Chapter 7

Polynomial Time Approximation Schemes

A fully polynomial time approximation scheme for an NP-hard optimization problem Q seems the best we can hope in approximation of optimal solutions to the problem Q: we can approximate an optimal solution to Q with an approximation ratio $1+\epsilon$ arbitrarily close to 1 in time polynomial in the input length and in the reciprocal of the error bound ϵ. Unfortunately, Theorem 6.4.1 immediately excludes the possibility of having fully polynomial time approximation schemes for many NP-hard optimization problems. In particular, a large class of NP-hard optimization problems is of the type of "subset problem", which ask to select a largest or smallest subset satisfying certain properties from a given set of objects. Most optimization problems related to graphs, such as the Independent Set problem and the Vertex Cover problem, belong to this class. Note that any such an NP-hard subset problem automatically satisfies the conditions of Theorem 6.4.1, thus has no fully polynomial time approximation scheme unless $\mathsf{P} = \mathsf{NP}$.

With the understanding that a given optimization problem Q is unlikely to have a fully-polynomial time approximation scheme, we are still interested in whether we can approximate in polynomial time optimal solutions to Q with approximation ratio arbitrarily close to 1. More precisely, for each *fixed* constant ϵ, we are interested in knowing whether there is an approximation algorithm for Q with approximation ratio bounded by $1+\epsilon$ whose running time is bounded by a polynomial (of the input length but not necessarily of $1/\epsilon$). Note that neither Theorem 6.4.1 nor Theorem 6.4.8 excludes the possibility of having this kind of approximation algorithms for an optimization problem Q even if Q satisfies the conditions in the corresponding theorem.

Definition 7.0.1 An optimization problem Q has a *polynomial-time approximation scheme* (PTAS), if for any fixed constant $\epsilon > 0$, there is a polynomial-time approximation algorithm for Q with approximation ratio bounded by $1 + \epsilon$.

In particular, an optimization problem with a fully polynomial-time approximation scheme has polynomial-time approximation schemes.

In this chapter, we present polynomial-time approximation schemes for a number of well-known optimization problems, including the Planar Graph Indep-Set problem, and the Euclidean Traveling Salesman problem. These problems, according to Theorem 6.4.1 and Theorem 6.4.8, have no fully-polynomial time approximation schemes unless $\mathsf{P} = \mathsf{NP}$ (A proof for the strong NP-hardness of Euclidean Traveling Salesman can be found in [52]). Therefore, polynomial-time approximation schemes seem the best approximation we can expect for these problems.

Polynomial-time approximation schemes for Planar Graph Indep-Set and for Euclidean Traveling Salesman are based on a polular technique that takes advantage of balanced separability of planar graphs and geometric problems and uses "separate-approximation".

We should point out that though most (non-fully) polynomial-time approximation schemes for optimization problems are of great theoretical importance, they are not very practical for small error bound ϵ. Typically, the running time of this kind of algorithms is proportional to at least $2^{1/\epsilon}$ or even to $n^{1/\epsilon}$, which is an enormous number when ϵ is small. For this, we include a brief discussion at the end of this chapter to discuss the efficiency of polynomial-time approximation schemes. In particular, we introduced the concept of "efficient polynomial-time approximation schemes" (EPTAS), which has drawn significant interests in recent research in approximation algorithms.

7.1 Optimization on planar graphs

A graph is *planar* if it can be embedded into the plane without edge crossing. There is a well-known linear-time algorithm by Hopcroft and Tarjan [71] that, given a graph, either constructs a planar embedding of the graph or reports that the graph is not planar. Planar graphs are of great practical interest (e.g, in designing integrated circuits or printed-circuit boards).

In this section, we consider optimization problems on planar graphs. Some NP-hard optimization problems on general graphs become tractable when they are restricted to planar graphs. For example, the Clique problem (given a graph G, find a largest clique, i.e., a largest subset S of vertices in G such that every pair of vertices in S are adjacent), which in general is NP-hard, can be solved in polynomial time on planar graphs, as follows. According to Kuratowski Theorem [61], a planar graph contains no clique of size larger than 4. Therefore, we can simply check every set of at most four vertices in a planar graph and find the largest clique. Since for a planar graph of n vertices, there are

$$\binom{n}{4} + \binom{n}{3} + \binom{n}{2} + \binom{n}{1} = O(n^4)$$

different sets of at most 4 vertices, the largest clique in the planar graph can be found in time $O(n^4)$.

On the other hand, some NP-hard optimization problems on general graphs remain NP-hard even when they are restricted on planar graphs. Examples include the Planar Graph Indep-Set problem and the Planar Graph Vertex-Cover problem. Therefore, finding optimal solutions for these problems on planar graphs seems as hard as finding optimal solutions for the problems on general graphs. However, graph planarity does seem to make some of these problems easier in the sense that for a class of optimization problems on planar graphs, we can derive polynomial-time approximation schemes, while the corresponding problems on general graphs have no polynomial-time approximation schemes unless P = NP. A general technique has been developed to obtain polynomial-time approximation schemes for a class of NP-hard optimization problems on planar graphs. For purposes of discussion, we will focus on the following problem to illustrate the technique, where by an *independent set* D in a graph G, we mean a subset of vertices in G in which no two vertices are adjacent:

Planar Graph Indep-Set

I_Q:	the set of planar graphs $G = (V, E)$
S_Q:	$S_Q(G)$ is the collection of all independent sets in G
f_Q:	$f_Q(G, D)$ is equal to the number of vertices in the independent set D in G
opt_Q:	max

The decision version Planar Graph Indep-Set (D) of the Planar Graph Indep-Set problem is NP-complete (see Section 1.4). Thus, it is straightforward to derive that the Planar Graph Indep-Set problem is NP-hard. Moreover, it is easy to check that the Planar Graph Indep-Set problem satisfies the conditions of Theorem 6.4.1. Therefore, the Planar Graph Indep-Set problem has no fully polynomial-time approximation scheme unless $\mathsf{P} = \mathsf{NP}$.

A (non-fully) polynomial-time approximation scheme is obtained for the Planar Graph Indep-Set problem, and for many other optimization problems on planar graphs, using the popular divide-and-conquer method based on the following *Planar Separator Theorem* by Lipton and Tarjan [95]. Interested readers are referred to the original article [95] for a formal proof for this theorem.

Theorem 7.1.1 (Planar Separator Theorem) *There is a linear time algorithm that, on a planar graph G of n vertices, partitions the vertex set of G into three disjoint subsets, A, B, and S, such that*

1. $|A|, |B| \leq 2n/3$;
2. $|S| \leq \sqrt{8n}$; *and*
3. *S separates A and B, i.e. there is no edge between A and B.*

Let $G = (V, E)$ be a planar graph and let (A, B, S) be a triple satisfying the conditions of Theorem 7.1.1. We will say that *the graph G is split into two smaller pieces A and B* (using the separator S). Let G_A be the subgraph of G induced by the vertex set A, that is, G_A is the subgraph of G that consists of all vertices in the set A and all edges whose both ends are in A. Similarly, let G_B be the subgraph of G induced by the vertex set B. Based on the fact that there is no edge in G that connects a vertex in A and a vertex in B, a simple observation is that if D_A and D_B are independent sets of the subgraphs G_A and G_B, respectively, then the union $D_A \cup D_B$ of the sets D_A and D_B is an independent set of the graph G. Moreover, since the size of a maximum independent set of the planar graph G is of order $\Omega(n)$ (this will be formally proved later) while the size of the separator S is of order $O(\sqrt{n})$, ignoring the vertices in the separator S does not seem to lose too much precision. Based on this observation, our divide-and-conquer method first recursively finds a large independent set D_A for the subgraph G_A and a large independent set D_B for the subgraph G_B (note that the subgraphs G_A and G_B are also planar graphs), then uses the union $D_A \cup D_B$ as an approximation to the maximum independent set for the graph G. This algorithm is given in Figure 7.1, where K is a constant to be determined later.

By the discussion above, the algorithm **PlanarIndSet** correctly returns an independent set for the planar graph G. Thus, it is an approximation

```
Algorithm. PlanarIndSet($K$)
Input:  a planar graph $G = (V, E)$
Output:  an independent Set $D$ in $G$

1.   If ($|V| \le K$) then
         find a maximum indepenent set $D$ in $G$ by exhaustive search;  Return($D$);
2.   else \\ At this point $|V| > K$.
2.1      split $V$ into $(A, B, S)$ as in Theorem 7.1.1;
2.2      recursively find an independent set $D_A$ for the subgraph $G_A$ and an
         independent set $D_B$ for the subgraph $G_B$;
2.3      Return($D_A \cup D_B$).
```

Figure 7.1: The algorithm **PlanarIndSet**

algorithm for the Planar Graph Indep-Set problem. We first study some properties of this algorithm.

The algorithm **PlanarIndSet** splits the graph G into small pieces. If the size of a piece is larger than K, then the algorithm further splits, recursively, the piece into two smaller pieces in linear time according to Theorem 7.1.1. Otherwise, the algorithm finds a maximum independent set for the piece using brute force method. Let us consider the number of pieces whose size is within a certain region.

A piece is *at level* 0 if its size is not larger than K. For a general $i \ge 0$, a piece is *at level* i if its size (i.e., the number of vertices in the piece) is in the region $((3/2)^{i-1}K, (3/2)^i K]$, i.e., if its size is strictly larger than $(3/2)^{i-1}K$ but not larger than $(3/2)^i K$. Since the graph has n vertices, the largest level number is bounded by $\log(n/K)/\log(3/2) = O(\log(n/K))$.

Lemma 7.1.2 *For any fixed i, each vertex of the graph G belongs to at most one piece at level i.*

Proof. Fix a vertex v of the graph G and let P be a piece containing the vertex v. Note that if P is not the whole graph G, then P must be obtained from splitting a larger piece.

