} \right\}$$

Note that given any $j \in \{1, \cdots, i\}$ and $q \in \{1, \cdots, t\}$, one can compute in polynomial time, the number of q-length increasing subsequences in $\langle a_1, \cdots, a_j \rangle$ that end at a_j; we denote this quantity by $\#I(j,q)$. The computation of $\#I(j,q)$ is based on a dynamic program using the following recurrence:

$$\#I(j,q) = \begin{cases} \sum\{\#I(j', q-1) \mid 1 \leq j' < j,\ a_{j'} < a_j\} & a_j \neq \emptyset,\ q \geq 2 \\ 1 & a_j \neq \emptyset,\ q = 1 \\ 0 & a_j = \emptyset \end{cases}$$

For ease of notation in the following, let $\#I(0,0) = 1$. For every $1 \leq j \leq i$, denote the set $\{e \in N_i^k : e > a_j\}$ by L_j, and also let $L_0 = N_i^k$. Note that, for each part $\mathcal{I}_{j,q}$ (in the partition of $\mathcal{I}$), its size $|\mathcal{I}_{j,q}| = \#I(j,q) \cdot \binom{|L_j|}{t-q}$ (the first term corresponds to a q length increasing subsequence of $\langle a_1, \cdots, a_j \rangle$, and the second term corresponds to a $t-q$ length increasing sequence from L_j). When $\tau \in \mathcal{P}(N_i^k)$ is picked u.a.r., the induced permutation on each set L_j (for $0 \leq j \leq i$) is also u.a.r. Hence for each part $\mathcal{I}_{j,q}$, the probability that any particular subsequence $s \in \mathcal{I}_{j,q}$ appears in $\langle a_1, \cdots, a_i, \tau \rangle$ is exactly $1/(t-q)!$. (the last $t-q$ entries of s come from the random permutation τ). So we have:

$$\begin{array}{l} E_\tau\big[|\{s \in \mathcal{I}_{j,q} : s \text{ is subsequence of } \langle a_1, \cdots, a_i, \tau \rangle\}|\big] \\ = \ \sum_{s \in \mathcal{I}_{j,q}} Pr_\tau\big[s \text{ is subsequence of } \langle a_1, \cdots, a_i, \tau \rangle\big] \\ = \ |\mathcal{I}_{j,q}| \cdot \frac{1}{(t-q)!} \\ = \ \#I(j,q) \cdot \binom{|L_j|}{t-q} \cdot \frac{1}{(t-q)!} \end{array}$$

Thus, we can write $S_i^k(a_1, \cdots, a_i)$ as

$$\begin{array}{l} E_{\tau \in \mathcal{P}(N_i^k)}\big[\ |\{s \in \mathcal{I} : s \text{ is subsequence of } \langle a_1, \cdots, a_i, \tau \rangle\}|\ \big] \\ = \ \sum_{j,q} E_{\tau \in \mathcal{P}(N_i^k)}\big[\ |\{s \in \mathcal{I}_{j,q} : s \text{ is subsequence of } \langle a_1, \cdots, a_i, \tau \rangle\}|\ \big] \\ = \ \sum_{j,q} \#I(j,q) \cdot \binom{|L_j|}{t-q} \cdot \frac{1}{(t-q)!} \end{array}$$

The lemma follows. ∎

Lemma 91. *For any $0 \leq i \leq n$ and any prefix (possibly empty) $\sigma(1), \cdots, \sigma(i) \in [n]$ of a permutation σ,*

$$\min_{\sigma(i+1) \in [n] \setminus \{\sigma(1), \cdots, \sigma(i)\}} U_{i+1}(\sigma(1), \cdots, \sigma(i), \sigma(i+1)) \leq U_i(\sigma(1), \cdots, \sigma(i))$$

Proof: Fix any i and a prefix $\sigma(1), \cdots, \sigma(i)$ of a permutation σ, and let $M = [n] \setminus \{\sigma(1), \cdots, \sigma(i)\}$. We first prove the following for an arbitrary $1 \leq k \leq h$:

$$S_i^k(\tau_k \circ \sigma(1), \cdots, \tau_k \circ \sigma(i)) = \frac{1}{n-i} \sum_{x \in M} S_{i+1}^k(\tau_k \circ \sigma(1), \cdots, \tau_k \circ \sigma(i), \tau_k(x)) \quad (6.2)$$

For ease of notation, let $a_j^k = \tau_k \circ \sigma(j)$ for all $1 \leq j \leq i$. Let $N_i^k = \mathsf{Image}(\tau_k) \setminus \{a_1^k, \cdots, a_i^k\} \subseteq [m]$ denote the remaining elements of $\mathsf{Image}(\tau_k)$, and $n_i^k = |N_i^k|$. Recall that,

$$S_i^k(a_1^k, \cdots, a_i^k) = E_\tau[\text{number of } t\text{-length increasing subsequences in } \langle a_1^k \cdots a_i^k, \tau \rangle]$$

where $\tau \in \mathcal{P}(N_i^k)$ is picked u.a.r. So multiplying both sides of (6.2) by $n_i^k! = |\mathcal{P}(N_i^k)|$, we can rewrite its left hand side as:

$$LHS' = n_i^k! \times S_i^k(a_1^k, \cdots, a_i^k) = \sum_{\tau \in \mathcal{P}(N_i^k)} \#I_t(a_1^k, \cdots, a_i^k, \tau) \quad (6.3)$$

Above, for any sequence s, $\#I_t(s)$ denotes the number of t-length increasing subsequences in s. To compute the right hand side of (6.2), we split the summation into $M^{(k)} = \{x \in M \mid \tau_k(x) \neq \emptyset\}$ and $M \setminus M^{(k)} = \{x \in M \mid \tau_k(x) = \emptyset\}$. Note that $|M| = n - i$ and $|M^{(k)}| = n_i^k$. For any $x \in M \setminus M^{(k)}$, it is easy to see that $S_{i+1}^k(a_1^k, \cdots, a_i^k, \tau_k(x)) = S_i^k(a_1^k, \cdots, a_i^k)$. Now the right hand side of (6.2) (scaled by $n_i^k!$) can be written as:

$$\begin{aligned} & n_i^k! \times \frac{n-i-n_i^k}{n-i} S_i^k(a_1^k, \cdots, a_i^k) + n_i^k! \times \frac{1}{n-i} \sum_{x \in M^{(k)}} S_{i+1}^k(a_1^k, \cdots, a_i^k, \tau_k(x)) \\ = \; & (1 - \frac{n_i^k}{n-i}) LHS' + n_i^k! \times \frac{1}{n-i} \sum_{x \in M^{(k)}} S_{i+1}^k(a_1^k, \cdots, a_i^k, \tau_k(x)) \end{aligned}$$

Thus in order to prove (6.2), it suffices to show:

$$LHS' = (n_i^k - 1)! \times \sum_{x \in M^{(k)}} S_{i+1}^k(a_1^k, \cdots, a_i^k, \tau_k(x)) \quad (6.4)$$

Note that τ_k induces a bijection between $M^{(k)}$ and N_i^k: $|M^{(k)}| = |N_i^k|$ and $\tau_k(M^{(k)}) = N_i^k$. Thus we can rewrite the right hand side in (6.4) as:

$$(n_i^k - 1)! \sum_{y \in N_i^k} S_{i+1}^k(a_1^k, \cdots, a_i^k, y) = \sum_{y \in N_i^k} \sum_{\tau' \in \mathcal{P}(N_i^k \setminus y)} \#I_t(a_1^k, \cdots, a_i^k, y, \tau')$$

To prove (6.4), using the expression for LHS' from (6.3), it suffices to show that:

$$\sum_{\tau \in \mathcal{P}(N_i^k)} \#I_t(a_1^k, \cdots, a_i^k, \tau) = \sum_{y \in N_i^k} \sum_{\tau' \in \mathcal{P}(N_i^k \setminus y)} \#I_t(a_1^k, \cdots, a_i^k, y, \tau')$$

Now observe that $\mathcal{P}(N_i^k) = \sqcup_{y \in N_i^k} \{\langle y, \tau' \rangle \mid \tau' \in \mathcal{P}(N_i^k \setminus y)\}$. Thus the summations in the two expressions above run over exactly the same set of sequences, and this implies equality (6.4) which in turn gives equation (6.2). We are now ready to complete the proof of the lemma.

$$\begin{aligned}
&\min_{\sigma(i+1) \in M} U_{i+1}(\sigma(1), \cdots, \sigma(i), \sigma(i+1)) \leq \frac{1}{n-i} \sum_{x \in M} U_{i+1}(\sigma(1), \cdots, \sigma(i), x) \\
&= \frac{1}{n-i} \sum_{x \in M} \sum_{k=1}^{h} w_k \big[t + r \cdot S_{i+1}^k(\tau_k \circ \sigma(1), \cdots, \tau_k \circ \sigma(i), \tau_k(x)) \big] \\
&= \frac{|M|}{n-i} \cdot t \sum_{k=1}^{h} w_k + \frac{r}{n-i} \sum_{x \in M} \sum_{k=1}^{h} w_k \cdot S_{i+1}^k(\tau_k \circ \sigma(1), \cdots, \tau_k \circ \sigma(i), \tau_k(x)) \\
&= t \sum_{k=1}^{h} w_k + r \cdot \sum_{k=1}^{h} w_k \cdot \frac{1}{n-i} \sum_{x \in M} S_{i+1}^k(\tau_k \circ \sigma(1), \cdots, \tau_k \circ \sigma(i), \tau_k(x)) \\
&= t \sum_{k=1}^{h} w_k + r \cdot \sum_{k=1}^{h} w_k \cdot S_i^k(\tau_k \circ \sigma(1), \cdots, \tau_k \circ \sigma(i)) \qquad \text{(Using equation 6.2)} \\
&= \sum_{k=1}^{h} w_k \cdot U_i^k(\tau_k \circ \sigma(1), \cdots, \tau_k \circ \sigma(i)) \\
&= U_i(\sigma(1), \cdots, \sigma(i))
\end{aligned}$$

Thus we have the lemma. ∎

6.3.2 Applying the pessimistic estimators

We now use the upper-bound functions U_i for $i = 1, \ldots, n$ described in the previous subsection to obtain a deterministic approximation algorithm for the permutation flow shop problem. This algorithm follows the general framework of the method of pessimistic estimators.

Theorem 92. *There is a deterministic polynomial time $3\sqrt{\min\{m, n\}}$ approximation algorithm for the permutation flow shop scheduling problem with makespan objective.*

Proof: We now describe our final deterministic algorithm:

1. Decompose the matrix P of processing times according to Lemma 86, to obtain $h \leq mn$ permutation-matrices with corresponding weights $\{\Pi_k, w_k\}_{k=1}^{h}$, such that $P = \sum_{k=1}^{h} w_k \cdot \Pi_k$ and $\sum_{k=1}^{h} w_k$ equals the trivial lower-bound for the permutation flow shop instance.

2. For each $1 \leq k \leq h$, τ_k denotes the partial permutation corresponding to permutation-matrix Π_k.

3. For each $i = 1, \cdots, n$: set $\sigma(i) \leftarrow x$ for the value $x \in [n] \setminus \{\sigma(1), \cdots, \sigma(i-1)\}$ that minimizes the function value $U_i(\sigma(1), \cdots, \sigma(i-1), x)$.

As mentioned earlier, the decomposition in step 1 can be carried out in polynomial time using an edge-coloring algorithm. In step 3, the algorithm uses the efficiently computable functions $\{U_i\}_{i=0}^{n}$ (see Lemma 90) to fix the solution σ step by step. Hence the above algorithm runs in polynomial time. The rest of this proof shows that it achieves the desired approximation guarantee.

We claim that for each $i \in \{0, \cdots, n\}$, $U_i(\sigma(1), \cdots, \sigma(i)) \leq W \cdot (t+2)$ where $W = \sum_{k=1}^{h} w_k$ is the trivial lower-bound (recall that $t = 3\sqrt{r} \leq 3\sqrt{\min\{m,n\}}$). Assuming that the base case (i.e. $i = 0$) for this claim holds, using Lemma 91 and induction, we obtain that the claim is true for all values of $i \geq 1$. It remains to prove the claim for $i = 0$: here U_0 takes no arguments and is a fixed value $U_0 = tW + r\sum_{k=1}^{h} w_k \cdot S_0^k$. From the definition of the S_i^ks, we have that each S_0^k is the expected number of t-length increasing subsequences in a u.a.r. permutation of $\mathsf{Image}(\tau_k)$. Since $\mathsf{Image}(\tau_k)$ has at most r elements, using linearity of expectation, it follows that $S_0^k \leq \binom{r}{t} \cdot \frac{1}{t!}$ for every $k = 1, \cdots, h$. We have,

$$\begin{aligned} U_0 &\leq tW + rW\binom{r}{t}\frac{1}{t!} = tW + rW\frac{r!}{(r-t)!t!}\frac{1}{t!} \leq tW + rW\frac{r^t}{(t!)^2} \\ &\leq tW + rW\left(\frac{re^2}{t^2}\right)^t = tW + rW\left(\frac{e^2}{9}\right)^t = tW + rW\left(\frac{e}{3}\right)^{6\sqrt{r}} \leq W \cdot (t+2) \end{aligned}$$

Now observe that after the last step, $E_n(\sigma(1), \cdots, \sigma(n))$ is exactly the value $C^*(\sigma)$ (at this point all positions have been fixed, so there is no randomness left in the expected value E_n). Since the function U_n upper bounds E_n, we have $C^*(\sigma) = E_n(\sigma(1), \cdots, \sigma(n)) \leq U_n(\sigma(1), \cdots, \sigma(n)) \leq W \cdot (t+2)$. Now the theorem follows from the fact that W equals the trivial lower-bound for the permutation flow shop instance and $r \leq \min\{m, n\}$. ∎

Remark. The deterministic algorithm described above can be viewed as greedily fixing the permutation of jobs one by one, where the next job to be added is chosen according to a certain potential function. This algorithm is similar to the well-known "insertion heuristic" first suggested by Nawaz, Enscore and Ham [116], that initially orders jobs in decreasing order of job lengths and inserts them one by one into the current permutation, always choosing the position for a job in the current permutation that causes the least increase in makespan. The difference is that our algorithm adds jobs in the order of the final permutation (so at any point, a prefix of the permutation is fixed), whereas the insertion heuristic [116] just fixes a relative ordering of the first few jobs allowing unscheduled jobs to be scheduled in between later on. This insertion heuristic is known to have excellent performance in practice [142], but there is no known analytic explanation (there is an $\Omega(\sqrt{m})$ lower bound [120]). A natural way of analyzing such a heuristic would be to define a potential function that improves on every step of the greedy procedure and relate this function to the makespan. Although we have not been able to extend our current analysis to prove an $O(\sqrt{m})$ worst case bound for the insertion heuristic, we show that another greedy procedure does achieve this bound (Theorem 92), and these ideas may help in obtaining a tight bound for the insertion heuristic.

6.4 Weighted sum of completion times

In this section, we consider the permutation flow shop problem with the objective being the weighted sum of completion times. We show that our algorithm for the makespan objective can be used within an LP-based approach to obtain an $O(\sqrt{\min\{m,n\}})$ approximation algorithm for weighted completion time. This approach is similar to that used in Queyranne and Sviridenko [127] (and many other papers [75, 23, 76]), where the authors considered a class of job shop problems (these do not have the permutation constraint). We consider the following linear relaxation for the permutation flow shop problem with weighted completion time as objective. In fact this LP [127] is a relaxation for even the usual flow shop

problem (without the permutation constraint).

$$\min \sum_{j=1}^{n} w_j \cdot C_j, \tag{6.5}$$

$$z_{1,j} \geq p_{1,j}, \qquad \forall 1 \leq j \leq n \tag{6.6}$$

$$z_{i,j} \geq z_{i-1,j} + p_{i,j}, \qquad \forall 2 \leq i \leq m, 1 \leq j \leq n \tag{6.7}$$

$$\sum_{j \in A} p_{i,j} \cdot z_{i,j} \geq \frac{1}{2}(\sum_{j \in A} p_{i,j})^2 + \frac{1}{2} \sum_{j \in A} p_{i,j}^2, \qquad \forall A \subseteq [n], 1 \leq i \leq m, \tag{6.8}$$

$$C_j = z_{m,j}, \qquad \forall 1 \leq j \leq n \tag{6.9}$$

$$z_{i,j} \geq 0, \qquad \forall 1 \leq i \leq m, 1 \leq j \leq n \tag{6.10}$$

Here each variable $z_{i,j}$ denotes the time when job j's operation on machine i is completed; and $C_j = z_{m,j}$ is the completion time of job j. Constraints (6.7) ensure that the operations of each job are performed in the flow shop order. Constraints (6.8) are a relaxation of the machine resource constraints. The weighted completion time objective is captured in (6.5). As shown in Queyranne [126], this LP can be solved in polynomial time using the Ellipsoid algorithm (the separation oracle for the exponential-sized constraints (6.8) reduces to a submodular function minimization [136]). The algorithm first obtains an optimal solution (z, C) to the above LP. Then it reduces the weighted completion time problem to one of minimizing makespan as outlined below.

1. Group the jobs $[n]$ based on their fractional completion times C_j. For each integer $a \geq 0$, group G_a consists of all jobs $1 \leq j \leq n$ such that $C_j \in (2^a, 2^{a+1}]$. Note that there are at most n non-empty groups.

2. For each non-empty group G_a, run the $O(\sqrt{\min\{m, n\}})$ approximation algorithm for minimizing makespan, to obtain a permutation π_a of G_a.

3. Output the permutation of jobs $[n]$ given by $\pi_0, \pi_1, \cdots$.

The remaining analysis is identical to the one in Queyranne and Sviridenko [127]; however it is presented here in the context of permutation flowshop, for the sake of completeness. For any group G_a, let $l_a = \max\{\sum_{i=1}^{m} p_{i,j} \mid j \in G_a\}$ denote the maximum job length, and $L_a = \max\{\sum_{j \in G_a} p_{i,j} \mid 1 \leq i \leq m\}$ the maximum machine load. We first prove the following auxiliary lemma.

Lemma 93. *For each group G_a, $\max\{l_a, L_a\} \leq 2^{a+2}$*

Proof: For any job j, constraints (6.7) imply $C_j = z_{m,j} \geq z_{m-1,j} + p_{m,j} \geq \cdots \geq \sum_{i=1}^{m} p_{i,j}$. Hence $l_a = \max\{\sum_{i=1}^{m} p_{i,j} \mid j \in G_a\} \leq \max\{C_j \mid j \in G_a\} \leq 2^{a+1}$.

For any machine i, constraint (6.8) applied to subset $A = G_a$ implies $\sum_{j \in G_a} p_{i,j} \cdot z_{i,j} \geq \frac{1}{2}(\sum_{j \in G_a} p_{i,j})^2$. Furthermore, constraints (6.7) imply $z_{i,j} \leq C_j$ for all machines i and jobs j. Hence, $\frac{1}{2}(\sum_{j \in G_a} p_{i,j})^2 \leq \sum_{j \in G_a} p_{i,j} \cdot C_j \leq 2^{a+1} \sum_{j \in G_a} p_{i,j}$. In other words, $\sum_{j \in G_a} p_{i,j} \leq 2^{a+2}$ for all machines i. Thus $L_a \leq 2^{a+2}$. ∎

Theorem 94. *There is a polynomial time $O(\sqrt{\min\{m,n\}})$ approximation algorithm for minimizing weighted completion time in the permutation flow shop problem.*

Proof: Recall that the approximation guarantee of our algorithm for minimizing makespan (Section 6.3) is relative to the trivial lower bound. Along with the Lemma 7.2, we obtain that for each group G_a, the resulting makespan $C_{max}(\pi_a) \leq \rho \cdot \max\{l_a, L_a\} \leq \rho \cdot 2^{a+2}$, where $\rho = O(\sqrt{\min\{m,n\}})$ is the approximation guarantee for the makespan problem. Under the final permutation $\langle \pi_0, \pi_1, \cdots \rangle$, the completion time of each job in group G_a is at most $\sum_{b=0}^{a} C_{max}(\pi_b) \leq 4\rho \sum_{b=0}^{a} 2^b \leq 8\rho \cdot 2^a$. But in the LP solution, $C_j \geq 2^a$ for all $j \in G_a$ and groups G_a. This implies that the weighted completion time of the final permutation is at most 8ρ times the optimal LP value. ∎

Integrality gap of the linear program (6.5)-(6.10). We observe that the example of Potts et al. [124] (comparing permutation and non-permutation schedules) also gives an $\Omega(\sqrt{\min\{m,n\}})$ lower bound on the integrality gap of the linear program (6.5)-(6.10). So our algorithm is the best possible approximation algorithm based on this linear programming relaxation. For any $n \in \mathbb{N}$, let $\mathcal{I}_n$ denote the following instance of permutation flow shop: there are n jobs and $2n$ machines; for each $j = 1, \cdots, n$, job j has processing time 1 on machines j and $2n + 1 - j$, and 0 elsewhere. It was shown in [124] that the optimal makespan $C^*_{max}(\mathcal{I}_n) \geq \sqrt{2n}$ for all $n \geq 1$. Consider the objective of minimizing the *total completion time* for instance $\mathcal{I}_n$; i.e. the weighted completion time objective with all weights $w_j = 1$ ($1 \leq j \leq n$). Note that for any $1 \leq k \leq n$, *any* set of k jobs in the instance $\mathcal{I}_n$ is equivalent to the instance $\mathcal{I}_k$. Hence, for any permutation of the jobs, the completion time of the k-th job in the permutation is at least $C^*_{max}(\mathcal{I}_k) \geq \sqrt{2k}$, for all $1 \leq k \leq n$. Thus the optimal total completion time of instance $\mathcal{I}_n$ is at least $\sum_{k=1}^{n} \sqrt{2k} = \Omega(n^{3/2})$. We now construct a fractional feasible solution (z, C) to the linear program (6.5)-(6.10) having objective value $O(n)$, which would establish the claimed integrality gap. The z-variables of each job $j \in [n]$ are set as follows: $z_{i,j} = 0$ for $1 \leq i < j$, $z_{i,j} = 1$ for $j \leq i < 2n - j + 1$, and $z_{i,j} = 2$ for $2n - j + 1 \leq i \leq 2n$. This fixes the fractional completion-time $C_j = z_{2n,j} = 2$ for all jobs j, and the objective value is $2n$. The

only non-trivial constraints to check are (6.7) and (6.8). From the construction of solution (z, C), it follows that constraints (6.7) are satisfied. To see that constraints are (6.8) are satisfied, consider any machine i: for every $A \subseteq [n]$, the right-hand-side of (6.8) is either 0 or 1; moreover whenever it is 1, the left-hand-side of (6.8) is 1 as well.

Credits: The results in this chapter are from *"Tight Bounds for Permutation Flowshop Scheduling"* [113], obtained jointly with Maxim Sviridenko. Recently and independently of our work, Sotelo and Poggi de Aragao [143] designed a deterministic approximation algorithm for minimizing makespan in the permutation flow shop problem. The performance guarantee of their algorithm is $2\sqrt{2n+m}$ which is slightly worse than ours. Although their algorithm and analysis are different from ours they also use the connection to increasing subsequences in permutations.

Chapter 7

Maximum Quadratic Assignment

7.1 Introduction

Quadratic assignment is a basic problem in combinatorial optimization, which generalizes several other problems such as *traveling salesman* [7], *linear arrangement* [47, 79] and *dense k subgraph* [50]. The input to quadratic assignment consists of two $n \times n$ symmetric non-negative matrices $W = (w_{i,j})$ and $D = (d_{i,j})$. Given matrices W, D, and a permutation $\pi : [n] \to [n]$, the quadratic assignment objective is $Q(\pi) := \sum_{i,j \in [n], i \neq j} w_{i,j} \cdot d_{\pi(i),\pi(j)}$.

There are two variants of the Quadratic Assignment Problem. In the *Minimum Quadratic Assignment* problem, the objective is to find a permutation π that minimizes $Q(\pi)$. In this paper we study the *Maximum Quadratic Assignment* (Max-QAP) problem, where the objective is to find a permutation π that maximizes $Q(\pi)$. An indication of the hardness of approximating this problem is that it contains the well-known *dense k subgraph* problem as a special case. As mentioned in Chapter 2, the best known approximation guarantee for dense k subgraph is $n^{1/3-\delta}$ [50] (for some universal constant $\delta > 0$), and improving this is a long-standing open question.