Assume the contrary that the vertex v is contained in two different pieces P and Q at level i. Then both P and Q have size larger than $(3/2)^{i-1}K$ but not larger than $(3/2)^i K$. Consider the "splitting chains" for P and Q:

$$P_1, P_2, \ldots, P_t, \qquad \text{and} \qquad Q_1, Q_2, \ldots, Q_s$$

where $P_1 = P$, $Q_1 = Q$, $P_t = Q_s = G$, the piece P_j is obtained from splitting the piece P_{j+1} for $j = 1, \ldots, t-1$, the piece Q_h is obtained from splitting the piece Q_{h+1} for $h = 1, \ldots, s-1$, and all pieces P_1, …, P_t, Q_1, …, Q_s contain the vertex v. Note that the piece Q is not in the sequence P_1, P_2, …, P_t since by Theorem 7.1.1, the piece Q is split into two smaller pieces of size at most $(2/3)(3/2)^i K = (3/2)^{i-1} K$ while all pieces in the sequence P_1, P_2, …, P_t have size at least as large as $|P|$, which is larger than $(3/2)^{i-1}K$. Similarly, the piece P is not in the sequence Q_1, Q_2, …, Q_s. Let j be the smallest index such that $P_j = Q_h$ for some h (such an index j must exist since $P_t = G = Q_s$). Then $P_j \neq P$ and $P_j \neq Q$, and by the assumption on the index j, we have $P_{j-1} \neq Q_{h-1}$. Therefore, the piece P_j is split into two different pieces P_{j-1} and Q_{h-1}. Now the fact that both pieces P_{j-1} and Q_{h-1} contain the vertex v contradicts Theorem 7.1.1. □

Therefore, for each fixed i, all pieces at level i are disjoint. Since each piece at level i consists of more than $(3/2)^{i-1}K$ vertices, there are no more than $(2/3)^{i-1}(n/K)$ pieces at level i, for all i. We summarize these discussions into the following facts.

Fact 1. There are at most n pieces at level 0, each is of size at most K;
Fact 2. For each $i > 0$, there are at most $(2/3)^{i-1}(n/K)$ pieces at level i, each is of size bounded by $(3/2)^i K$; and
Fact 3. There are at most $O(\log n)$ levels.

Now we are ready to analyze the algorithm.

Lemma 7.1.3 *Algorithm* **PlanarIndSet** *runs in time* $O(n \log n + 2^K n)$.

Proof. For each piece at level $i > 0$, we apply Theorem 7.1.1 to split it into two smaller pieces in time linear to the size of the piece. Since the total number of vertices belonging to pieces at level i is bounded by n, we conclude that the total time spent by the algorithm **PlanarIndSet** on pieces at level i is bounded by $O(n)$ for each $i > 0$. Since there are only $O(\log n)$ levels, the algorithm **PlanarIndSet** takes time $O(n \log n)$ on piece splitting.

For each piece P at level 0, which has size bounded by K, the algorithm finds a maximum independent set by checking all subsets of vertices of the piece P. There are at most 2^K such subsets in P, and each such a subset can be checked in time linear to the size of the piece. Therefore, finding a maximum independent set in the piece P takes time $O(2^K |P|)$. By Lemma 7.1.2, all pieces at level 0 are disjoint. We conclude that the algorithm **PlanarIndSet** spends time $O(2^K n)$ on pieces at level 0. In summary, the running time of the algorithm **PlanarIndSet** is bounded by $O(n \log n + 2^K n)$. □

Now let us consider the approximation ratio for the algorithm **PlanarIndSet**.

Fix an $i > 0$. Suppose that we have l pieces P_1, P_2, ..., P_l, of size n_1, n_2, ..., n_l, respectively, at level i. By Lemma 7.1.2, all these pieces are disjoint. Thus, $n_1 + n_2 + \cdots + n_l = n' \leq n$. For each piece P_q of size n_q, a separator S_q of size bounded by $\sqrt{8n_q} < 3\sqrt{n_q}$ is constructed to split the piece P_q into two smaller pieces. The vertices in the separator S_q will be ignored in the further consideration. Thus, the total number of vertices in the separators for the pieces P_1, P_2, ..., P_l at level i, which will be ignored in the further consideration, is bounded by

$$3\sqrt{n_1} + 3\sqrt{n_2} + \cdots + 3\sqrt{n_l}.$$

It is well-known that under the condition $n_1 + n_2 + \cdots + n_l = n'$ the above summation will be maximized when all n_1, n_2, ..., n_l are equal. That is, when $n_1 = n_2 = \cdots n_l = n'/l$. Hence, the above summation is bounded by

$$\underbrace{3\sqrt{n'/l} + 3\sqrt{n'/l} + \cdots + 3\sqrt{n'/l}}_{l \text{ terms}} = 3l\sqrt{n'/l} = 3\sqrt{n'l} \leq 3\sqrt{nl}.$$

Now, since the number l of pieces at level i is bounded by $(2/3)^{i-1}(n/K)$ (see Fact 2), the total number of vertices belonging to separators for pieces at level i is bounded by

$$3\sqrt{n \times \left(\frac{2}{3}\right)^{i-1} \frac{n}{K}} = \frac{3n}{\sqrt{K}} \left(\sqrt{\frac{2}{3}}\right)^{i-1}.$$

Let F denote the set of all vertices that belong to a separator at some level. Let h be the largest level number. Then $h = O(\log n)$ (see Fact 3) and we derive

$$|F| \leq \sum_{i=1}^{h} \left(\frac{3n}{\sqrt{K}}\right) \left(\sqrt{\frac{2}{3}}\right)^{i-1} \leq \left(\frac{3n}{\sqrt{K}}\right) \sum_{i=1}^{\infty} \left(\sqrt{\frac{2}{3}}\right)^{i-1} = \frac{18n}{\sqrt{K}}, \qquad (7.1)$$

where we have used the fact $\sum_{i=1}^{\infty}(\sqrt{2/3})^{i-1} = 1/(1 - \sqrt{2/3}) \leq 6$.

Therefore, when the number K is large enough, the total number of vertices contained in the separators is small compared with the total number n of vertices in the graph G.

Now we derive an upper bound and a lower bound for the size of an optimal solution, i.e., a maximum independent set, to the planar graph G.

Lemma 7.1.4 *Suppose that the planar graph G has n vertices. Let D be the independent set constructed by the algorithm* **PlanarIndSet** *on input G and let F be the set of all vertices that are contained in any separators constructed by the algorithm* **PlanarIndSet**. *Then*

$$n/4 \leq Opt(G) \leq |D| + |F|,$$

where $Opt(G)$ is the size of a maximum independent set in the graph G.

PROOF. Since the graph G is planar, by the famous Four-Color Theorem [3, 4], G can be colored with at most 4 colors such that no two adjacent vertices in G are of the same color. Since all vertices colored with the same color in such coloring form an independent set for G, and there are at least $n/4$ vertices in G colored with the same color, the size $Opt(G)$ of a maximum independent set in the graph G is at least $n/4$.

Now we consider the upper bound for $Opt(G)$. Let $D_{\max}$ be a maximum independent set of the graph G and let P be a piece at level 0. It is easy to see that $D_{\max} \cap P$ is an independent set in the piece P, which cannot be larger than the maximum independent set $D_{\max}^P$ of P constructed by the algorithm **PlanarIndSet**. Note that the independent set D constructed by the algorithm **PlanarIndSet** is the union of $D_{\max}^P$ over all pieces at level 0. Let Γ_0 be the collection of all level 0 pieces. Note that

$$(\bigcup_{P \in \Gamma_0} P) \cup F$$

is the set of all vertices in the graph G, where the sets $\bigcup_{P \in \Gamma_0} P$ and F are disjoint. Therefore,

$$D_{\max} = \bigcup_{P \in \Gamma_0} (D_{\max} \cap P) \cup (D_{\max} \cap F).$$

This gives (note that all level 0 pieces in Γ_0 are disjoint)

$$\begin{aligned} Opt(G) &= |D_{\max}| \leq \sum_{P \in \Gamma_0} (|D_{\max} \cap P|) + |D_{\max} \cap F| \\ &\leq \sum_{P \in \Gamma_0} |D_{\max}^P| + |F| \\ &= |D| + |F|. \end{aligned}$$

The lemma is proved. □

Now we are ready to derive the approximation ratio for the algorithm **PlanarIndSet**. From Lemma 7.1.4, $Opt(G) \leq |D| + |F|$. Thus,

$$\frac{Opt(G)}{|D|} \leq 1 + \frac{|F|}{|D|} \leq 1 + \frac{|F|}{Opt(G) - |F|}.$$

Combining this with the inequalities $Opt(G) \geq n/4$ (see Lemma 7.1.4) and $|F| \leq 18n/\sqrt{K}$ (see Inequality (7.1)), we obtain

$$\begin{aligned} \frac{Opt(G)}{|D|} &\leq 1 + \frac{|F|}{Opt(G) - |F|} \leq 1 + \frac{|F|}{(n/4) - |F|} \\ &\leq 1 + \frac{18n/\sqrt{K}}{(n/4) - 18n/\sqrt{K}} = 1 + \frac{72}{\sqrt{K} - 72}. \end{aligned}$$

Now for any fixed constant ϵ, if we let

$$K \geq (72(1 + 1/\epsilon))^2 = 5184(1 + 1/\epsilon)^2,$$

then the algorithm **PlanarIndSet**(K) produces an independent set D for the planar graph G with the approximation ratio

$$\frac{Opt(G)}{|D|} \leq 1 + \frac{72}{\sqrt{K} - 72} 1 + \frac{72}{72(1 + 1/\epsilon) - 72} \leq 1 + \epsilon$$

in time $O(n \log n + n2^{5184(1+1/\epsilon)^2})$ (see Lemma 7.1.3). For a fixed $\epsilon > 0$, this is a polynomial-time algorithm. We conclude with the following theorem.

Theorem 7.1.5 *The* Planar Graph Indep-Set *problem has a polynomial time approximation scheme.*

Note that the algorithm **PlanarIndSet** is *not* a fully polynomial-time approximation scheme for the Planar Graph Indep-Set problem since its time complexity is not bounded by a polynomial of n *and* $1/\epsilon$.

Other optimization problems on planar graphs that have polynomial-time approximation schemes but have no fully polynomial-time approximation schemes include the Planar Graph Vertex-Cover problem, the Planar Graph H-Matching problem, the Planar Graph Dominating-Set problem, and some others (see [52] for precise definitions). Most of these polynomial-time approximation scheme algorithms use the similar technique as the one we described for the Planar Graph Indep-Set problem, i.e., using the divide-and-conquer method and the Planar Separator Theorem (Theorem 7.1.1) to separate a planar graph into small pieces by separators of small size, using brute force method to solve the problem for the small pieces, and combining the solutions to the small pieces into an approximation solution to the original planar graph.