7.1.1 Our Results

We give an $O(\sqrt{n} \log^2 n)$ approximation algorithm for Max-QAP, which is the first non-trivial approximation guarantee for this problem. In fact, this bound also holds when the matrices W and D are *asymmetric*. Using standard scaling arguments,

our algorithm reduces Max-QAP to a special case (called *0-1* Max-QAP) where the matrices have only 0-1 entries; in this case matrices W, D naturally correspond to a pair of undirected graphs. We then present an $O(\sqrt{n})$ approximation algorithm for 0-1 Max-QAP. We note that 0-1 Max-QAP itself contains the dense k subgraph problem as a special case. The algorithm for 0-1 Max-QAP involves taking the better of the following two approaches: **(1)** The first algorithm outputs a random permutation on appropriately chosen (equal-sized) dense subgraphs (or submatrices) of W and D. To find these subgraphs, we use a 2-approximation algorithm for *Vertex Cover* [150] in one graph, and an $\frac{n}{k}$-approximation algorithm for *Dense k Subgraph* [50] in the other graph. **(2)** The second algorithm uses local search to obtain a constant factor approximation for a new problem, *Common Star Packing*, which also defines a feasible solution to Max-QAP. These results appear in Section 7.2

We also consider a special case of the general Max-QAP, where one of the matrices W or D satisfies triangle inequality. For the Max-QAP*with triangle inequality*, we give a $\frac{2e}{e-1} \approx 3.16$ approximation algorithm, that improves the previously best known ratio of 4 due to Arkin et al. [9]. Our approach here is as follows. We first define an auxiliary linear ordering problem and show that it is equivalent (up to a factor 2) to Max-QAP with triangle inequality. This auxiliary problem generalizes the *Maximum Vertex Cover* problem [3, 51]. We obtain an $\frac{e}{e-1}$ approximation algorithm for the auxiliary problem, by rounding a natural LP-relaxation for it; this implies the result for Max-QAP with triangle inequality. These results appear in Section 7.3.

In Section 7.4 we note that a natural LP relaxation (c.f. Adams and Johnson [1]) for Max-QAP can be shown to have an $\tilde{\Omega}(\sqrt{n})$ integrality gap. On the other hand, when restricted to Max-QAP with triangle inequality the integrality gap of this LP is at most $\frac{2e}{e-1}$.

7.1.2 Related Work

Quadratic assignment is an extensively studied combinatorial optimization problem. The book by Cela [22] surveys several bounding techniques, exact algorithms, and polynomially solvable special cases. Surveys on the quadratic assignment problem include Pardalos and Wolkowitz [122], Loilola et al. [105], and Burkard et al. [20].

Approximation algorithms for maximum quadratic assignment have been obtained in many special cases. One that is most relevant to this paper is a 4-approximation algorithm for Max-QAP when either W or D satisfies the *triangle inequality*, due to Arkin et al. [9]. Another closely related special case is the *dense k*

subgraph problem, where W represents an undirected graph and D corresponds to a k-clique. The best known approximation ratio for general dense k subgraph is n^c, where $c < \frac{1}{3}$ is some universal constant [50]; whereas in the special case of edge weights satisfying triangle inequality, Hassin et al. [82] gave a 2-approximation algorithm.

We now list some other special cases of Max-QAP for which approximation algorithms have been considered. In *capacitated star packing* [80, 8], D consists of a set of vertex disjoint stars, and a 3-approximation algorithm is given in Arkin et al. [8]. In obtaining our algorithm for 0-1 Max-QAP, we use a variant (called Common Star Packing) of the capacitated star packing problem, for which we provide a constant approximation algorithm. *Maximum clustering with given sizes* is the special case of Max-QAP when D is the union of vertex disjoint cliques: assuming that W satisfies triangle inequality, Hassin and Rubinstein [81] gave a $(\frac{1}{1/2-3/k})$-approximation algorithm where k is the smallest cluster size. For maximum clustering under a general W matrix, Feo and Khellaf [54] gave an s-approximation when each clique has size s.

For *dense instances* of the 0-1 Max-QAP problem, there is a PTAS known due to Arora et al. [10]. Dense instances are those where both underlying graphs have $\Omega(n^2)$ edges. In our algorithm for Max-QAP with triangle inequality, we encounter a generalization of a previously studied problem 'Maximum Vertex Cover'. The Max-Vertex-Cover problem is APX-hard, and the best known approximation ratio is $\frac{4}{3} - \epsilon$ for some universal constant $\epsilon > 0$, due to Feige and Langberg [51].

Unlike Max-QAP, the *Minimum Quadratic Assignment* problem remains hard to approximate even when one of the matrices satisfies triangle inequality. Sahni and Gonzales [132] showed that the general case of this problem is hard to approximate to any factor. Queranne [125] later showed that it is NP-hard to approximate this problem to any polynomial factor, even when D corresponds to a line metric. Special cases of minimum quadratic assignment, where D is a metric and W corresponds to certain classes of graphs have been studied in [72, 71, 78].

7.2 General Maximum Quadratic Assignment

The *maximum quadratic assignment* (Max-QAP) problem is the following: given two $n \times n$ symmetric non-negative matrices $W = (w_{i,j})$ and $D = (d_{i,j})$, find a

permutation π of $[n]$ that maximizes:

$$\sum_{i,j\in[n],i\neq j} w_{i,j} \cdot d_{\pi(i),\pi(j)}$$

We obtain an $O(\sqrt{n}\log^2 n)$ approximation algorithm for this problem. A special case of Max-QAP arises when the matrices W and D have only 0-1 entries, we refer to this problem as *0-1* Max-QAP. At the loss of an $O(\log^2 n)$ factor, we first reduce the general Max-QAP to 0-1 Max-QAP: this step uses standard scaling arguments (Lemma 95). Then we obtain an $O(\sqrt{n})$ approximation algorithm for 0-1 Max-QAP (Section 7.2.1).

Lemma 95 (Reduction to 0-1 Max-QAP)**.** *An α approximation algorithm for 0-1* Max-QAP *implies an $O(\alpha \cdot \log^2 n)$ approximation algorithm for general* Max-QAP.

Proof: We assume that neither matrix consists of all zeroes, otherwise the problem is trivial. By scaling matrices W and D, we assume that the maximum entry in both matrices is exactly 1. Let OPTdenote the optimal value of this Max-QAP instance and π the permutation that achieves this; note that $1 \leq \mathsf{OPT} \leq n^2$. We now modify the matrices W and D, by setting to 0 all entries of value smaller than $\frac{1}{2n^2}$. This reduces the optimal value by at most $\frac{1}{2} \leq \frac{\mathsf{OPT}}{2}$, so the optimal value of the modified instance is at least $\frac{\mathsf{OPT}}{2}$.

Now partition the entries of matrix W into $g = \lceil \lg(2n^2) \rceil$ groups so that all entries in the k-th group lie in the range $[\frac{1}{2^k}, \frac{1}{2^{k-1}}]$. Let A_k denote the $n \times n$ 0-1 matrix that has 1s at all positions corresponding to group k entries and 0s everywhere else. Then we have $\frac{1}{2}W \leq \sum_{k=1}^{g} \frac{1}{2^k} A_k \leq W$. By performing an identical operation on D, we can obtain 0-1 matrices $\{B_k\}_{k=1}^{g}$ such that $\frac{1}{2}D \leq \sum_{k=1}^{g} \frac{1}{2^k} B_k \leq D$. For the optimal permutation π, we can express the objective value corresponding to π as:

$$\begin{aligned}\sum_{i,j\in[n],i\neq j} w_{i,j} \cdot d_{\pi(i),\pi(j)} &\leq 4 \cdot \sum_{i,j\in[n],i\neq j} \left(\sum_{k=1}^{g} \frac{1}{2^k} A_k(i,j)\right) \cdot \left(\sum_{l=1}^{g} \frac{1}{2^l} B_l(\pi(i),\pi(j))\right) \\ &= 4 \sum_{k=1}^{g} \sum_{l=1}^{g} \left[\frac{1}{2^{k+l}} \sum_{i,j\in[n],i\neq j} A_k(i,j) \cdot B_l(\pi(i),\pi(j))\right]\end{aligned}$$

The left-hand-side above is at least $\frac{\mathsf{OPT}}{2}$, which implies that there is some pair of values $k, l \in \{1, \cdots, g\}$ such that:

$$\frac{1}{2^{k+l}} \sum_{i,j\in[n],i\neq j} A_k(i,j) \cdot B_l(\pi(i),\pi(j)) \geq \frac{\mathsf{OPT}}{8g^2}$$

Thus if we could approximate 0-1 Max-QAP within an α factor, applying this algorithm to the pair $\langle A_k, B_l \rangle$ gives a permutation σ where:

$$\begin{aligned}\frac{\mathsf{OPT}}{8g^2\alpha} &\leq \frac{1}{2^{k+l}} \sum_{i,j \in [n], i \neq j} A_k(i,j) \cdot B_l(\sigma(i), \sigma(j)) \\ &\leq \sum_{i,j \in [n], i \neq j} w(i,j) \cdot d(\sigma(i), \sigma(j))\end{aligned}$$

The algorithm for Max-QAP would run the α-approximation for 0-1 Max-QAP on all pairs $\langle A_k, B_l \rangle$ (for $1 \leq k, l \leq g$) and return the best permutation found. From the above, it follows that this is an $O(\alpha \log^2 n)$ approximation for general Max-QAP. ■

7.2.1 Algorithm for 0-1 Max-QAP

In this section, we focus on 0-1 Max-QAP and obtain an $O(\sqrt{n})$ approximation algorithm. In this case, the problem can be stated in terms of two n-vertex undirected simple graphs G and H, where the goal is to find a one-to-one mapping of vertices of G to those of H such that the number of common edges is maximized. For an undirected graph G', we let $E(G')$ denote its set of edges. For graph G' on vertex-set $[n]$ and permutation $\pi : [n] \to [n]$, let $\pi(G')$ denote the graph on vertices $[n]$ with edge-set $E(\pi(G')) = \{(i,j) \mid i,j \in [n], (\pi^{-1}(i), \pi^{-1}(j)) \in E(G')\}$. For two undirected graphs G_1 and G_2 both defined on vertex set $[n]$, $G_1 \cap G_2$ denotes the graph on vertices $[n]$ with $E(G_1 \cap G_2) = E(G_1) \cap E(G_2)$. In graph terms, the *0-1* Max-QAP problem is as follows: given undirected graphs G and H each defined on vertex-set $[n]$, find a permutation $\pi : [n] \to [n]$ that maximizes $|E(\pi(G) \cap H)|$.

Let π^* denote the optimal permutation and $O = \pi^*(G) \cap H$ the optimal graph, with $\mathsf{OPT} = |E(O)|$ edges. It is clear that $\mathsf{OPT} \leq |E(G)|, |E(H)|$. Suppose that k is the number of *non-isolated* vertices (i.e. vertices of degree at least one) in the optimal graph O. The final algorithm is the better of two algorithms that we describe next.

Algorithm 1 (Star packing). Before we present the algorithm, we need some definitions. A *star* in graph G' is a subset of edges $S \subseteq E(G')$ that are all incident to some common vertex (called the *center*). The size of a star is the number of edges in it. A *star packing* in an undirected graph is a collection of vertex-disjoint (non-empty) stars. The *size vector* of a star packing is a tuple $\langle c_1, \cdots, c_p \rangle$ where

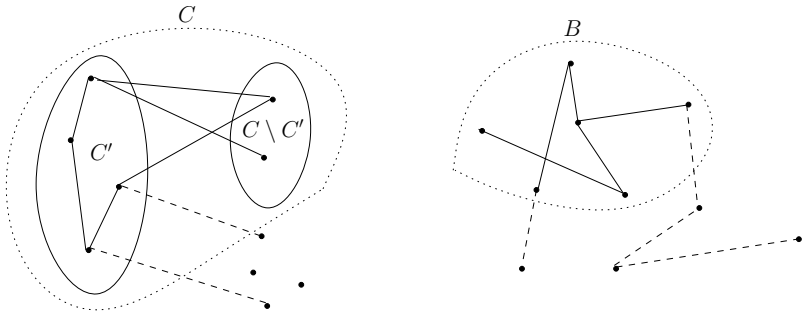
p denotes the number of stars and $c_1, \cdots, c_p$ denote the sizes of all stars in this packing. Given two undirected graphs G and H, a *common star packing* consists of star packings $\mathcal{S}$ in G and $\mathcal{T}$ in H such that $\mathcal{S}$ and $\mathcal{T}$ have identical size-vectors. The *value* of a common star packing given by a pair $(\mathcal{S}, \mathcal{T})$ of star packings is $\sum_{i=1}^{p} c_i$ where $\langle c_1, \cdots, c_p \rangle$ denotes the common size vector of $\mathcal{S}$ and $\mathcal{T}$. In Section 7.2.2, we give a 5-approximation algorithm for computing a maximum value common star packing in two graphs.

The first algorithm involves computing an approximate maximum value common star packing in G and H, using the algorithm in Section 7.2.2. Observe that any common star packing in graphs G and H of value v naturally corresponds to a solution to 0-1 Max-QAP on G, H with v edges: map corresponding stars in the common star packing to each other.

Claim 96. *The solution computed by Algorithm 1 has $\Omega(k)$ edges.*

Proof: Consider the optimal graph O and let F denote any spanning forest in O. Since there are k non-isolated vertices in O, forest F has at least $k/2$ edges. For each tree T in forest F, pick an arbitrary root vertex r and assign a level to each edge $e \in T$, which equals the number of edges on the path from e to r in tree T. Observe that we have two star packings: $\mathcal{S}_e$ consisting of all edges in even levels of trees in F, and $\mathcal{S}_o$ consisting of all edges in odd levels. It is easy to verify that $\mathcal{S}_e$ and $\mathcal{S}_o$ are indeed star packings, and that one of them has at least half the edges in F. Thus we have a star packing in O with at least $\frac{k}{4}$ edges. Finally note that any star packing in O corresponds to a common star packing in G and H. Thus running a 5-approximation algorithm for maximum value common star packing gives a solution with at least $\frac{k}{20}$ edges. ∎

Algorithm 2 (Modified random). The second algorithm involves computing a random mapping between appropriate dense subgraphs of G and H. Recall that the optimal graph O has $n - k$ isolated vertices. Let $V_1, V_2 \subseteq [n]$ respectively denote the set of G-vertices and H-vertices that are mapped to the isolated vertices in O. For an undirected graph G' on vertex-set $[n]$ and subset $U \subseteq [n]$, $G'[U]$ denotes the subgraph of G' induced on U. Observe that either $E(G[V_1]) = \emptyset$ or $E(H[V_2]) = \emptyset$: otherwise, modifying permutation π^* on vertices V_1 would give a solution with more than OPT edges. Suppose that $E(G[V_1]) = \emptyset$ (the case $E(H[V_2]) = \emptyset$ is identical), then graph G has a vertex cover of size k (namely $[n] \setminus V_1$). In this case, we run a 2-approximation algorithm for vertex-cover on G that computes a set $C' \subseteq [n]$ of $2k$ vertices that covers all edges (c.f. Vazirani [150]). Augment the set C' by adding to it, k highest degree vertices from $[n] \setminus C'$ to obtain a set C having $3k$ vertices.



Graph G and its subgraph C Graph H and its subgraph B

Figure 7.1: The subgraphs used in random mapping.

Claim 97. $|E(G[C])| \geq \mathsf{OPT}$.

Proof: Since C' is a vertex cover for G, edges of O can be partitioned into: (1) $E_1 \subseteq E(O)$ edges induced on C', and (2) $E_2 \subseteq E(O)$ edges between C' and $[n] \setminus C'$. By definition, $E_1 \subseteq E(G[C']) \subseteq E(G[C])$. Since all edges of O are induced on k vertices, $|E_2|$ is at most the number of edges incident to the k highest degree vertices in $[n] \setminus C'$ (each of which has its other end-point in C'). Thus the number of edges between C' and $C \setminus C'$ is at least $|E_2|$. Since $|E(G[C'])| \geq |E_1|$ (from above), we have $|E(G[C])| \geq |E_1| + |E_2| = \mathsf{OPT}$. ∎

Next we apply an $O(\frac{n}{k})$-approximation algorithm for *dense k subgraph* (c.f. Feige et al. [50]) to compute a $3k$-vertex subgraph in H having the maximum number of edges. Let this solution be induced on vertex set B. Note that H contains a k-vertex subgraph with at least OPT edges (corresponding to O), so $H[B]$ contains at least $\Omega(\frac{k}{n}) \cdot \mathsf{OPT}$ edges. Figure 7.1 depicts the dense subgraphs in G and H. Algorithm 2 finally returns a *uniformly random* mapping from C to B (other vertices are mapped arbitrarily). Observe that the expected number of common edges in such a random mapping is at least:

$$\begin{aligned}\frac{1}{(3k)^2}|E(G[C])| \cdot |E(H[B])| &= \frac{1}{(3k)^2} \cdot \Omega\left(\frac{k}{n}\right) \cdot \mathsf{OPT}^2 \\ &= \Omega\left(\frac{1}{nk}\right) \cdot \mathsf{OPT}^2\end{aligned}$$

since $G[C]$ has $\Omega(\mathsf{OPT})$ edges and $H[B]$ has $\Omega(\frac{k}{n}) \cdot \mathsf{OPT}$ edges.

Combining algorithms 1 and 2. Finally we output the better of the solutions from Algorithms 1 and 2. The number of edges in this solution is at least $\max\{\Omega(k), \Omega(\frac{1}{nk})\mathsf{OPT}^2\} \geq \sqrt{\Omega(k \cdot \frac{1}{nk} \cdot \mathsf{OPT}^2)} = \Omega(\frac{1}{\sqrt{n}}) \cdot \mathsf{OPT}$. We note that the second algorithm can be easily derandomized using conditional expectation, to give the following.

Theorem 98. *There is an $O(\sqrt{n})$ approximation for 0-1* Max-QAP. *Hence there is an $O(\sqrt{n}\log^2 n)$ approximation algorithm for* Max-QAP.

Remarks. A possible simplification to our algorithm could be just to output the better of maximum common star packing and a uniformly random permutation. However this algorithm achieves only an approximation ratio $\Omega(n^{2/3})$ as shown by an example where both graphs G and H are cliques on $n^{2/3}$ vertices. We note that this simpler algorithm can in fact be shown to achieve a $\Theta(n^{2/3})$ approximation guarantee. Hence in our algorithm, it is important to find appropriate dense subgraphs before applying a random permutation.

A tight example for our algorithm is when G and H are identical $\sqrt{n}$ regular graphs that contain a perfect matching. In this case, both Algorithms 1 and 2 return solutions of value $O(n)$, whereas the optimal value is $\approx n\sqrt{n}$.

7.2.2 Maximum Value Common Star Packing

In this section, we consider the maximum value common star packing problem: given two undirected n-vertex graphs G, H and a number $1 \leq p \leq n$, compute a maximum value common star packing in G and H that consists of exactly p stars. We present a 5-approximation algorithm for this problem. A related problem is *capacitated star packing* [8], where given a single weighted complete graph and a fixed size vector $\overline{s} = \langle s_1, \cdots, s_p \rangle$, the goal is to compute a maximum weight star packing having size vector $\overline{s}$. Arkin et al. [8] gave a 3-approximation algorithm for capacitated star packing. Our algorithm for *common star packing* is based on local-search and is similar to the algorithm in [8].

In Algorithm 1 for 0-1 Max-QAP, we require the maximum value common star packing when the number of stars p is not fixed: for this purpose we run the algorithm for fixed p (described below) over all values of $1 \leq p \leq n$, and pick the best common star packing.

The algorithm for common star packing always maintains a common star packing given by a pair of star packings $\mathcal{S} = \{S_1, \cdots, S_p\}$ in G and $\mathcal{T} = \{T_1, \cdots, T_p\}$ in H,

where S_i and T_i have the same size (for all $1 \leq i \leq p$). For a common star packing $\langle \mathcal{S}, \mathcal{T} \rangle$ as above, we denote by $E(\mathcal{S}) = \sqcup_{i=1}^{p} E(S_i)$ (resp. $E(\mathcal{T}) = \sqcup_{i=1}^{p} E(T_i)$) the set of edges in $\mathcal{S}$ (resp. $\mathcal{T}$). Observe that the value of this common star packing is $|E(\mathcal{S})| = |E(\mathcal{T})|$. We define a bijection $\sigma : E(\mathcal{S}) \to E(\mathcal{T})$ that maps each edge in $E(S_i)$ to some edge in $E(T_i)$ (for every $1 \leq i \leq p$).

Given any common star packing $\langle \mathcal{S}, \mathcal{T} \rangle$, a local move is specified by a tuple $\langle i, x_i, y_i, c_i' \rangle$ where:

- Index $1 \leq i \leq p$ specifies a pair of stars $S_i \in \mathcal{S}$ and $T_i \in \mathcal{T}$.
- Vertices $x_i \in G$ and $y_i \in H$ denote new centers of the stars.
- $0 \leq c_i' \leq n$ denotes the new (common) size of the stars.

Let $v = |E(\mathcal{S})| = |E(\mathcal{T})|$ denote the value of the common star packing. Applying move $\langle i, x_i, y_i, c_i' \rangle$ to $\langle \mathcal{S}, \mathcal{T} \rangle$ involves the following modifications (below, two edges are called independent if they are not incident to a common vertex).

1. Remove edges $E(S_i)$ from $E(\mathcal{S})$ and $E(T_i) = \sigma(E(S_i))$ from $E(\mathcal{T})$.
2. Let $X_i \subseteq E(\mathcal{S})$ denote the edges of $E(\mathcal{S})$ incident to vertex x_i in graph G. Remove X_i from $E(\mathcal{S})$ and $\sigma(X_i)$ from $E(\mathcal{T})$.
3. Let $Y_i \subseteq E(\mathcal{T})$ denote the edges of $E(\mathcal{T})$ incident to vertex y_i in graph H. Remove Y_i from $E(\mathcal{T})$ and $\sigma^{-1}(Y_i)$ from $E(\mathcal{S})$.
4. Let A_i denote any set of c_i' edges incident to vertex x_i in G such that edges in A_i are independent of edges in $E(\mathcal{S})$. If there does not exist such an A_i, the local move *fails*.
5. Let B_i denote any set of c_i' edges incident to vertex y_i in H such that edges in B_i are independent of edges in $E(\mathcal{T})$. If there does not exist such a B_i, the local move *fails*.
6. Add A_i to $E(\mathcal{S})$ and B_i to $E(\mathcal{T})$, and augment bijection σ so that $\sigma(A_i) = B_i$.

In steps 1-3, we only remove corresponding pairs of edges (under bijection σ) from $E(\mathcal{S})$ and $E(\mathcal{T})$. This ensures that after these modifications $\langle \mathcal{S}, \mathcal{T} \rangle$ remains a common star packing. Furthermore, the value of $\langle \mathcal{S}, \mathcal{T} \rangle$ after step 3 is $v - |S_i| - |X_i| - |Y_i|$. If the local move does not fail, then we obtain sets A_i and B_i in steps 4-5.

Note that A_i (resp. B_i) corresponds to a c'_i size star centered at x_i (resp. y_i) in graph G (resp. H). By its definition, star A_i (resp. B_i) can be added to $\mathcal{S}$ (resp. $\mathcal{T}$) to obtain a star packing. Since $|A_i| = |B_i| = c'_i$, after step 6, $\langle \mathcal{S}, \mathcal{T} \rangle$ is a common star packing of value $v + c'_i - |S_i| - |X_i| - |Y_i|$. Finally the local move is said to be *improving* iff it does not fail and the increase in value $c'_i - |S_i| - |X_i| - |Y_i| > 0$.