The algorithm **PlanarIndSet** of time $O(n \log n + n2^{5184(1+1/\epsilon)^2})$ is hardly practical, even for a moderate value ϵ. Research on improving the time complexity of polynomial-time approximation schemes for optimization problems on planar graphs has performed. For example, a difference separating technique has been proposed by Baker [10]. We briefly describe the idea below based on the PLANAR GRAPH INDEP-SET problem. Let G be a planar graph. Embed G into the plane. Now the vertices on the unbounded face of the embedding give the first *layer* of the graph G. By peeling the first layer, i.e., deleting the vertices in the first layer, we obtain (maybe more than one) several separated pieces, each of which is a planar graph embedded in the plane. Now the first layers of these pieces form the second layer for the graph G. By peeling the second layer of G, we obtain the third layer, and so on. Define the *depth* of the planar graph G to be the maximum number of layers of the graph. Baker observed that for a graph of constant depth, a maximum independent set can be constructed in polynomial time by dynamic programming techniques. Moreover, for any planar graph G of arbitrary depth, if we remove one layer out of every K consecutive layers, where K is a constant, we obtain a set of separated planar graphs of constant depth. Now for each such graph of constant depth, we construct a maximum independent set. The union of these maximum independent sets forms an independent set for the original graph G. For sufficiently large K, the number of vertices belonging to the removed layers is very small and thus gives only a small error in the approximation. Baker [10] shows that this method produces a polynomial-time approximation scheme for the PLANAR GRAPH INDEP-SET problem with running time bounded by $O(8^{1/\epsilon} n/\epsilon)$.

7.2 Optimization for geometric problems

The techniques described in the previous section for approximation of optimization problems on planar graphs are based on the well-known divide and conquer method that has been extensively applied in computer algorithm design. Based on this classical technique, systematic methods have been recently developed in designing polynomial time approximation schemes for a set of famous optimization problems on Euclidean space $\mathcal{E}^d$. Roughly speaking, the new methods work in a dynamic programming manner, which partition the Euclidean space into smaller areas, construct and store good approximations for all possible situations for each smaller area, and recursively construct a good approximation for each situation for a larger area based on the approximations for the smaller areas. The techniques are general for any Euclidean space of a fixed dimension d. We will discuss in detail the construction of a polynomial time approximation scheme for the TRAVELING SALESMAN problem on Euclidean plane $\mathcal{E}^2$. Explanation will be briefly provided on how the techniques are extended to other geometric problems and to general Euclidean space $\mathcal{E}^d$ for any constant d.

Each point p in the Euclidean plane $\mathcal{E}^2$ is given by two real numbers that are the x- and y-coordinates of the point. The Euclidean distance between two points $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ is given by the formula

$$dist(p_1, p_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Let S be a set of n points in $\mathcal{E}^2$, a *traveling salesman tour* on S (or simply a *salesman tour*) is a closed walk that visits all points in S. The EUCLIDEAN TRAVELING SALESMAN problem (abbr. EUCLIDEAN TSP) is to construct a salesman tour of minimum length for a given set of points in $\mathcal{E}^2$.

It is known that EUCLIDEAN TSP is NP-hard in the strong sense [50, 104]. Therefore by Theorem 6.4.8, EUCLIDEAN TSP has no fully polynomial time approximation scheme unless $\mathsf{P} = \mathsf{NP}$.

A salesman tour is *polygonal* if it consists of a finite number of line segments. Since the Euclidean distance satisfies the *triangle inequality rule* that the length of the straight line segment connecting two points is not larger than the length of any other path connecting the two points, it is clear that any minimum salesman tour on a set S of n points can be given by a cyclically ordered sequence of the n points that specifies the polygonal salesman tour of n segments on the n points. We will concentrate on polygonal salesman tours. Sometimes in the discussion we may prefer to have the tours "bent" at points that are not in the original set S in order to make the tours satisfy certain special properties. These bends can be easily removed at the end of our approximation: once such a bent polygonal salesman tour π is constructed and suppose that it is a good approximation to the minimum salesman tour, we can simply delete points in π that do not belong to S to obtain a tour π_c that contains only points in S. Note that deleting a point p in the tour $\pi = \cdots p_1 p p_2 \cdots$ is equivalent to replacing the path $[p_1, p, p_2]$ of two line segments by a straight line segment $[p_1, p_2]$, which, by the triangle inequality rule, does not increase the length of the tour. Therefore, after deleting the points not in S from the tour π, we get a polygonal salesman tour π_c, which only bends at points in the set S and has performance at least as good as that of the original tour π.

7.2.1 Well-disciplined instances

We first show that when we study approximation algorithms for EUCLIDEAN TSP, we can perform a preprocessing to simplify the instance format and concentrate on only very well-behaved instances of EUCLIDEAN TSP.

Definition 7.2.1 Fixed a constant $\epsilon > 0$. An instance $S = \{p_1, p_2, \ldots, p_n\}$ of EUCLIDEAN TSP is *ϵ-disciplined* if for each point $p_i = (x_i, y_i)$ in S, the coordinates x_i and y_i can be written as $x_i = a_i + 0.5$ and $y_i = b_i + 0.5$, where a_i and b_i are integers, and $0 < x_i, y_i < n/\epsilon$.

A direct consequence from the above definition is that the distance between any two different points in an ϵ-disciplined instance of EUCLIDEAN TSP is at least 1. More importantly, the following lemma shows that it will suffice to concentrate on approximation schemes for ϵ-disciplined instances. For a salesman tour π of an instance S of EUCLIDEAN TSP, we let $|\pi|$ be the length of the tour π, and let $Opt(S)$ be the length of a minimum salesman tour in S.

Lemma 7.2.1 *Given any instance S of* EUCLIDEAN TSP*, and any constant $0 < \epsilon < 1$, there is an ϵ-disciplined instance S_ϵ constructible from S in linear time, such that from any salesman tour π_ϵ for S_ϵ satisfying $|\pi_\epsilon|/Opt(S_\epsilon) \leq 1 + \epsilon$, we can construct in linear time a salesman tour π for S satisfying $|\pi|/Opt(S) \leq 1 + 7\epsilon$.*

PROOF. Let Q_0 be the smallest axis-aligned square that contains all the n points in S. Since a translation of the Euclidean plane $\mathcal{E}^2$ (i.e., fix a and b and map each point (x, y) in $\mathcal{E}^2$ to the point $(x+a, y+b)$) and a proportional expanding or shrinking of $\mathcal{E}^2$ (i.e., fix a c and map each point (x, y) to the point (cx, cy)) do not change the difficulty of approximation solutions to an instance of EUCLIDEAN TSP, we can assume without loss of generality that the lower-left corner of the square Q_0 is at the origin point $(0, 0)$ and that the side length of the square Q_0 is $\lfloor n/\epsilon \rfloor$.

Place an $\lfloor n/\epsilon \rfloor \times \lfloor n/\epsilon \rfloor$ grid on the square Q_0 so that each cell in the grid is a 1×1 square whose four corners are of integral coordinates. We construct a new instance S_ϵ as follows: for each point p_i in S, we create a new point p_i' that is at the center of the 1×1 cell containing p_i (if p_i is on the boundary of more than one cell, then pick the center of any of these cells as p_i'). Note that the x- and y-coordinates of each point $p_i' = (x_i', y_i')$ are of the form $x'_i = a_i + 0.5$ and $y'_i = b_i + 0.5$, where a_i and b_i are integers. Moreover, the distance between the point p'_i and the corresponding point p_i in S is bounded by $\sqrt{2}/2$. Let $S_\epsilon = \{p'_1, p'_2, \ldots, p'_n\}$. It is clear that the set S_ϵ is an ϵ-disciplined instance of EUCLIDEAN TSP. Note that the n points in the set S_ϵ may not be all different: a point may have more than one copy in the set S_ϵ. Finally, we observe that the set S_ϵ can be constructed from the set S in linear time. In fact, it is not necessary to construct the $\lfloor n/\epsilon \rfloor \times \lfloor n/\epsilon \rfloor$ grid: the point p'_i in S_ϵ can be easily determined from the coordinates of the corresponding point p_i in S.

Since Q_0 is the smallest square containing S, either both horizontal sides or both vertical sides of Q_0 contain points in S. In particular, there are two points in S whose distance is at least $\lfloor n/\epsilon \rfloor > n/\epsilon - 1$. Therefore, the length of any salesman tour for S is larger than $2n/\epsilon - 2$. Similarly, by the construction of the instance S_ϵ, there are two points in S_ϵ whose distance is larger than $n/\epsilon - 2$, so the length of any salesman tour for S_ϵ is larger than $2n/\epsilon - 4$.

Now let π_ϵ be a salesman tour for the ϵ-disciplined instance S_ϵ. We construct a salesman tour π for the instance S as follows. We trace the salesman tour π_ϵ and at each point p'_i we make a straight line round-trip from p'_i to the corresponding point p_i in S. Note that such a straight line round-trip from p'_i to a corresponding point in S increases the tour length by at most $\sqrt{2}$. Therefore, this process results in a salesman tour π for S whose length is bounded by $|\pi_\epsilon| + n\sqrt{2}$ (of course, we can further apply the triangle inequality rule on π that may result in a further shorter salesman tour for S). In particular, we have shown

$$\frac{|\pi|}{|\pi_\epsilon|} \le 1 + \frac{n\sqrt{2}}{|\pi_\epsilon|} < 1 + \epsilon, \tag{7.2}$$

here we have used the fact $|\pi_\epsilon| > 2n/\epsilon - 4$, $\epsilon < 1$, and assumed $n \ge 8$.

The above method can also be used to estimate the value $Opt(S_\epsilon)$ in terms of $Opt(S)$: starting with an optimal salesman tour of S and adding a straight line round-trip from each point p_i in S to the corresponding point p'_i in S_ϵ result in a salesman tour of S_ϵ whose length is bounded by $Opt(S) + n\sqrt{2}$. Thus, the value $Opt(S_\epsilon)$ is bounded by $Opt(S) + n\sqrt{2}$. Combining this with the the lower bound $Opt(S) > 2n/\epsilon - 2$ gives

$$\frac{Opt(S_\epsilon)}{Opt(S)} \le 1 + \epsilon. \tag{7.3}$$

Now if the salesman tour π_ϵ satisfies $|\pi_\epsilon|/Opt(S_\epsilon) \leq 1+\epsilon$, then we have

$$\frac{|\pi|}{Opt(S)} = \frac{|\pi|}{|\pi_\epsilon|} \cdot \frac{|\pi_\epsilon|}{Opt(S_\epsilon)} \cdot \frac{Opt(S_\epsilon)}{Opt(S)} \leq (1+\epsilon)^3 \leq 1+7\epsilon,$$

here we have used inequalities (7.2) and (7.3) and the assumption $\epsilon < 1$. □

7.2.2 A PTAS for Euclidean TSP

The polynomial-time approximation scheme for Euclidean TSP is based on an important *Structure Theorem*. In this subsection, we first state the Structure Theorem, assuming its correctness, and present our algorithm. A proof for the Structure Theorem will be given in the next subsection.