The algorithm is initialized with $\mathcal{S}$ and $\mathcal{T}$ being zero-value star packings in graphs G and H respectively. Then it performs any sequence of improving local moves, until no further improvement is possible. The value of the solution increases by at least one in each step, and the maximum value of a common star packing is n (number of vertices). So the number of iterations is at most n. The number of local moves at any step is at most n^4, and each local move (steps 1-6) can be easily performed in polynomial time. Thus the entire algorithm runs in polynomial time.

We now argue that any locally optimal solution is a 5-approximate solution. Let $\mathcal{S} = \{S_1, \cdots, S_p\}$ in G and $\mathcal{T} = \{T_1, \cdots, T_p\}$ in H denote the common star packing at a local optimum, where $|S_i| = |T_i| = c_i$ for all $1 \leq i \leq p$. Similarly let $\mathcal{S}^* = \{S^*_1, \cdots, S^*_p\}$ in G and $\mathcal{T}^* = \{T^*_1, \cdots, T^*_p\}$ in H denote the optimal common star packing, where $|S^*_i| = |T^*_i| = c^*_i$ for all $1 \leq i \leq p$. For any vertex $u \in G$ (resp. $v \in H$) define $\tau(\mathcal{S}, u)$ (resp. $\tau(\mathcal{T}, v)$) to be the number of edges of $E(\mathcal{S})$ (resp. $E(\mathcal{T})$) that are incident at u (resp. v). Also define $\mathsf{touch}(S^*_i) := \sum_{u \in S^*_i} \tau(\mathcal{S}, u)$ and $\mathsf{touch}(T^*_i) := \sum_{v \in T^*_i} \tau(\mathcal{T}, v)$. Since $\mathcal{S}$, $\mathcal{S}^*$, $\mathcal{T}$, and $\mathcal{T}^*$ are star packings,

$$\sum_{i=1}^{p} \mathsf{touch}(S^*_i) \leq 2 \sum_{i=1}^{p} c_i \quad \text{and} \quad \sum_{i=1}^{p} \mathsf{touch}(T^*_i) \leq 2 \sum_{i=1}^{p} c_i \tag{7.1}$$

Claim 99. *For any $1 \leq i \leq p$, we have $c^*_i - c_i - \mathsf{touch}(S^*_i) - \mathsf{touch}(T^*_i) \leq 0$.*

Proof: Fix a $1 \leq i \leq p$. Suppose for a contradiction that $c^*_i - c_i - \mathsf{touch}(S^*_i) - \mathsf{touch}(T^*_i) > 0$. Let $x_i \in G$ be the center of star S^*_i and $y_i \in H$ the center of star T^*_i. Let α_i (resp. β_i) denote the number of edges in $\mathcal{S}$ (resp. $\mathcal{T}$) incident to x_i (resp. y_i). Define $c'_i = c^*_i - \mathsf{touch}(S^*_i) - \mathsf{touch}(T^*_i) + \alpha_i + \beta_i$. Consider the local move $\langle i, x_i, y_i, c'_i \rangle$. Observe that when this move is applied to $\langle \mathcal{S}, \mathcal{T} \rangle$, we have $|X_i| = \alpha_i$ and $|Y_i| = \beta_i$ in steps 2-3. Since $\mathsf{touch}(S^*_i)$ is the number of edges of $\mathcal{S}$ incident to star S^*_i and α_i is the number of edges in $\mathcal{S}$ incident to x_i, there are at least $|S^*_i| - (\mathsf{touch}(S^*_i) - \alpha_i)$ edges of star S^*_i that are independent of $E(\mathcal{S})$ in step 4. Since $c'_i \leq c^*_i + \alpha_i - \mathsf{touch}(S^*_i)$, step 4 succeeds. By an identical argument, it follows that step 5 also succeeds. Finally, the increase in value by this local move is:

$$c'_i - c_i - |X_i| - |Y_i| = c^*_i - c_i - \mathsf{touch}(S^*_i) - \mathsf{touch}(T^*_i) > 0$$

But this contradicts the fact that $\langle \mathcal{S}, \mathcal{T} \rangle$ is a local optimum. ∎

Adding the p expressions given by Claim 99 and using Equation (7.1), we have:

$$\sum_{i=1}^{p} c_i^* - \sum_{i=1}^{p} c_i - \sum_{i=1}^{p} \mathsf{touch}(S_i^*) - \sum_{i=1}^{p} \mathsf{touch}(T_i^*) \leq 0$$

$$\implies \sum_{i=1}^{p} c_i^* - 5 \sum_{i=1}^{p} c_i \leq 0$$

Hence any local optimum is a 5-approximate solution.

Theorem 100. *There is a 5-approximation algorithm for maximum value common star packing.*

7.2.3 Asymmetric Maximum Quadratic Assignment

We note that our algorithm for the general Max-QAP problem extends readily to the case when the matrices W, D are asymmetric. The reduction to 0-1 Max-QAP (Lemma 95) clearly holds in the asymmetric case as well. Hence it suffices to consider the directed version of 0-1 Max-QAP, where given two n-vertex directed graphs, the goal is to find a permutation of one graph that maximizes the number of common directed edges. Following the notation in Section 7.2.1, if k denotes the number of non-isolated vertices in the optimal graph O, then Claim 96 implies that O contains a star-packing (in the undirected sense) of size at least $k/4$. It follows that there is either an In-star packing (where edges of each star are directed to its center) or an Out-star packing (where edges of each star are directed away from its center) having size $k/8$. The Common Star Packing algorithm of Section 7.2.2 easily extends to give a constant factor approximation for computing a maximum value common In-star (resp. Out-star) packing in two directed graphs. So Algorithm 1 is guaranteed to find a solution of value $\Omega(k)$.

In Algorithm 2, we consider both graphs as being undirected. Then exactly as before, we obtain two $3k$ vertex subgraphs such that one of them has $\Omega(1)\mathsf{OPT}$ edges and the other $\Omega(\frac{k}{n})\mathsf{OPT}$ edges. Finally observe that a uniformly random mapping of two r-vertex directed graphs having m_1 and m_2 edges results in $\frac{m_1 \cdot m_2}{r^2}$ common *directed* edges in expectation. So Algorithm 2 outputs a solution of value $\Omega(\frac{1}{nk})\mathsf{OPT}^2$. Thus we obtain the following.

Corollary 101. *There is an $O(\sqrt{n} \log^2 n)$ approximation algorithm for asymmetric* Max-QAP.

7.3 Max-QAP under Triangle Inequality

In this section we treat the special case of the Maximum Quadratic Assignment Problem where the entries of matrix D satisfy the triangle inequality, i.e. $d_{ij} + d_{jk} \geq d_{ik}$ for all $i, j, k \in [n]$. Recall that W and D are symmetric.

Let G and H be the complete undirected graphs with edge weights defined by matrices W and D, respectively. Let M be the matching in graph H (where weights satisfy triangle inequality) obtained by the straightforward greedy algorithm: pick the heaviest edge in graph H and delete all incident edges, in the remaining graph choose the heaviest edge and so on. We will call such a matching M *greedy*. Note that $|M| = \lfloor n/2 \rfloor$ since H is a complete graph. Each edge e of graph H is incident with either one or two edges of matching M. For any edge $e \in H$, let $m(e)$ be the edge in M with the largest weight that is incident to e. By the construction of the greedy matching M, we have $d_{m(e)} \geq d_e$ for all $e \in H$.

We consider the following modification of the given Max-QAP instance, which we call the *auxiliary problem*. Find a permutation π of $[n]$ that maximizes:

$$\sum_{i,j \in [n], i \neq j} w_{i,j} \cdot d_{m(\pi(i),\pi(j))}, \tag{7.2}$$

i.e. the weight of each edge (i, j) in graph G is multiplied by the weight of edge $m(\pi(i), \pi(j))$ in graph H (which is incident to the edge $(\pi(i), \pi(j))$ in H). Let OPT^* be the value of the optimal solution to the Max-QAP instance, and let Aux^* be the optimal value of the auxiliary problem (7.2). We first prove a simple lemma based on triangle inequality.

Lemma 102. $\mathsf{Aux}^* \geq \mathsf{OPT}^* \geq \mathsf{Aux}^*/2$

Proof: Since $d_{m(e)} \geq d_e$ for all edges $e \in H$, we obtain the first inequality $\mathsf{Aux}^* \geq \mathsf{OPT}^*$.

Consider now an optimal solution (permutation) σ for the auxiliary problem (7.2), that maps vertices of G to those of H. Let σ' denote the random permutation where we swap assignments along each edge of matching M with probability $1/2$. More precisely, consider an edge $(u, v) \in M$ such that i and j are the two vertices of graph G mapped to the endpoints of this, i.e. $u = \sigma(i)$ and $v = \sigma(j)$. We set $\sigma'(i) = u$, $\sigma'(j) = v$ with probability $1/2$, and $\sigma'(i) = v$, $\sigma'(j) = u$ with probability $1/2$. This process is repeated independently for all edges of the greedy matching M.

We now prove that the expected value of the original Max-QAP instance on the random permutation σ' is at least $\mathsf{Aux}^*/2$ that would imply the second inequality of the lemma. Consider an edge (i,j) in graph G, and let $\sigma(i)=u$ and $\sigma(j)=v$. If $(u,v)\in M$, then the expectation of the term corresponding to (i,j) in the objective function of Max-QAP on permutation σ' is exactly $w_{ij}d_{u,v}$. If $(u,v)\notin M$ and both u and v are incident to edges from M, then let $\bar{u}$ and $\bar{v}$ be the other endpoints of edges from M incident to u and v, i.e. $(u,\bar{u})\in M$ and $(v,\bar{v})\in M$. In this case, by triangle inequality the expectation of the objective function term corresponding to (i,j) on permutation σ' is exactly:

$$\begin{aligned} w_{ij}(d_{u,v}+d_{u,\bar{v}}+d_{\bar{u},v}+d_{\bar{u},\bar{v}})/4 &\geq w_{ij}\frac{\max\{d_{u,\bar{u}},d_{v,\bar{v}}\}}{2} \\ &= w_{ij}\cdot d_{m(\sigma(i),\sigma(j))}/2 \end{aligned}$$

Analogously, if vertex u or v (say v) is the single vertex of H that is not incident to any edge of the greedy matching M and $(u,\bar{u})\in M$, then the expectation of the objective function term corresponding to (i,j) on permutation σ' is:

$$w_{ij}(d_{u,v}+d_{\bar{u},v})/2 \geq w_{ij}\cdot d_{u,\bar{u}}/2 = w_{ij}\cdot d_{m(\sigma(i),\sigma(j))}/2.$$

Summing up the contribution to the Max-QAP objective over all edges $(i,j)\in G$, the expected value of permutation σ' is at least $\frac{\mathsf{Aux}^*}{2}$ which implies $\mathsf{OPT}^*\geq \mathsf{Aux}^*/2$. ■

7.3.1 Algorithm for the auxiliary problem

In the rest of the section, we will show how to construct a $(1-\frac{1}{e})$ approximation algorithm for the auxiliary problem. We consider the following more general problem. The input is an undirected edge-weighted graph $G=(V,E,w)$ with nonnegative edge weights $w_e\geq 0$ for $e\in E$ and nonnegative numbers $\{\Delta_i\}_{i=1}^n$. The goal is to find a permutation π of vertices of graph G that maximizes objective:

$$2\sum_{(i,j)\in E} w_{ij}\cdot\left(\sum_{p=\min\{\pi_i,\pi_j\}}^{n}\Delta_p\right) \tag{7.3}$$

We first reduce the auxiliary problem (7.2) to one of the above form (7.3). The weighted graph G is the complete graph on vertex set $V=[n]$ with edge-weights W.

Let $l = \lfloor n/2 \rfloor$ and $D_1 \geq D_2 \geq \cdots \geq D_l$ be the edge-weights of greedy matching M. We set $\Delta_{2q} = D_q - D_{q+1}$ for all $1 \leq q \leq l$ (here $D_{l+1} = 0$), and all other Δs are set to 0. We also renumber vertices in graph H, in the auxiliary problem (7.2), so that the edges in the greedy matching M are chosen in the order $(1,2),\ (3,4), \cdots,\ (2l-1, 2l)$. Now for any permutation π and vertices i, j in G, we have

$$d_{m(\pi(i),\pi(j))} = \sum_{p=\min\{\pi(i),\pi(j)\}}^{n} \Delta_p$$

Hence objective (7.3) equals objective (7.2) for every permutation (note that (7.2) sums over unordered pairs i, j whereas (7.3) sums over ordered pairs). In the following, we obtain an approximation algorithm for Problem (7.3).

Problem (7.3) generalizes the *Maximum Vertex Cover* problem where, given an edge-weighted undirected graph and a number k, the goal is to find k vertices that cover the maximum weight of edges. The maximum vertex cover problem is APX-hard [123] and the best known approximation ratio is $\approx \frac{3}{4}$ [51]. We present a $(1 - \frac{1}{e})$ approximation algorithm for problem (7.3) using a natural LP relaxation. In the following, x-variables are assignment variables mapping vertices to positions, and each variable z_{ijs} denotes whether either of vertices $i, j \in V$ is mapped to some position in $\{1, \cdots, s\}$ (where $s \in [n]$).

$$\begin{aligned}
\max \quad & 2 \cdot \sum_{(i,j)\in E} w_{ij} \sum_{s=1}^{n} \Delta_s z_{ijs}, \\
& z_{ijs} \leq \sum_{t=1}^{s} x_{it} + \sum_{t=1}^{s} x_{jt},\ \forall (i,j) \in E,\ \forall s = 1, \ldots, n && (7.4)\\
& z_{ijs} \leq 1, \qquad \forall (i,j) \in E,\ \forall s = 1, \ldots, n && (7.5)\\
& \sum_{i \in V} x_{it} = 1, \qquad \forall t = 1, \ldots, n && (7.6)\\
& \sum_{t=1}^{n} x_{it} = 1, \qquad \forall i \in V && (7.7)\\
& x_{it} \geq 0, \qquad \forall i \in V,\ \forall t = 1, \ldots, n && (7.8)\\
& z_{ijs} \geq 0, \qquad \forall (i,j) \in E,\ \forall s = 1, \ldots, n && (7.9)
\end{aligned}$$

Our algorithm is quite a natural randomized rounding of the optimal solution (x^*, z^*) of the above linear program. For each position $t = 1, \ldots, n$ we treat constraint (7.6) as a density function and choose a vertex $i \in V$ at random according

to this distribution, to assign to position t. After that, each position $t = 1, \ldots, n$ has one chosen vertex. If a vertex is chosen by many positions then we assign it to the *earliest* one. The vertices not chosen by any position in the previous step of the algorithm are assigned to arbitrary empty positions.

We now derive the expected performance guarantee of the algorithm. For each edge $(i, j) \in E$ we estimate its contribution to the objective function:

$$
\begin{aligned}
& w_{ij} \sum_{s=1}^{n} \Delta_s \cdot Pr(i \text{ or } j \text{ is assigned a position } \leq s) \\
&= w_{ij} \sum_{s=1}^{n} \Delta_s \cdot \left(1 - \prod_{t=1}^{s} (1 - x^*_{it} - x^*_{jt})\right) && (7.10) \\
&\geq w_{ij} \sum_{s=1}^{n} \Delta_s \left(1 - e^{-\sum_{t=1}^{s}(x^*_{it} + x^*_{jt})}\right) && (7.11) \\
&\geq w_{ij} \sum_{s=1}^{n} \Delta_s \left(1 - \frac{1}{e}\right) \cdot \min\left\{\sum_{t=1}^{s} (x^*_{it} + x^*_{jt}), 1\right\} && (7.12) \\
&= \left(1 - \frac{1}{e}\right) w_{ij} \sum_{s=1}^{n} \Delta_s z^*_{ijs}
\end{aligned}
$$

Inequality (7.11) follows from the fact that $1 + x \leq e^x$ for all $x \in \mathbb{R}$, and inequality (7.12) from $1 - e^{-x} \geq (1 - \frac{1}{e})x$ for $0 \leq x \leq 1$. Therefore, the total expected objective function value of the rounded solution is at least $1 - \frac{1}{e}$ times the optimal value of the linear programming relaxation. Combined with Lemma 102, we have the following.

Theorem 103. *There is a* $\frac{2e}{e-1}$ *approximation algorithm for* Max-QAP *with triangle inequality.*

Derandomization. The above randomized rounding algorithm can be derandomized using *conditional expectations* since we have an exact expression for the expected objective value (7.10). Similarly, the algorithm in Lemma 102, that obtains a solution to Max-QAP from one for problem (7.2) can be easily derandomized. Hence we obtain a deterministic algorithm in Theorem 103.

7.4 Some Remarks on an LP Relaxation for Max-QAP

Consider the following integer program for Max-QAP. We have assignment variables $x_{i,p}$ between vertices of the two graphs, and variables $y_{i,p,j,q}$ denote whether "i maps to p *and* j maps to q". The LP relaxation $\mathcal{LP}$ is obtained by dropping the integrality condition on variables, and is given below.

$$\begin{aligned}
\max \sum_{i,j\in[n],i\neq j} w_{ij} \sum_{p,q\in[n],p\neq q} & d_{pq}\cdot y_{i,p,j,q}, \\
\sum_{i=1}^{n} x_{i,p} &= 1, && \forall\ 1\le p\le n \\
\sum_{p=1}^{n} x_{i,p} &= 1, && \forall\ 1\le i\le n \\
\sum_{i=1}^{n} y_{i,p,j,q} &= x_{j,q}, && \forall\ 1\le p,j,q\le n \\
\sum_{p=1}^{n} y_{i,p,j,q} &= x_{j,q}, && \forall\ 1\le i,j,q\le n \\
\sum_{j=1}^{n} y_{i,p,j,q} &= x_{i,p}, && \forall\ 1\le i,p,q\le n \\
\sum_{q=1}^{n} y_{i,p,j,q} &= x_{i,p}, && \forall\ 1\le i,j,p\le n \\
x_{i,p} &\ge 0, && \forall\ 1\le i,p\le n \\
y_{i,p,j,q} &\ge 0, && \forall\ 1\le i,p,j,q\le n.
\end{aligned}$$

General Max-QAP**.** The *dense k subgraph* problem is the special case of Max-QAP when matrix D is the incidence matrix of a k-clique (i.e. $d_{pq}=1$ if $1\le p,q\le k$, and $d_{pq}=0$ otherwise), and W is the incidence matrix of the input graph. We note that in the case of dense k subgraph, this LP can be shown to have an integrality gap of $O(\sqrt{n})$: $\mathcal{LP}$ is at least as good as the standard LP for dense k subgraph, which has integrality gap $\approx \frac{n}{k}$ (due to Goemans); in addition $\mathcal{LP}$ cannot have value larger than $\min\{k^2, m\}$ (where m is number of edges in the input graph), so its integrality gap is at most k. The following example shows that this is nearly tight. Consider $k=\sqrt{n}$, and a random k-regular graph (see Wormald [154] for such models) corresponding to W. With high probability, the optimal value of the integer

program can be bounded by $O(k \log n)$. However, the LP solution when all $x_{i,p} = \frac{1}{n}$ can be shown to have objective value $\Omega(k^2)$. This gives an $\Omega(\frac{\sqrt{n}}{\log n})$ lower bound on the integrality gap of $\mathcal{LP}$.

Triangle inequality Max-QAP**.** We also observe that for Max-QAP with triangle inequality, our approximation algorithm (Section 7.3) implies the same upper bound on the integrality gap for the above LP relaxation. Given an optimal solution (x, y) to $\mathcal{LP}$ for Max-QAP, we induce a solution $(\tilde{x}, z)$ for the LP relaxation for problem (7.3), where $\tilde{x}_{it} = x_{it}$ for all $i, t \in [n]$ and $z_{ijs} = \sum_{p=1}^{s}(\sum_{q=1}^{n} y_{ipjq} + \sum_{q=s+1}^{n} y_{iqjp})$ for all $1 \leq i < j \leq n$, $s \in [n]$. One can easily check that $z_{ijs} \leq 1$ and $z_{ijs} \leq \sum_{t=1}^{s}(\tilde{x}_{it} + \tilde{x}_{jt})$:

$$z_{ijs} = \sum_{p=1}^{s}\sum_{q=1}^{n} y_{ipjq} + \sum_{q=s+1}^{n}\sum_{p=1}^{s} y_{iqjp} \leq \sum_{p=1}^{n}\sum_{q=1}^{n} y_{ipjq} = 1,$$

and,

$$\begin{aligned} z_{ijs} &= \sum_{p=1}^{s}\sum_{q=1}^{n} y_{ipjq} + \sum_{p=1}^{s}\sum_{q=s+1}^{n} y_{iqjp} \\ &\leq \sum_{p=1}^{s} x_{ip} + \sum_{p=1}^{s}\sum_{q=1}^{n} y_{iqjp} \\ &= \sum_{p=1}^{s}(x_{ip} + x_{jp}) \end{aligned}$$

So $(\tilde{x}, z)$ is indeed feasible. As for the first inequality in Lemma 102, we can argue that the LP-objective (in problem (7.3)) of $(\tilde{x}, z)$ is at least the LP-objective of (x, y) in $\mathcal{LP}$. Then the LP-rounding algorithm for problem (7.3) together with the second part of Lemma 102 implies that the integrality gap of the above LP for Max-QAP is at most $\frac{2e}{e-1}$, when one of the matrices satisfies triangle inequality.

Credits: The results in this chapter are from *"On the Maximum Quadratic Assignment Problem"* [114], obtained jointly with Maxim Sviridenko.

Bibliography

[1] W. P. Adams and T. A. Johnson. Improved Linear Programming-based Lower Bounds for the Quadratic Assignment Problem. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, volume 16, pages 43–77, 1994.

[2] Micah Adler and Brent Heeringa. Approximating Optimal Binary Decision Trees. In *Proceedings of the 11th International Workshop on Approximation, Randomization and Combinatorial Optimization*, pages 1–9, 2008.

[3] A. Ageev and M. Sviridenko. Pipage rounding: a new method of constructing algorithms with proven performance guarantee. *Journal of Combinatorial Optimization*, 8:307–328, 2004.

[4] Ajit Agrawal, Philip Klein, and R. Ravi. When trees collide: an approximation algorithm for the generalized steiner problem on networks. *SIAM Journal on Computing*, 24(3):440–456, 1995.

[5] K. Altinkemer and B. Gavish. Heuristics for unequal weight delivery problems with a fixed error guarantee. *Operations Research Letters*, 6:149–158, 1987.

[6] K. Altinkemer and B. Gavish. Heuristics for delivery problems with constant error guarantees. *Transportation Research*, 24:294–297, 1990.

[7] David L. Applegate, Robert E. Bixby, Vasek Chvátal, and William J. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2006.

[8] E. Arkin, R. Hassin, S. Rubinstein, and M. Sviridenko. Approximations for Maximum Transportation Problem with Permutable Supply Vector and other Capacitated Star Packing Problems. *Algorithmica*, 39:175–187, 2004.

[9] E.M. Arkin, R. Hassin, and M. Sviridenko. Approximating the maximum quadratic assignment problem. *Information Processing Letters*, 77:13–16, 2001.

[10] S. Arora, A. Frieze, and H. Kaplan. A new rounding procedure for the assignment problem with applications to dense graph arrangement problems. *Mathematical Programming*, 92(1):1–36, 2002.

[11] Sanjeev Arora. Polynomial-time Approximation Schemes for Euclidean TSP and other Geometric Problems. *Journal of the ACM*, 45(5):753–782, 1998.

[12] Brenda S. Baker. Approximation Algorithms for NP-Complete Problems on Planar Graphs. *Journal of ACM*, 41:153–180, 1994.

[13] N. Bansal, A. Blum, S. Chawla, and A. Meyerson. Approximation Algorithms for Deadline-TSP and Vehicle Routing with Time Windows. pages 166–174, 2004.

[14] D. Bertsimas, P. Chervi, and M. Peterson. Computational approaches to stochastic vehicle routing problems. *Transportation Science*, 29:342–352, 1995.

[15] D. Bertsimas, P. Jaillet, and A.R. Odoni. A priori optimization. *Operations Research*, 38(6):1019 – 1033, 1990.

[16] Dimitris J. Bertsimas. A vehicle routing problem with stochastic demand. *Operations Research*, 40(3):574–585, 1992.