According to Lemma 7.2.1, we only need to concentrate on ϵ-disciplined instances for Euclidean TSP. Fix $0 < \epsilon < 1$. Let $S = \{p_1, \ldots, p_n\}$ be an ϵ-disciplined instance for Euclidean TSP. Let Q_0 be the bounding square of S, where the lower-left corner of Q_0 is at the origin $(0,0)$, and each side of Q_0 is of length 2^{h_0}, where $h_0 = \lceil \log(n/\epsilon) \rceil = O(\log n)$.

The bounding square Q_0 can be partitioned into four equal size smaller squares by a horizontal segment and a vertical segment. Recursively, suppose Q is a $d \times d$ square that contains more than one point in S, then we partition Q into four $(d/2) \times (d/2)$ squares using a horizontal segment and a vertical segment. The partition stops when a square contains no more than one point in S. The resulting structure will be called a (regular) *dissection* of the bounding square Q_0 (see Figure 7.2(A) for illustration). The squares constructed in the dissection, including those that are further partitioned into smaller squares, will all be called *squares* of the dissection. The sides of the squares will be called *square edges*. Since the instance S is ϵ-disciplined, the edge length of each square in the dissection of Q_0 is a positive integer. Moreover, no point in S is on the boundary of any square.

The dissection of Q_0 will be represented by a *quad-tree* (i.e., a 4-ary tree) T_0 whose root corresponds to the bounding square Q_0. In general, each node v in T_0 corresponds to a square Q_v and the four children of v correspond to the four smaller squares resulted from the partition of Q_v. Figure 7.3(A) shows the quad-tree for the dissection in Figure 7.2(A), where the children of a node are ordered from left to right in terms of the clockwise ordering of the four smaller squares, starting from the lower-left one.

The root of the quad-tree T_0 will be called the *level-*0 *node* in T_0. In general, a node in T_0 is a *level-i node* if its parent is at level $i-1$. A square corresponding to a level-i node in T_0 is called a *level-i square*.

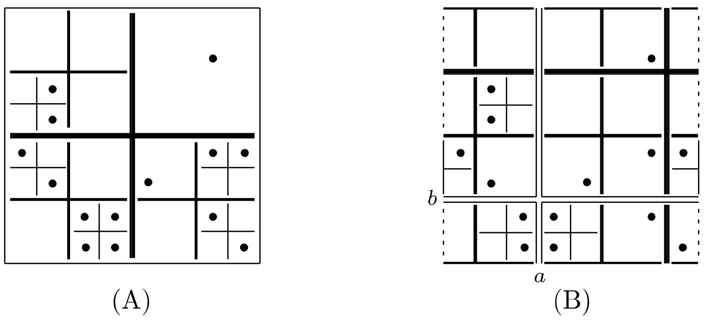

Figure 7.2: (A) a regular dissection; (B) a (a, b)-shifted dissection

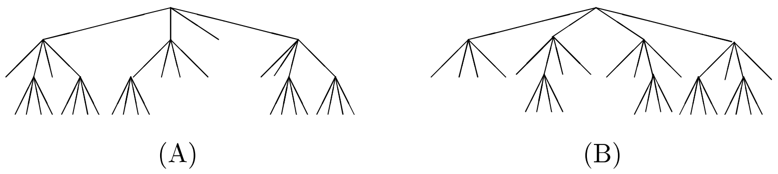

Figure 7.3: The quadtrees for the dissections (A) and (B) in Figure 7.2

Note that the depth of the quad-tree T_0 is bounded by $h_0 = \lceil \log(n/\epsilon) \rceil = O(\log n)$. Moreover, since each node in T_0 either contains points in S or has a sibling containing points in S, and two squares at the same level contain no common points in S, the number of nodes at each level of T_0 is bounded by $O(n)$ (independent of ϵ). In consequence, the total number of nodes in the quad-tree T_0 is bounded by $O(h_0 n)$.

Given a square Q and the set S_Q of points in S contained in Q, it is rather simple to go through the set S_Q and distribute the points into the four smaller squares resulted from the partition of Q. Therefore, each level of the quad-tree T_0 can be constructed in time $O(n)$. In consequence, the quad-tree T_0 can be constructed from S in time $O(h_0 n)$.

An important concept is the *shifted dissection structure*. Let a and b be two integers, $0 \leq a, b < 2^{h_0}$. We first identify, i.e., "paste", the two vertical edges of the bounding square Q_0 then cut Q_0 along the vertical line $x = a$ (see Figure 7.4(A) and (B), which use the same point set S as in Figure 7.2(A)). This is equivalent to cyclically rotating the square Q_0 to the left by a units. Then similarly, we identify the two horizontal edges of the resulting square then cut the square along the horizontal line $y = b$ (see Figures 7.4(C)). This is equivalent to cyclically rotating the square downwards by b units. Now we put a regular dissection structure on the

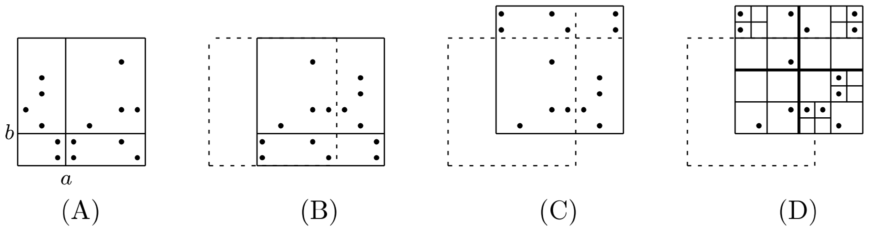

Figure 7.4: A shifted dissection structure

resulting square (see Figure 7.4(D)). This dissection is call the (a, b)-*shifted dissection* of the bounding square Q_0. The (a, b)-shifted dissection again partitions the bounding square Q_0 into "squares", with cuts along lines $x = a$ and $y = b$, and the two vertical edges and the two horizontal edges of Q_0 identified.

The (a, b)-shifted dissection can also be constructed directly on the original bounding square Q_0 with the x-coordinate shifted cyclically to the right by a units and the y-coordinate shifted cyclically upwards by b units. Regard Q_0 as the "square" by identifying the opposite edges of Q_0 and cutting Q_0 along the vertical line $x = a$ and the horizontal line $y = b$. Now the partition of Q_0 into four smaller squares is by the vertical line $x = (a+2^{h_0-1}) \bmod 2^{h_0}$ and the horizontal line $y = (b + 2^{h_0-1}) \bmod 2^{h_0}$. In general, if a square Q is bounded by four lines $x = x_0$, $y = y_0$, $x = (x_0 + 2^i) \bmod 2^{h_0}$ and $y = (y_0 + 2^i) \bmod 2^{h_0}$, then the partition of the square Q into four smaller squares is by the vertical line $x = (x_0 + 2^{i-1}) \bmod 2^{h_0}$ and the horizontal line $y = (y_0 + 2^{i-1}) \bmod 2^{h_0}$. This is illustrated in Figure 7.2(B), where the (a, b)-shifted dissection is given on the same bounding square Q_0 for the same set S of points as in Figure 7.2(A). Note that the points in the set S are not shifted with the dissection. Readers are advised to convince themselves that the figures in Figure 7.2(B) and Figure 7.4(D) give the same dissection structure.

As for regular dissections, the (a, b)-shifted dissection can also be represented by a quad-tree of depth $O(h_0)$ and $O(h_0 n)$ nodes, with all related terminologies transferred. Figure 7.3(B) gives the quad-tree for the (a, b)-shifted dissection in Figure 7.2(B) (again the children of each node are ordered from left to right in terms of the clockwise order of the four smaller squares starting from the lower-left one).

Let $D_{a,b}$ be the (a, b)-shifted dissection of the bounding square Q_0. Let e be a square edge in the dissection $D_{a,b}$. The $m + 1$ points on e that divide the edge e into m equal length segments will be called the $(1/m)$-*portals* of the square edge e.

Definition 7.2.2 A salesman tour π is (r,m)-*light* with respect to the (a,b)-shifted dissection $D_{a,b}$ if for every square edge e of $D_{a,b}$, the tour π crosses e at most r times, and each crossing of π on e is at a $(1/m)$-portal of e.

The polynomial-time approximation scheme algorithm for the Euclidean TSP is heavily based on the following Structure Theorem. We will first assume the correctness of the theorem and use it directly in our development of the algorithm. A proof for the Structure Theorem will be given in the next subsection.

Theorem 7.2.2 (The Structure Theorem) *Let S be an ϵ-disciplined instance for* Euclidean TSP *and let Q_0 be the $2^{h_0} \times 2^{h_0}$ bounding square of S, with lower-left corner at the origin $(0,0)$ and $h_0 = \lceil \log(n/\epsilon) \rceil$. Then there is a constant c_0 such that for at least half of the pairs (a,b) of integers, $0 \leq a,b < 2^{h_0}$, there exists a (c_0, c_0h_0)-light salesman tour $\pi_{a,b}$ with respect to the (a,b)-shifted dissection of Q_0 satisfying $|\pi_{a,b}| \leq (1+\epsilon)Opt(S)$.*

We remark that the constant c_0 in Theorem 7.2.2 is independent of the number n of points in the set S, but dependent of the given constant ϵ.

Based on Theorem 7.2.2, our algorithm proceeds as follows. For each pair (a,b) of integers, $0 \leq a,b < 2^{h_0}$, we apply a dynamic programming algorithm to construct an optimal (c_0, c_0h_0)-light salesman tour $\pi_{a,b}$ with respect to the (a,b)-shifted dissection of Q_0. According to Theorem 7.2.2, the shortest salesman tour π_o among all the (c_0, c_0h_0)-light salesman tours we construct over all (a,b)-shifted dissections of Q_0, $0 \leq a,b \leq 2^{h_0}$, will satisfies the condition $|\pi_o| \leq (1+\epsilon)Opt(S)$. Note that there are only $2^{h_0} \times 2^{h_0} = O(n^2)$ such pairs (a,b) satisfying $0 \leq a,b < 2^{h_0}$.