[17] 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*, 37(2):653–670, 2007.

[18] Avrim Blum, Prasad Chalasani, Don Coppersmith, William R. Pulleyblank, Prabhakar Raghavan, and Madhu Sudan. The minimum latency problem. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pages 163–171, 1994.

[19] A. Bompadre, M. Dror, and J.B. Orlin. Improved bounds for vehicle routing solutions. *Discrete Optimization*, 3(4):299–316, 2006.

[20] R.E. Burkard, E. Cela, P. Pardalos, and L.S. Pitsoulis. The quadratic assignment problem. In *Handbook of Combinatorial Optimization, D.Z. Du, P.M. Pardalos (Eds.)*, volume 3, pages 241–339. Kluwer Academic Publishers, 1998.

[21] Costas Busch, Ryan LaFortune, and Srikanta Tirthapura. Improved sparse covers for graphs excluding a fixed minor. In *Proceedings of the 26th Annual ACM Symposium on Principles of Distributed Computing*, pages 61–70, 2007.

[22] Eranda Cela. *The Quadratic Assignment Problem: Theory and Algorithms*. Springer, 1998.

[23] S. Chakrabarti, C. Phillips, A. Schulz, D. Shmoys, C. Stein, and J. Wein. Improved Scheduling Algorithms for Minsum Criteria. In *Proceedings of the 23rd International Colloquium on Automata, Languages, and Programming*, pages 646–657, 1996.

[24] V. T. Chakravarthy, V. Pandit, S. Roy, P. Awasthi, and M. Mohania. Decision Trees for Entity Identification: Approximation Algorithms and Hardness Results. In *Proceedings of the ACM-SIGMOD Symposium on Principles of Database Systems*, 2007.

[25] I-M. Chao. A tabu search method for the truck and trailer routing problem. *Computer and Operations Research*, 29:469–488, 2002.

[26] Moses Charikar, Chandra Chekuri, Ashish Goel, and Sudipto Guha. Rounding via trees: deterministic approximation algorithms for group Steiner trees and k median. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 114–123, 1998.

[27] Moses Charikar, Chandra Chekuri, To yat Cheung, Zuo Dai, Ashish Goel, Sudipto Guha, and Ming Li. Approximation algorithms for directed Steiner problems. *Journal of Algorithms*, 33:73–91, 1999.

[28] Moses Charikar and Balaji Raghavachari. The Finite Capacity Dial-A-Ride Problem. In *Proceedings of 39th Annual Symposium on Foundations of Computer Science*, pages 458–467, 1998.

[29] Kamalika Chaudhuri, Brighten Godfrey, Satish Rao, and Kunal Talwar. Paths, trees, and minimum latency tours. pages 36–45, 2003.

[30] Chandra Chekuri, Nitish Korula, and Martin Pal. Improved Algorithms for Orienteering and Related Problems. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete algorithms*, pages 661–670, 2008.

[31] Chandra Chekuri and Martin Pal. A recursive greedy algorithm for walks in directed graphs. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 245–253, 2005.

[32] Chandra Chekuri and Martin Pal. An $O(\log\ n)$ Approximation Ratio for the Asymmetric Traveling Salesman Path Problem. *Theory of Computing*, 3:197–209, 2007.

[33] B. Chen, C. Potts, and G. Woeginger. A review of machine scheduling: complexity, algorithms and approximability. *Handbook of combinatorial optimization*, 3:21–169, 1998.

[34] N. Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. *GSIA, CMU-Report 388*, 1977.

[35] J. Chuzhoy, S. Guha, E. Halperin, G. Kortsarz, S. Khanna, R. Krauthgamer, and S. Naor. Asymmetric k-center is log^*n-hard to Approximate. *Journal of the ACM*, 52(4):538–551, 2005.

[36] V. Chvatal. A greedy heuristic for the set covering problem. *Mathematics of Operations Research*, 4:233–235, 1979.

[37] M. Conforti and G. Cornuejols. Submodular set functions, matroids and the greedy algorithm: tight worst-case bounds and some generalizations of the Rado-Edmonds theorem. *Discrete Applied Mathematics*, 7:257–275, 1984.

[38] R. Conway, W. Maxwell, and L. Miller. *Theory of scheduling*. Addison-Wesley Publishing Co.

[39] Jean-Francois Cordeau and Gilbert Laporte. The Dial-a-Ride Problem (DARP): Variants, modeling issues and algorithms. *4OR: A Quarterly Journal of Operations Research*, 1(2), 2003.

[40] A. Czumaj and C. Scheideler. A New Algorithmic Approach to the General Lovasz Local Lemma with Applications to Scheduling and Satisfiability Problems. In *Proceedings of the 32nd Annual Symposium on Theory of Computing*.

[41] Willem E. de Paepe, Jan Karel Lenstra, Jiri Sgall, Ren A. Sitters, and Leen Stougie. Computer-Aided Complexity Classification of Dial-a-Ride Problems . *Informs Journal on Computing*, 16(2):120–132, 2004.

[42] M. Dror, G. Laporte, and P. Trudeau. Vehicle routing with stochastic demands: properties and solution frameworks. *Transportation Science*, 23:166176, 1989.

[43] Moshe Dror. Vehicle Routing with Stochastic Demands: Models & Computational Methods. In *Modeling Uncertainty: International Series In Operations Research & Management Science*, volume 46(8), pages 625–649. Springer.

[44] G. Even, N. Garg, J. Könemann, R. Ravi, and A. Sinha. Covering Graphs Using Trees and Stars. *Operations Research Letters*, 32(4):309–315.

[45] Jittat Fakcharoenphol, Chris Harrelson, and Satish Rao. The k-traveling repairmen problem. *ACM Transactions on Algorithms*, 3(4), 2007.

[46] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *Journal of Computer and System Sciences*, 69(3):485–497, 2004.

[47] U. Feige and J.R. Lee. An improved approximation ratio for the minimum linear arrangement problem. *Information Processing Letters*, 101(1):26–29, 2007.

[48] U. Feige and C. Scheideler. Improved bounds for acyclic job shop scheduling. In *Proceedings of the 30th Annual Symposium on Theory of Computing*.

[49] Uriel Feige. A threshold of $\ln n$ for set cover. *Journal of the ACM*, 45:634–652, 1998.

[50] Uriel Feige, Guy Kortsarz, and David Peleg. The Dense k-Subgraph Problem. *Algorithmica*, 29(3):410–421, 2001.

[51] Uriel Feige and Michael Langberg. Approximation Algorithms for Maximization Problems Arising in Graph Partitioning. *Journal of Algorithms*, 41(2):174–211, 2001.

[52] Uriel Feige, David Peleg, and Guy Kortsarz. The Dense k -Subgraph Problem. *Algorithmica*, 29(3):410–421, 2001.

[53] Uriel Feige and Mohit Singh. Improved Approximation Ratios for Traveling Salesperson Tours and Paths in Directed Graphs. In *Proceedings of the 10th International Workshop on Approximation, Randomization and Combinatorial Optimization*, pages 104–118, 2007.

[54] T. Feo and M. Khellaf. A class of bounded approximation algorithms for graph partitioning . *Networks*, 20:181–195, 1990.

[55] A. Fishkin, K. Jansen, and M. Mastrolilli. On minimizing average weighted completion time: a PTAS for the job shop problem with release dates. In *Proceedings of the 14th International Symposium on Algorithms and Computation*, pages 319–328, 2003.

[56] J. Framinan, J. Gupta, and R. Leisten. A review and classification of heuristics for permutation flow-shop scheduling with makespan objective. *Journal of the Operational Research Society*, 55:1243–1255, 2004.

[57] A. Frank. On Connectivity properties of Eulerian digraphs. *Annals of Discrete Mathematics*, 41, 1989.

[58] Greg N. Frederickson, Matthew S. Hecht, and Chul E. Kim. Approximation algorithms for some routing problems. *SIAM Journal on Computing*, 7(2):178–193, 1978.

[59] A. Frieze. On the length of the longest monotone subsequence of a random permutation . *The Annals of Applied Probability*, 1(2):301–305, 1991.

[60] A. Frieze, G. Galbiati, and F. Maffioli. On the worst-case performance of some algorithms for the asymmetric travelling salesman problem. *Networks*, 12:23–39, 1982.

[61] N. Garg, G. Kojevod, and R. Ravi. A Polylogarithmic Approximation Algorithm for the Group Steiner Tree Problem. *Journal of Algorithms*, 37(1):66–84, 2000.

[62] Naveen Garg. A 3-Approximation for the Minimum Tree Spanning k Vertices. In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, pages 302–309, 1996.

[63] Naveen Garg. Saving an epsilon: a 2-approximation for the k-MST problem in graphs. In *Proceedings of the 37th annual ACM Symposium on Theory of computing*, pages 396–402, 2005.

[64] Naveen Garg, Anupam Gupta, Stefano Leonardi, and Piotr Sankowski. Stochastic analyses for online combinatorial optimization problems. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete algorithms*, pages 942–951, 2008.

[65] Inge Li Gørtz. Topics in Algorithms: Data Structures on Trees and Approximation Algorithms on Graphs. *Ph.D. Thesis, IT University of Copenhagen*, 2005.

[66] Inge Li Gørtz. Hardness of Preemptive Finite Capacity Dial-a-Ride. In *Proceedings of the 9th International Workshop on Approximation, Randomization and Combinatorial Optimization*, pages 200–211, 2006.

[67] Michel X. Goemans and Dimitris J. Bertsimas. Survivable networks, linear programming relaxations and the parsimonious property. *Mathematical Programming*, 60:145–166, 1993.

[68] Michel X. Goemans and David P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995.

[69] Anupam Gupta. Steiner points in tree metrics don't (really) help. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete algorithms*, pages 220–227, 2001.

[70] Anupam Gupta, MohammadTaghi Hajiaghayi, Viswanath Nagarajan, and R Ravi. Dial a Ride from k-forest. *To Appear: ACM Transactions on Algorithms (Preliminary version in European Syposium on Algorithms, 2007)*.

[71] N. Guttmann-Beck and R. Hassin. Approximation algorithms for min-sum p-clustering. *Discrete Applied Mathematics*, 89:125–142, 1998.

[72] N. Guttmann-Beck and R. Hassin. Approximation algorithms for minimum tree tree partition. *Discrete Applied Mathematics*, 87:117–137, 1998.

[73] M. Haimovich and A. H. G. Rinnooy Kan. Bounds and heuristics for capacitated routing problems. *Mathematics of Operations Research*, 10:527–542, 1985.

[74] Mohammad Taghi Hajiaghayi and Kamal Jain. The prize-collecting generalized steiner tree problem via a new approach of primal-dual schema. In

Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete algorithm, pages 631–640, 2006.

[75] L.A. Hall, A.S. Schulz, D.B. Shmoys, and J. Wein. Scheduling to Minimize Average Completion Time: Off–Line and On–Line Approximation Algorithms. *Mathematics of Operations Research*, 22:513–544, 1997.

[76] L.A. Hall, D.B. Shmoys, and J. Wein. Scheduling to Minimize Average Completion Time: Off–line and On–line Algorithms. In *Proceedings of the 7th annual ACM-SIAM Symposium on Discrete Algorithms*, pages 142–151, 1996.

[77] Eran Halperin and Robert Krauthgamer. Polylogarithmic inapproximability. In *Proceedings of the 35th Annual Symposium on Theory of Computing*, pages 585–594, 2003.

[78] R. Hassin, A. Levin, and M. Sviridenko. Approximating the minimum quadratic assignment problems. *Manuscript*, 2008.

[79] R. Hassin and S. Rubinstein. Approximation algorithms for maximum linear arrangement. *Information Processing Letters*, 80:171–177, 2001.

[80] R. Hassin and S. Rubinstein. Robust matchings. *SIAM Journal of Discrete Mathematics*, 15:530–537, 2002.

[81] R. Hassin and S. Rubinstein. An improved approximation algorithm for the metric maximum clustering problem with given cluster sizes. *Information Processing Letters*, 98:92–95, 2006.

[82] R. Hassin, S. Rubinstein, and A. Tamir. Approximation Algorithms for Maximum Dispersion. *Operations Research Letters*, 21:133–137, 1997.