For notational simplicity, we let $m_0 = c_0h_0$.

Consider an (a,b)-shifted dissection $D_{a,b}$ of Q_0. Let π be a (c_0, m_0)-light salesman tour with respect to $D_{a,b}$. For each square Q in $D_{a,b}$, the salesman tour π passes through all points in S contained in Q, and the crossings of π over the boundaries of Q form a sequence of $(1/m_0)$-portals on the square edges of Q (note that a $(1/m_0)$-portal may appear more than once in the sequence). We will call this sequence a *crossing sequence* of the square Q. The line segments of the salesman tour π that are contained in the square Q (and pass through all points of S contained in the square Q) will be called the *partial salesman tour* (of π) in the square Q. Note that each sequence of even number of $(1/m_0)$-portals on the square edges of Q can be interpreted as a crossing sequence of Q for some salesman tour. We say that a partial salesman tour in a square Q is *consistent* with a crossing sequence σ if the partial salesman tour crosses the $(1/m_0)$-portals of the square edges of Q in exactly the same order given in the crossing sequence σ. Note that there may be more than one partial salesman tour consistent with the same crossing sequence.

Our algorithm works as follows. For each square Q in the dissection $D_{a,b}$, we construct for each possible crossing sequence σ of Q the shortest partial salesman tour in Q consistent with σ. The algorithm runs in a dynamic programming manner, starting from the leaves of the 4-ary tree $T_{a,b}$ for the dissection $D_{a,b}$. The algorithm is given in Figure 7.5.

```
Algorithm. ETSP($S, a, b$)
INPUT: an $\epsilon$-disciplined instance $S$ and integers $a$ and $b$
OUTPUT: an optimal $(c_0, m_0)$-light salesman tour on the $(a, b)$-shifted dissection

 1. construct the quad-tree $T_{a,b}$ for the $(a, b)$-shifted dissection $D_{a,b}$;
 2. for (each node $v$ in the tree $T_{a,b}$) do
      \\ starting from the leaves of $T_{a,b}$ and going bottom-up
        for (each crossing sequence $\sigma$ of the square $Q_v$ for node $v$)
            if ($v$ is a leaf) then  \\ $Q_v$ contains at most one point in $S$
                construct a shortest partial salesman tour in $Q_v$ consistent with $\sigma$;
            else \\ the children of $v$ are 4 smaller squares
                construct the shortest partial salesman tour in $Q_v$ consistent with $\sigma$,
                based on the tours constructed for the four smaller squares in $Q_v$.
```

Figure 7.5: Constructing the (c_0, m_0)-light salesman tour for $D_{a,b}$.

We give a more detailed explanation for the algorithm **ETSP**(S, a, b). Suppose the crossing sequence σ of the square Q_v corresponding to the node v in the quad-tree $T_{a,b}$ is given:

$$\sigma = [I_1, O_1, I_2, O_2, \ldots, I_r, O_r],$$

where I_i and O_i are the $(1/m_0)$-portals for the salesman tour to enter and leave the square Q_v, respectively, and $r \leq 4c_0$. In case v is a leaf in the quad-tree $T_{a,b}$, the corresponding square Q_v contains at most one point in the set S. Therefore, the shortest partial salesman tour in Q_v consistent with the crossing sequence σ can be constructed easily: if Q_v contains no point in S, then the shortest partial salesman tour in Q_v consistent with

σ should consist of the r line segments $[I_1, O_1]$, ..., $[I_r, O_r]$; while if Q_v contains a single point p in S, then the shortest partial salesman tour in Q_v should consist of one "bent" line segment $[I_i, p, O_i]$ plus $r-1$ straight line segments $[I_j, O_j]$, $j \neq i$. Since c_0 is a constant, the shortest salesman tour in Q_v consistent with the crossing sequence σ can be constructed in constant time $O(1)$.

Now consider the case where the node v is not a leaf. Then the square Q_v is partitioned into four smaller squares Q'_1, Q'_2, Q'_3, and Q'_4. Note that there are four edges of the smaller squares that are not on the edges of Q_v but are shared by the smaller squares. We will call these edges the "inner edges" of the smaller squares.

Let π be a (c_0, m_0)-light salesman tour with the crossing sequence σ on the square Q_v. If we trace π on its crossings on the edges of the smaller squares Q'_1, Q'_2, Q'_3, and Q'_4, we obtain a sequence σ_0 of $(1/m_0)$-portals on the edges of the smaller squares. It is easy to see that this sequence σ_0 can be obtained by merging the crossing sequence σ and a sequence σ'_0 of $(1/m_0)$-portals on the inner edges of the smaller squares, with the restriction that at most c_0 portal appearances from each inner edge may appear in the sequence σ'_0 (note that a portal on an inner edge may appear more than once in the sequence σ'_0). The four corresponding crossing sequences σ'_1, σ'_2, σ'_3, and σ'_4 for the four smaller squares Q'_1, Q'_2, Q'_3, and Q'_4, respectively, can be uniquely determined from the sequence σ_0. Moreover, if the partial salesman tour of π in the square Q_v is the shortest over all partial salesman tours in Q_v consistent with the crossing sequence σ, then the partial salesman tour of π in each Q'_i of the smaller squares, $1 \leq i \leq 4$, must be the shortest over all partial salesman tours in Q'_i consistent with the crossing sequence σ'_i.

Therefore, to construct the shortest partial salesman tour in Q_v consistent with the crossing sequence σ, we examine all sequences σ_0 that are obtained by merging the crossing sequence σ and a sequence σ'_0 of portals on the inner edges of the smaller squares, with the restriction that at most c_0 portal appearances from each inner edge may appear in the sequence σ'_0. We consider the complexity of systematically enumerating all these sequences.

A sequence σ_0 can be obtained as follows. Pick at most c_0 portal appearances from each inner edge of the smaller squares and let P_0 be the set of all these selected portal appearances. Now we properly insert each of the portals in P_0 into the crossing sequence σ. Of course, many sequences constructed this way do not give valid crossing sequences on the smaller squares. But this can be checked easily from the sequences themselves.

Each inner edge e of the smaller squares has $m_0 + 1$ $(1/m_0)$-portals.

Therefore, there are

$$(m_0+1)^{c_0} + (m_0+1)^{c_0-1} + \cdots + (m_0+1) + 1 \le 2(m_0+1)^{c_0}$$

ways to pick at most c_0 portal appearances from e. Therefore, totally there are at most $2^4(m_0+1)^{4c_0}$ ways to construct a set P_0 of portal appearances, in which each inner edge has at most c_0 portal appearances. Once the set P_0 is decided, the number of ways to insert the portal appearances in P_0 into the crossing sequence σ is bounded by (note that both the set P_0 and the crossing sequence σ have at most $4c_0$ portal appearances):

$$(4c_0+1)(4c_0+2)\cdots(4c_0+4c_0) = \frac{(8c_0)!}{(4c_0)!}.$$

Therefore, the number of sequences τ_0 that may represent valid crossing sequences for the four smaller squares consistent with the crossing sequence σ of Q_v is bounded by

$$2^4(m_0+1)^{4c_0} \cdot \frac{(8c_0)!}{(4c_0)!} = O((\log n)^{O(1)}),$$

and these sequences can be enumerated systematically in time $O((\log n)^{O(1)})$ (note that each sequence is of length $O(c_0) = O(1)$ so operations on each sequence can always be done in constant time).

By our algorithm, the shortest partial salesman tour consistent with each crossing sequence for each smaller square has been constructed and stored. Therefore, for each valid set $\{\sigma_1', \sigma_2', \sigma_3', \sigma_4'\}$ of crossing sequences for the smaller squares Q_1', Q_2', Q_3', and Q_4', consistent with the crossing sequence σ of Q_v, we can easily construct the corresponding partial salesman tour consistent with σ in the square Q_v. Examining all valid sets of crossing sequences for the smaller squares will result in the shortest partial salesman tour consistent with the crossing sequence σ in the square Q_v.

Summarizing the above discussion, we conclude that for each node v in the quad-tree $T_{a,b}$ and for each crossing sequence σ of the square Q_v for v, the shortest partial salesman tour in Q_v consistent with σ can be constructed in time $O((\log n)^{O(1)})$.

Now we count the number of different crossing sequences for a given square Q_v. For each edge e of Q_v, a crossing sequence may cross the portals of e at most c_0 times. There are at most $2(m_0+1)^{c_0}$ ways to pick no more than c_0 portal appearances on the edge e, and there are totally at most $2^4(m_0+1)^{4c_0}$ ways to pick no more than c_0 portal appearances from each of the four edges of Q_v. For each such selection of portal appearances, a permutation of these selected portal appearances gives a crossing sequence of Q_v. Since each such selection contains at most $4c_0$ portal appearances, the total number of possible crossing sequences of Q_v is bounded by

$$2^4(m_0+1)^{4c_0}(4c_0)! = O((\log n)^{O(1)}).$$

Therefore, the algorithm **ETSP**(S, a, b) spends at most time

$$O((\log n)^{O(1)}) \cdot O((\log n)^{O(1)}) = O((\log n)^{O(1)})$$

on each node in the quad-tree $T_{a,b}$. Since the number of nodes in the quad-tree $T_{a,b}$ is bounded by $O(n \log n)$, we conclude that to construct an optimal (c_0, m_0)-light salesman tour for the dissection $D_{a,b}$ takes time $O(n(\log n)^{O(1)})$.

Remark. There are also some "obvious" restrictions we should keep in mind when we construct crossing sequences for the squares in a dissection. For example, suppose that a square edge e is on the boundary of the original bounding square Q_0, then no portals of e should be picked in any crossing sequence of the square since an optimal (c_0, m_0)-light salesman tour will definitely not cross the edge e. Moreover, in the crossing sequences of the level-0 square Q for the (a, b)-shifted dissection, if a portal in an "out-portal" on a edge of Q, then the same position on the opposite edge of Q should be an "in-portal" since the opposite edges of Q are actually the same line in the original bounding square Q_0. These obvious restrictions can all be easily checked.

Theorem 7.2.3 *For any fixed $\epsilon > 0$, there is an $O(n^3(\log n)^{O(1)})$-time approximation algorithm for the* Euclidean TSP *problem that for any instance S constructs a salesman tour π satisfying $|\pi|/Opt(S) \leq 1 + \epsilon$.*

Proof. Let $\delta = \epsilon/7$. According to Lemma 7.2.1, we can construct a δ-disciplined instance S_δ in linear time such that for any salesman tour π_δ for S_δ satisfying $|\pi_\delta|/Opt(S_\delta) \leq 1 + \delta$, we can construct in linear time a salesman tour π for S satisfying $|\pi|/Opt(S) \leq 1 + 7\delta = 1 + \epsilon$.

The theorem is concluded since according to the above analysis: for each (a, b)-shifted dissection $D_{a,b}$, we can construct in time $O(n(\log n)^{O(1)})$ the optimal (c_0, c_0h_0)-light salesman tour with respect to $D_{a,b}$. By Theorem 7.2.2, the salesman tour π_δ that is the shortest over all (c_0, c_0h_0)-light salesman tours constructed for all shifted dissections must satisfy $|\pi_\delta|/Opt(S_\delta) \leq 1 + \delta$. Moreover, the total number of shifted dissections is bounded by $O(n^2)$. $\square$

We remark that in the time complexity $O(n^3(\log n)^{O(1)})$ of the algorithm in Theorem 7.2.3, both the constant coefficient and the constant exponent of the logorithmic function depend on the given ϵ. In particular, the algorithm is not a fully polynomial-time approximation scheme.

The time complexity of the algorithm in Theorem 7.2.3 can be improved if we are allowed to use randomization in our computation. According to Theorem 7.2.2, for at least half of the pairs (a, b), the optimal $(c_0, c_0 h_0)$-light salesman tour $\pi_{a,b}$ with respect to the (a, b)-shifted dissection $D_{a,b}$ satisfies $|\pi_{a,b}| \leq (1 + \epsilon)Opt(S)$. Therefore, if we randomly pick, say, 10 pairs of (a, b) and construct the optimal $(c_0, c_0 h_0)$-light salesman tour for each of the corresponding shifted dissections, then the probability that the shortest $\pi_{a,b}$ of these ten $(c_0, c_0 h_0)$-light salesman tours satisfies the condition $|\pi_{a,b}| \leq (1 + \epsilon)Opt(S)$ is as large as $1 - 1/2^{10} > 0.999$. Therefore, using randomization, the time-consuming enumeration of all the $O(n^2)$ pairs of (a, b) can be avoided. This gives the following theorem.

Theorem 7.2.4 *For any fixed $\epsilon > 0$ and any fixed $\delta > 0$, there is an $O(n(\log n)^{O(1)})$-time randomized approximation algorithm for the* Euclidean TSP *problem that for any instance S constructs a salesman tour π satisfying $|\pi|/Opt(S) \leq 1 + \epsilon$ with probability at least $1 - \delta$.*

7.2.3 Proof for the Structure Theorem

For completeness, we present a detailed proof for the Structure Theorem (Theorem 7.2.2) in this subsection. Readers may skip this subsection on their first reading. This will not affect continuous understanding of the rest of the book.

Let $S = \{p_1, \ldots, p_n\}$ be an ϵ-disciplined instance for Euclidean TSP. Let Q_0 be the bounding square of S, where the lower-left corner of Q_0 is at the origin $(0, 0)$, and each side of Q_0 is of length 2^{h_0}, where $h_0 = \lceil \log(n/\epsilon) \rceil = O(\log n)$. For each pair (a, b) of integers, $0 \leq a, b < 2^{h_0}$, denote by $D_{a,b}$ the (a, b)-shifted dissection of Q_0.

The Structure Theorem claims that for at least half of the pairs (a, b) of integers, $0 \leq a, b < 2^{h_0}$, there is a $(c_0, c_0 h_0)$-light salesman tour $\pi_{a,b}$ with respect to $D_{a,b}$ satisfying $|\pi_{a,b}| \leq (1 + \epsilon)Opt(S)$, where c_0 is a constant.

To prove the Structure Theorem, we start with a shortest salesman tour π_o for the instance S and show how the salesman tour π_o can be modified into a $(c_0, c_0 h_0)$-light salesman tour $\pi_{a,b}$ without much increase in tour length. For this, we need to show that how the shortest salesman tour π_o is modified so that the number of crossings at each square edge of $D_{a,b}$ is bounded by

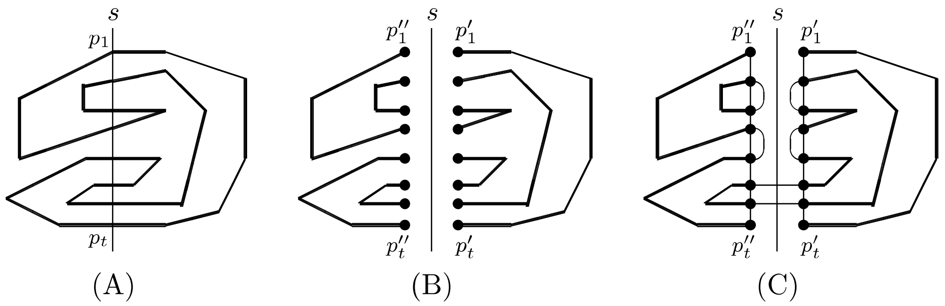

Figure 7.6: Reducing the number of crossings by patching

the constant c_0 and that all crossings occur only at the $(1/(c_0h_0))$-portals of the square edge.

Intuitively, the total number of crossings of the shortest salesman tour π_o over the square edges in $D_{a,b}$ should not be very large since a large number of crossings over a line segment should be very costful. However, it is still possible that the number of crossings of π_o over *some* square edges exceeds the constant c_0. Therefore, we first need to discuss how to reduce the number of crossings of a salesman tour over a particular square edge. The second requirement, that crossings only occur at portals of the square edges, is relatively easier to achieve since moving a crossing to its nearest portal is not very expensive. In the following, we formally implement these ideas.

First we consider the number of crossings over a particular square edge. Note by a "crossing" we mean the salesman tour hits the square edge from one side of the edge then, maybe after some "zigzag" moves along the edge, leaves the edge to the other side. In particular, it will not count as a crossing if the tour hits the edge then leaves the edge back to the same side. The reduction of the number of crossings on a square edge is based on the following "Patching Lemma".

Lemma 7.2.5 (Patching Lemma) *Let s be a line segment and π be a salesman tour for an instance S of* Euclidean TSP. *Then there is a salesman tour π' for S such that $|\pi'| \leq |\pi| + 3|s|$ and π' crosses s at most twice.*

Proof. Without loss of generality, assume that s is a vertical line.

Let $p_1, p_2, \ldots, p_t$ be the points on s at which π crosses, sorted by their y-coordinates in nonincreasing order (see Figure 7.6(A)), with $t \geq 3$. Here we do not exclude the possibility that some of these points are identical. Duplicate each point p_i into two copies p'_i and p''_i and imagine that the points

$p'_1, \ldots, p'_t$ are connected to the "right part" of the tour π while the points $p''_1, \ldots, p''_t$ are connected to the "left part" of the tour π (see Figure 7.6(B)). Now add edges

$$(p'_1, p'_2), (p'_2, p'_3), \ldots, (p'_{t-1}, p'_t), (p''_1, p''_2), (p''_2, p''_3), \ldots, (p''_{t-1}, p''_t). \qquad (7.4)$$

The total length of the added edges in (7.4) is bounded by $2|s|$ (the length of an edge is defined to be the Euclidean distance between its two endpoints). In the resulting graph, the vertices p'_1, p''_1, p'_t, and p''_t have degree 2, and all other vertices have degree 3. Based on the parity of t, we add another set of multiple edges to the graph.

In case t is even, we consider the two graphs that are obtained, respectively, by adding the following two sets of (multiple) edges, and pick among the two graphs the one that has smaller total edge length

$$\begin{gathered}(p'_2, p'_3), (p'_4, p'_5), \ldots, (p'_{t-4}, p'_{t-3}), (p''_2, p''_3), (p''_4, p''_5), \ldots, (p''_{t-4}, p''_{t-3}),\\ \text{and } (p'_{t-2}, p''_{t-2}), (p'_{t-1}, p''_{t-1}),\\ \text{and} \qquad (7.5)\\ (p'_3, p'_4), (p'_5, p'_6), \ldots, (p'_{t-3}, p'_{t-2}), (p''_3, p''_4), (p''_5, p''_6), \ldots, (p''_{t-3}, p''_{t-2}),\\ \text{and } (p'_2, p''_2), (p'_{t-1}, p''_{t-1}).\end{gathered}$$

In case t is odd, we consider the two graphs that are obtained, respectively, by adding the following two sets of (multiple) edges, and pick among the two graphs the one that has smaller total edge length:

$$\begin{gathered}(p'_2, p'_3), (p'_4, p'_5), \ldots, (p'_{t-3}, p'_{t-2}), (p''_2, p''_3), (p''_4, p''_5), \ldots, (p''_{t-3}, p''_{t-2}),\\ \text{and } (p'_{t-1}, p''_{t-1}),\\ \text{and} \qquad (7.6)\\ (p'_3, p'_4), (p'_5, p'_6), \ldots, (p'_{t-2}, p'_{t-1}), (p''_3, p''_4), (p''_5, p''_6), \ldots, (p''_{t-2}, p''_{t-1}),\\ \text{and } (p'_2, p''_2),\end{gathered}$$

(see Figure 7.6(C) for illustration, where the thinner edges are the added edges). The idea here is that the total length of the added multiple edges in (7.5) or (7.6) is bounded by $|s|$ (note that the length of the edges (p'_i, p''_i) is 0). Moreover, the salesman tour π and all the added edges form a graph G in which every vertex has an even degree. The sum of the edge lengths of the resulting graph G is bounded by $|\pi| + 3|s|$.

It is well-known in graph theory (see Appendix A, Theorems A.1 and A.2) that in a graph whose all vertices have even degrees, there is a *Eulerian tour* (i.e., a tour that passes each edge in the graph exactly once). Therefore, the Eulerian tour π' in the graph G forms a salesman tour for the instance S, which crosses the line segment s at most twice and has length bounded by $|\pi| + 3|s|$. $\square$

Now we are ready to prove the Structure Theorem. Let $c_0 = \lceil 64/\epsilon + 3 \rceil$.