[83] J. Hastad. Clique is hard to approximate within $n^{1-\varepsilon}$. *Acta Mathematica*, 182:105–142, 1999.

[84] M. Held and R.M. Karp. The travelling salesman problem and minimum spanning trees. *Operations Research*, 18:1138–1162, 1970.

[85] D. Hochbaum and W. Maass. Approximation Schemes for Covering and Packing Problems in Image Processing and VLSI. *Journal of ACM*, 32:130–136, 1985.

[86] D.S. Hochbaum and D.B. Shmoys. A unified approach to approximation algorithms for bottleneck problems. *Journal of the ACM*, 33:533–550, 1986.

[87] D.S. Hochbaum and D.B. Shmoys. Using dual approximation algorithms for scheduling problems: theoretical and practical results. *Journal of the ACM*, 34:144–162, 1987.

[88] W.L. Hsu and G.L. Nemhauser. Easy and hard bottleneck location problems. *Discrete Applied Mathematics*, 1:209–216, 1979.

[89] O.H. Ibarra and C.E. Kim. Fast approximation algorithms for the knapsack and subset sum problems. *Journal of the ACM*, 22:463–468, 1976.

[90] B. Jackson. Some remarks on arc-connectivity, vertex splitting, and orientation in digraphs. *Journal of Graph Theory*, 12(3):429–436, 1988.

[91] Patrick Jaillet. A priori solution of a travelling salesman problem in which a random subset of the customers are visited. *Operations Research*, 36(6):929 – 936, 1988.

[92] D.S. Johnson. Approximation algorithms for combinatorial optimization problems. *Journal of Computer and System Sciences*, 9:256–278, 1974.

[93] R.M. Karp. A characterization of the minimum cycle mean in a digraph. *Discrete Mathematics*, 23(3):309–311, 1978.

[94] S. Khuller, A. Moss, and J. Naor. The budgeted maximum coverage problem. *Information Processing Letters*, 70:39–45, 1999.

[95] Samir Khuller, Balaji Raghavachari, and Neal Young. Balancing minimum spanning and shortest path trees. In *Proceedings of the 4th Annual ACM-SIAM Symposium on Discrete algorithms*, pages 243–250, 1993.

[96] K.Jansen, R.Solis-Oba, and M. Sviridenko. Makespan Minimization in Job Shops: a Linear Time Approximation Scheme. *SIAM Journal of Discrete Mathematics*, 16:288–300, 2003.

[97] Philip Klein, Serge A. Plotkin, and Satish Rao. Excluded minors, network decomposition, and multicommodity flow. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, pages 682–690, 1993.

[98] Jon Kleinberg and David Williamson. Unpublished note. 1998.

[99] S. Rao Kosaraju, Teresa M. Przytycka, and Ryan S. Borgstrom. On an Optimal Split Tree Problem. In *Proceedings of the 6th International Workshop on Algorithms and Data Structures*, pages 157–168, 1999.

[100] S. Krumke, J. Rambau, and S. Weider. An approximation algorithm for the nonpreemptive capacitated dial-a-ride problem. *Preprint 00-53, Konrad-Zuse-Zentrum fr Informationstechnik Berlin*, 2000.

[101] Fumei Lam and Alantha Newman. Traveling salesman path problems. *Mathematical Programming*, 113(1):39–59, 2008.

[102] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys. Sequencing and scheduling: Algorithms and complexity. In *Handbook in Operations Research and Management Science*, volume 4, pages 445–522. North-Holland, 1993.

[103] F. Lazebnik, V.A. Ustimenko, and A.J. Woldar. A New Series of Dense Graphs of High Girth. *Bulletin of the AMS*, 32(1):73–79, 1995.

[104] B.F. Logan and L.A. Shepp. A Variational Problem for Random Young Tableaux. *Advances in Mathematics*, 26:206–222, 1977.

[105] E.M. Loilola, N.M.M. De Abreu, P.O. Boaventura-Netto, P.M. Hahn, and T. Querido. A survey for the quadratic assignment problem (Invited Review). *European Journal of Operational Research*, 176:657–690, 2006.

[106] L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13:383–390, 1975.

[107] W. Mader. Construction of all n-fold edge-connected digraphs (German). *European Journal of Combinatorics*, 3:63–67, 1982.

[108] J.S.B. Mitchell. Guillotine Subdivisions Approximate Polygonal Subdivisions: Part II – A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems. *SIAM Journal on Computing*, 28(4):1298–1309, 1999.

[109] S. Mitrović-Minić and G. Laporte. The Pickup and Delivery Problem with Time Windows and Transshipment. *Information Systems and Operational Research*, 44:217–227, 2006.

[110] C. Mues and S. Pickl. Transshipment and time windows in vehicle routing. In *Proceedings of the 8th International Symposium on Parallel Architectures, Algorithms and Networks*, pages 113–119, 2005.

[111] Viswanath Nagarajan and R. Ravi. Poly-logarithmic approximation algorithms for directed vehicle routing problems. In *Proceedings of the 10th International Workshop on Approximation, Randomization and Combinatorial Optimization*, pages 257–270, 2007.

[112] Viswanath Nagarajan and R. Ravi. The Directed Minimum Latency Problem. In *Proceedings of the 11th International Workshop on Approximation, Randomization and Combinatorial Optimization*, pages 193–206, 2008.

[113] Viswanath Nagarajan and Maxim Sviridenko. Tight Bounds for Permutation Flowshop Scheduling. *To Appear: Mathematics of Operations Research (Preliminary version in Integer Programming and Combinatorial Optimization, 2008)*.

[114] Viswanath Nagarajan and Maxim Sviridenko. On the Maximum Quadratic Assignment Problem. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete algorithms*, pages 516–524, 2009.

[115] Y. Nakao and H. Nagamochi. Worst case analysis for pickup and delivery problems with transfer. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E91-A(9), 2008.

[116] M. Nawaz, E. Enscore Jr., and I. Ham. A heuristic algorithm for the m-machine n-job flow-shop sequencing problem. *OMEGA International J. Management Science*, 11:91–95, 1983.

[117] George L. Nemhauser and Laurence A. Wolsey. *Integer and Combinatorial Optimization*. Wiley Interscience Publication, USA, 1999.

[118] G.L. Nemhauser and L. Wolsey. Maximizing submodular set functions: formulations and analysis of algorithms. In *Studies of Graph and Discrete Programming*, pages 279–301, 1981.

[119] E. Nowicki and C. Smutnicki. Worst-case analysis of an approximation algorithm for flow-shop scheduling. *Operations Research Letters*, 8:171–177, 1989.

[120] E. Nowicki and C. Smutnicki. New results in the worst-case analysis for flow-shop scheduling. *Discrete Applied Mathematics*, 46:21–41, 1993.

[121] R. Panigrahy and S. Vishwanathan. An $O(\log^* n)$-approximation for the asymmetric p-center problem. *Journal of Algorithms*, 27:259–268, 1998.

[122] P. Pardalos and eds. H. Wolkowitz. *Proceedings of the DIMACS Workshop on Quadratic Assignment Problems*, volume 16. 1994.

[123] E. Petrank. The hardness of approximation: gap location. *Computational Complexity*, 4:133–157, 1994.

[124] C. Potts, D. Shmoys, and D. Williamson. Permutation vs. nonpermutation flow shop schedules. *Operations Research Letters*, 10:281–284, 1991.

[125] M. Queyranne. Performance ratio of polynomial heuristics for triangle inequality quadratic assignment problems. *Operations Research Letters*, 4:231–234, 1986.

[126] M. Queyranne. Structure of a simple scheduling polyhedron. *Mathematical Programming A*, 2:263–285, 1993.

[127] M. Queyranne and M. Sviridenko. Approximation Algorithms for Shop Scheduling Problems with Minsum Objective. *Journal of Scheduling*, 5:287–305, 2002.

[128] P. Raghavan. Probabilistic construction of deterministic algorithms: approximating packing integer programs. *Journal of Computer and System Sciences*, 37:130–143, 1988.

[129] G. Reinelt. *The Traveling Salesman: Computational Solutions for TSP Applications*. Springer, 1994.

[130] G. Robins and A. Zelikovsky. Tighter Bounds for Graph Steiner Tree Approximation. *SIAM Journal on Discrete Mathematics*, 19:122–134, 2005.

[131] H. Röck and G. Schmidt. Machine aggregation heuristics in shop-scheduling. *Methods of Operations Research*, 45:303–314, 1983.

[132] S. Sahni and T. Gonzalez. P-complete approximation problems. *Journal of the ACM*, 23:555–565, 1976.

[133] M.W.P. Savelsbergh and M. Sol. The general pickup and delivery problem. *Transportation Science*, 29:17–29, 1995.

[134] F. Schalekamp and D. Shmoys. Algorithms for the universal and a priori TSP. *Operations Research Letters*, 36(1):1–3, 2008.

[135] Stephan Scheuerer. A tabu search heuristic for the truck and trailer routing problem. *Computer and Operations Research*, 33:894–909, 2006.

[136] A. Schrijver. *Combinatorial optimization: Polyhedra and efficiency*. Springer-Verlag, 2003.

[137] Nicola Secomandi and Francois Margot. Reoptimization Approaches for the Vehicle Routing Problem with Stochastic Demands. *To appear: Operations Research*.

[138] Danny Segev and Gil Segev. Approximate k-Steiner Forests via the Lagrangian Relaxation Technique with Internal Preprocessing. In *Proceedings of the 14th Annual European Symposium on Algorithms*, pages 600–611, 2006.

[139] S. Sevast'janov. On some geometric methods in scheduling theory: a survey. *Discrete Applied Mathematics*, 55:59–82, 1994.

[140] D. Shmoys, C. Stein, and J. Wein. Improved Approximation Algorithms for Shop Scheduling Problems . *SIAM Journal on Computing*, 23(3):617–632, 1994.

[141] David Shmoys and Kunal Talwar. A Constant Approximation Algorithm for the a priori Traveling Salesman Problem. In *Proceedings of the 13th International Conference on Integer Programming and Combinatorial Optimization*, pages 331–343, 2008.

[142] C. Smutnicki. Some results of the worst-case analysis for flow shop scheduling. *European Journal of Operational Research*, 109:66–87, 1998.

[143] D. Sotelo and M. Poggi de Aragao. An Approximation Algorithm for the Permutation Flow Shop Scheduling Problem via Erdos-Szekeres Theorem Extensions . *Manuscript*, 2008.

[144] J. Michael Steele. Variations on the monotone subsequence theme of Erdős and Szekeres. In *Discrete probability and algorithms*, volume 72 of *IMA Vol. Math. Appl.*, pages 111–131. Springer, 1995.

[145] W. Stewart and B. Golden. Stochastic Vehicle Routing: a comprehensive approach. *European Journal of Operational Research*, 14:371–385, 1983.

[146] M. Sviridenko. A Note on Permutation Flow Shop Problem. *Annals of Operations Research*, 129:247–252, 2004.

[147] M. Sviridenko. A note on maximizing a submodular set function subject to knapsack constraint. *Operations Research Letters*, 32:41–33, 2004.

[148] Paolo Toth and Daniele Vigo, editors. *The vehicle routing problem*. Society for Industrial and Applied Mathematics, 2001.

[149] J. H. van Lint and R. M. Wilson. *A Course in Combinatorics*. Cambridge University Press, 1992.

[150] Vijay V. Vazirani. *Approximation Algorithms*. Springer, 2002.

[151] Santosh Vempala and Mihalis Yannakakis. A convex relaxation for the asymmetric tsp. In *Proceedings of the 10th annual ACM-SIAM Symposium on Discrete Algorithms*, pages 975–976, 1999.

[152] A.M. Vershik and S.V. Kerov. Asymptotics of the Plancherel measure of the symmetric group and the limit form of Young tableaux. *Dokl. Akad. Nauk SSSR*, 233:1024–1027, 1977.

[153] David Williamson. Analysis of the held-karp heuristic for the traveling salesman problem. *Master's thesis, MIT Computer Science*, 1990.

[154] N.C. Wormald. Models of random regular graphs. In *Surveys in Combinatorics (J.D. Lamb and D.A. Preece, eds)*, volume 276, pages 239–298. London Mathematical Society Lecture Note Series, 1999.