Put a $2^{h_0} \times 2^{h_0}$ uniform grid structure on the bounding square Q_0 by 2^{h_0} equally spaced vertical lines and 2^{h_0} equally spaced horizontal lines (note we identify the opposite sides of the square Q_0). These lines will be called *grid lines*. Note that every square edge in the dissection $D_{a,b}$ is on a grid line. Recall that a square is a *level-i square* if its corresponding node is at level i in the quad-tree $T_{a,b}$ for the dissection $D_{a,b}$. A level-i square is a $2^{h_0-i} \times 2^{h_0-i}$ square, and the maximum level number is h_0. The square edges of a level-i square will be called *level-i square edges*. We say that a grid line ℓ is *at level-i* if i is the smallest integer such that a level-i square edge is on ℓ. Note that a level-i grid line may also contain square edges of level number larger than i.

Let π_o be a polygonal salesman tour for the instance S and $|\pi_o| = Opt(S)$. Let ℓ be a level-i grid line (either vertical or horizontal). We discuss how to reduce the number of crossings of π_o on the square edges of $D_{a,b}$ on ℓ, using the Patching Lemma. The simplest way is to apply the Patching Lemma to each level-i square edge on ℓ so that the number of crossings on each of these edges is bounded by 2. Note that this also automatically ensures that the number of crossings on each square edge of level larger than i on ℓ is also bounded by 2, since each level-j square edge on ℓ, where $j > i$, must be entirely contained in a level-i square edge on ℓ. Unfortunately, this simple method may be expensive due to the following observation: suppose that more than c_0 crossings occur on a level-j square edge e_j on ℓ, where $j > i$, then, to replace these crossings by at most 2 crossings, applying the Patching Lemma directly to the level-i square edge e_i containing e_j would possibly increase the tour length by $3|e_i|$, while applying the Patching Lemma to the square edge e_j has tour length increase bounded by $3|e_j|$. The edge length $|e_i|$ can be much larger than the edge length $|e_j|$.

Based on this observation, we apply the Patching Lemma in a “bottom up” manner starting from the shortest square edges, i.e., the square edges of the highest level number, on the grid line ℓ. The patching procedure, which is called **Modify**(ℓ), is given in Figure 7.7.

To analyze the algorithm, we introduce two new notations. Let $\omega(\pi_o, \ell)$ be the total number of crossings of the shortest salesman tour π_o on the grid line ℓ, and let $\rho(\ell, j)$ be the number of times the algorithm **Modify**(ℓ) applies the Patching Lemma to a level-j square edge on the grid line ℓ.

```
Algorithm. Modify($\ell$)  \\ $\ell$ is a level-$i$ grid line
1. for ($j = h_0$ downto $i$) do
       for (each level-$j$ square edge $e$ on $\ell$) do
           if ($\pi$ crosses $e$ more than $c_0$ times) then apply Patching Lemma to $\pi$ and $e$.
```

Figure 7.7: Reducing the number of crossings on a grid line.

Here we have to be a bit more careful about the square edges in the shifted dissection $D_{a,b}$. Recall that a "square" in $D_{a,b}$ may be formed by several non-connected pieces in the original bounding square Q_0 (see Figure 7.2(B)). If a square edge e crosses a boundary side of Q_0, then the square edge is actually formed by two non-connected segments e' and e'' in Q_0. Therefore, in case there are more than c_0 crossings of π_o over e, the Patching Lemma should be applied to the two segments e' and e'' separately since formally applying the Patching Lemma to the square edge e would introduce a partial tour that crosses a boundary side of Q_0 and continues on the opposite side of Q_0. The two separated applications of the Patching Lemma on e' and e'' may leave up to 4 crossings (instead of 2) on the square edge e in $D_{a,b}$. Therefore, we can ensure that each application of the Patching Lemma to a square edge in $D_{a,b}$ replaces a set of more than c_0 crossings by a set of at most 4 crossings, thus reducing the number of crossings by at least $c_0 - 3$. This observation gives the following relation for the values $\rho(\ell, j)$ and $\omega(\pi_o, \ell)$:

$$\sum_{j=i}^{h_0} \rho(\ell, j) = \sum_{j=0}^{h_0} \rho(\ell, j) \leq \frac{\omega(\pi_o, \ell)}{c_0 - 3}. \tag{7.7}$$

The first equality in (7.7) is because this relation is independent of the level number of the grid line ℓ.

Since the length of a level-j square edge is $2^{h_0 - j}$, each application of the Patching Lemma to a level-j square edge increases the tour length by at most $3 \cdot 2^{h_0 - j}$. Therefore, the total increase in tour length by the algorithm **Modify**(ℓ) is bounded by

$$\sum_{j=i}^{h_0} 3 \cdot 2^{h_0 - j} \rho(\ell, j) = 3 \sum_{j=i}^{h_0} \cdot 2^{h_0 - j} \rho(\ell, j). \tag{7.8}$$

The algorithm **Modify**(ℓ) modifies the salesman tour and ensures that the number of crossings over each square edge on ℓ is bounded by c_0. However, here is a potential problem we need to take care of. Without loss of generality, suppose that ℓ is a vertical grid line. Patching on ℓ may introduce many "zigzag" moves along the line ℓ, which may cause new crossings over horizontal grid lines. Let ℓ' be such a horizontal grid line and let ω' be the set of new crossings over ℓ' caused by patching the grid line ℓ. Note that all these new crossings over ℓ' are along the line ℓ so the segment s' on ℓ' holding these crossings has length 0. Therefore, applying the Patching Lemma to s' and ω' will reduce the number of crossings to at most 2 *without increasing the tour length*! In order to avoid introducing new crossings on the grid line ℓ by the patching on s' and ω', we can actually apply the Patching Lemma twice, first to the segment s' and the crossings in ω' that occur on the left side of ℓ, then to the segment s' and the crossings in ω' that occur on the right side of ℓ. This will reduce the number of crossings on the segment s' to at most 4, without increasing the tour length and the number of crossings over the grid line ℓ.

After the application of the algorithm **Modify**(ℓ), each square edge on the grid line ℓ contains at most c_0 crossings. Now we move each crossing point p to its nearest $(1/(c_0h_0))$-portal on the level-i square edge e_i on ℓ in an obvious way: instead of crossing at p, we let the salesman tour first go along the grid line ℓ (without crossing ℓ) to the nearest $(1/(c_0h_0))$-portal p' of e_i, then cross e_i at the portal p' and go along ℓ (now on the other side of ℓ) to come back to the old crossing point p and continue the tour. Note that this will also move the crossing to a $(1/(c_0h_0))$-portal on the square edge containing p' at any level on ℓ since a $(1/(c_0h_0))$-portal on a level-i square edge e_i is also a $(1/(c_0h_0))$-portal on any level-j square edge contained in e_i, where $j \geq i$. Since the distance between two neighbor $(1/(c_0h_0))$-portals on a level-i square edge is $2^{h_0-i}/(c_0h_0)$, the above modification on the tour increases the tour length by at most $2^{h_0-i}/(c_0h_0)$. Since there are no more than $\omega(\pi_o, \ell)$ crossings over the grid line ℓ, the total increase in tour length to move the crossings to portals is bounded by $2^{h_0-i}\omega(\pi_o, \ell)/(c_0h_0)$. Combining this with (7.8), we conclude that we can modify the salesman tour π_o so that the number of crossings on each square edge on the grid line ℓ is bounded by c_0 and all crossings are only at $(1/(c_0h_0))$-portals of the square edges, with the tour length increase $\tau(\ell, i)$ bounded by

$$\tau(\ell, i) = 3\sum_{j=i}^{h_0} \cdot 2^{h_0-j}\rho(\ell, j) + \frac{2^{h_0-i}\omega(\pi_o, \ell)}{c_0h_0}. \tag{7.9}$$

Now instead of computing directly the tour length increase due to the above modification, we use a probabilistic argument. Look at a given (a, b)-

shifted dissection $D_{a,b}$. With respect to the dissection $D_{a,b}$, for each $i \geq 0$, there are 2^i vertical grid lines and 2^i horizontal grid lines of level i (note that we identify the opposite sides of the level-0 bounding square Q_0 of the (a,b)-shifted dissection $D_{a,b}$). Therefore, if the integers a and b are picked randomly (with a uniform distribution) from the set $\{0, 1, \cdots, 2^{h_0} - 1\}$, then for a fixed grid line ℓ (either vertical or horizontal), the probability that ℓ is a level-i grid line is $2^i/2^{h_0}$. Therefore, the expected value of the tour length increase on the grid line ℓ is bounded by

$$\begin{aligned}
& \sum_{i=0}^{h_0} \tau(\ell, i) \cdot \text{Prob}[\ell \text{ is a level-}i \text{ grid line}] \\
\leq \; & \sum_{i=0}^{h_0} \frac{2^i}{2^{h_0}} \tau(\ell, i) \\
= \; & \sum_{i=0}^{h_0} \frac{2^i}{2^{h_0}} \left(3 \sum_{j=i}^{h_0} \cdot 2^{h_0 - j} \rho(\ell, j) + \frac{2^{h_0 - i} \omega(\pi_o, \ell)}{c_0 h_0} \right) \\
= \; & 3 \sum_{i=0}^{h_0} \sum_{j=i}^{h_0} 2^{i-j} \rho(\ell, j) + \sum_{i=0}^{h_0} \frac{\omega(\pi_o, \ell)}{c_0 h_0} \\
= \; & 3 \sum_{j=0}^{h_0} \sum_{i=0}^{j} \frac{1}{2^i} \rho(\ell, j) + \sum_{i=0}^{h_0} \frac{\omega(\pi_o, \ell)}{c_0 h_0} \\
\leq \; & 6 \sum_{j=0}^{h_0} \rho(\ell, j) + \frac{2\omega(\pi_o, \ell)}{c_0} \\
\leq \; & \frac{6\omega(\pi_o, \ell)}{c_0 - 3} + \frac{2\omega(\pi_o, \ell)}{c_0} \\
\leq \; & \frac{8\omega(\pi_o, \ell)}{c_0 - 3}.
\end{aligned}$$

Here we have used the inequality (7.7).

Thus, if the integers a and b are picked randomly, then the expected value of the total tour length increase to modify the shortest salesman tour π_o into a $(c_0, c_0 h_0)$-light salesman tour with respect to $D_{a,b}$ is bounded by (recall $c_0 = \lceil (64/\epsilon) + 3 \rceil$):

$$\sum_{\text{grid line } \ell} \frac{8\omega(\pi_o, \ell)}{c_0 - 3} = \frac{8}{c_0 - 3} \sum_{\text{grid line } \ell} \omega(\pi_o, \ell) \leq \frac{\epsilon}{8} \sum_{\text{grid line } \ell} \omega(\pi_o, \ell). \tag{7.10}$$

To complete the proof, we show

$$\sum_{\text{grid line } \ell} \omega(\pi_o, \ell) \leq 4|\pi_o| = 4 \cdot Opt(S)$$

Recall that π_o is a polygonal salesman tour consisting of n segments connecting the points in S. For each segment s in π_o with length $|s| > 0$, let $|x_s|$ and $|y_s|$ be the length of the horizontal and vertical projections of s. Then the segment s can cross at most $|x_s|+1$ vertical grid lines and at most $|y_s|+1$ horizontal grid lines. We have

$$|x_s| + |y_s| + 2 \leq \sqrt{2(|x_s|^2 + |y_s|^2)} + 2 = |s|\sqrt{2} + 2,$$

where we have used the facts that the length $|s|$ of the segment s satisfies $|s|^2 = |x_s|^2 + |y_s|^2$ and $|x_s|^2 + |y_s|^2 \geq 2|x_s| \cdot |y_s|$. Therefore, the total number of crossings of the salesman tour π_o over all grid lines can be bounded by

$$\begin{aligned} \sum_{\text{grid line } \ell} \omega(\pi_o, \ell) &\leq \sum_{\text{segment } s} (|x_s| + |y_s| + 2) \leq \sum_{\text{segment } s} (|s|\sqrt{2} + 2) \\ &\leq \sum_{\text{segment } s} (|s|\sqrt{2} + 2|s|) \leq 4 \sum_{\text{segment } s} |s| \leq 4|\pi_o| = 4 \cdot Opt(S). \end{aligned}$$

Note here we have used the fact $|s| \geq 1$, which is true because the instance S is ϵ-disciplined. Combining this with (7.10), we conclude that for random integers a and b, $0 \leq a, b < 2^{h_0}$, the expected value of the total tour length increase to modify the shortest salesman tour π_o into a $(c_0, c_0 h_0)$-light salesman tour, with respect to $D_{a,b}$, is bounded by $\epsilon \cdot Opt(S)/2$. This, by Markov's inequality [102], implies that for at least half of the pairs (a, b), the total tour length increase to modify the shortest salesman tour π_o into a $(c_0, c_0 h_0)$-light salesman tour $\pi_{a,b}$, with respect to the (a, b)-shifted dissection $D_{a,b}$, is bounded by $\epsilon \cdot Opt(S)$. That is, the $(c_0, c_0 h_0)$-light salesman tour $\pi_{a,b}$ with respect to $D_{a,b}$ satisfies $|\pi_{a,b}| \leq (1 + \epsilon)Opt(S)$.

This completes the proof for the Structure Theorem.

Remark. The probabilistic argument used above is not necessary. In fact, direct counting, using the idea adopted in the probabilistic argment, would also derive the same result. For this, we first count the number of level-i grid lines with respect to each dissection $D_{a,b}$, then compute the tour length increase on this particular dissection $D_{a,b}$. Finally, we add the tour length increases over all dissections $D_{a,b}$, and will find out that the "average" tour length increase on each dissection is bounded by $\epsilon \cdot Opt(S)/2$. Now the same conclusion should be derived.

7.2.4 Generalization to other geometric problems

A number of important techniques have been described in the discussion of our polynomial time approximation scheme for the EUCLIDEAN TSP. The concepts of ϵ-disciplined instances and (c_0, m_0)-light salesman tours enable us to concentrate on very well-behaved instances and solutions. The Patching Lemma introduces an effective method to convert a solution to a well-behaved solution, and the Structure Theorem makes it possible to apply dynamic programming to search for an optimal well-behaved solution efficiently. This systematic echnique turns out to be very effective and powerful in development of approximation algorithms for geometric problems. In the following, we briefly describe the extensions of this technique to solve other geometric problems.

The extension of Theorem 7.2.3 to EUCLIDEAN TSP in higher dimensional Euclidean space $\mathcal{E}^d$ is natural, when d is a fixed constant. As before, we first make an instance S ϵ-disciplined by rescaling and perturbation, as we did in Lemma 7.2.1. Now the Patching Lemma is applied to a $(d-1)$-dimensional hypercube (instead of to a line segment as we did in Lemma 7.2.5) to reduce the number of crossings to the $(d-1)$-dimensional hypercube to at most 2. A similar Structure Theorem can be proved based on these modifications for EUCLIDEAN TSP in $\mathcal{E}^d$ which again makes the dynamic programming possible to search for a well-behaved salesman tour for S efficiently. We omit all details and refer the readers to Arora's original paper [5]. Here we only state the final result for this extension.

Theorem 7.2.6 *For any fixed $\epsilon > 0$ and any fixed integer d, there is a polynomial time approximation algorithm for the* EUCLIDEAN TSP *problem in the d-dimensional Euclidean space $\mathcal{E}^d$ that for any instance S constructs a salesman tour π satisfying $|\pi|/Opt(S) \leq 1 + \epsilon$.*

The technique can also be applied to develop polynomial time approximation schemes for the geometric problems listed below, where d is a fixed integer. For each of these problems, we need to modify our concepts of the ϵ-disciplined instances, the well-behaved solutions, the Patching Lemma, and the Structure Theorem accordingly. We also refer our readers to the original paper [5] for details.

> EUCLIDEAN STEINER TREE:
>
> Given a set S of n points in the Euclidean space $\mathcal{E}^d$, find a minimum cost tree connecting all points in S (the tree does not have to use only the given points in S as its nodes).

Partial TSP

Given a set S of n points in the Euclidean space $\mathcal{E}^d$ and an integer $k > 1$, find the shortest tour that visits at least k points in S.

Partial MST

Given a set S of n points in the Euclidean space $\mathcal{E}^d$ and an integer $k > 2$, find k points in S such that the minimum spanning tree on the k points is the shortest (over minimum spanning trees on all subsets of k points in S).

Theorem 7.2.7 *Each of the following geometric problems has a polynomial time approximation scheme:* Euclidean Steiner Tree*,* Partial TSP*, and* Partial MST*.*

7.3 Which problems have no PTAS?

Polynomial time approximation schemes offer an efficient method to construct solutions very close to the optimal solutions for optimization problems whose optimal solutions otherwise would be hard to construct. Therefore, it is desirable to know whether a given NP-hard optimization problem has a polynomial time approximation scheme. In Section 6.4, we have developed effective and powerful methods (Theorem 6.4.1 and Theorem 6.4.8) to identify NP-hard optimization problems that have no *fully* polynomial time approximation scheme. One would expect that a similar approach could offer equally effective and powerful methods for identifying NP-hard optimization problems with no (non-fully) polynomial time approximation scheme. However, the solution to this task turns out to require deeper understanding of the complexity of NP-hard optimization problems.

It is interesting and enlightening to have a brief historical review on the development of polynomial time approximation schemes for certain NP-hard optimization problems.

Consider the Makespan problem (see Section 5.2 for a precise definition for the problem). Graham initialized the study of approximation algorithms for this important optimization problem in 1966 [59] and showed that there is a polynomial time approximation algorithm for the problem with approximation ratio 2 (see Algorithm **Graham-Schedule** and Theorem 5.2.1). The algorithm is based on a very simple greedy method that assigns each job to the earliest available processor. Three years later he further showed that if a preprocessing is performed that sorts the jobs by their processing times in non-increasing order before the algorithm **Graham-Schedule** is applied, then the approximation ratio of the algorithm can be improved to 4/3 (see Theorem 5.2.3). The approximation ratio for the MAKESPAN problem then was continuously improved. In 1978, it was improved to 1.22, then to 1.20 and then to $72/61 = 1.18\cdots$ (see the introduction section in [66] for more detailed review and references of this line of research). This line of research was eventually closed by Hochbaum and Shmoys' polynomial time approximation scheme for the problem [66], which concludes that for any $\epsilon > 0$, there is a polynomial time approximation algorithm for the MAKESPAN problem with approximation ratio bounded by $1 + \epsilon$.

Another similar story has happened for the EUCLIDEAN TSP problem. A very neat approximation algorithm based on minimum spanning trees for the problem has an approximation ratio 2. Christofides' remarkable work, based on the approach of minimum spanning trees incorporated with observations in minimum weight complete matchings and Euler tours, improved this ratio to 1.5. Christofides' ratio for EUCLIDEAN TSP stood as the best result for two decades (see the introduction section in [5] for more detailed review and references for this line of research). In fact, the difficulty for improving Christofides' ratio had made people attempt to believe that EUCLIDEAN TSP has no polynomial time approximation scheme. A surprising breakthrough by Arora [5] was made 20 years after Christofides's algorithm, which presented a polynomial time approximation scheme for EUCLIDEAN TSP, as we described in Section 7.2. It is also interesting to point out that Arora's result was initiated from his attempt at proving the nonexistence of polynomial time approximation schemes for EUCLIDEAN TSP.

The efforts are not always as successful as this for some other NP-hard optimization problems. We give an example below. A *Boolean variable x* is a variable that can take values TRUE or FALSE. A *literal* is either a Boolean variable or a negation of a Boolean variable. A *clause* is a disjunction (i.e., OR) of literals. We say that an assignment makes a clause *satisfied* if the assignment makes at least one literal in the clause have value TRUE. Consider the following problem:

> MAX-2SAT
>
> Given a set F of clauses, each containing at most 2 literals, find an assignment of the boolean variables in F that maximizes the number of satisfied clauses.

In 1974, Johnson presented an approximation algorithm of ratio 2 for the

MAX-2SAT problem. This ratio was then improved to 1.34 in 1991, then to 1.14 in 1994 and to 1.075 in 1995 (see [40] for detailed review and references for this line of research). One might expect that this line of research would eventually lead to a polynomial time approximation scheme for the MAX-2SAT problem. This, actually, is not possible as more recent research has shown that unless $\mathsf{P} = \mathsf{NP}$, there is no polynomial time approximation algorithm of ratio 1.0476 for the MAX-2SAT problem [64].

Characterization of optimization problems that have no polynomial time approximation schemes has been a very active research area in the last three decades. More recent advances, which are deep and exciting, have been made in this direction that provide effective and powerful methods for identification of optimization problems with no polynomial time approximation schemes. We will describe these results systematically later in this book. In the following, we mention some simple techniques, which can be used to prove the nonexistence of polynomial time approximation schemes for certain optimization problems.

If an optimization problem remains NP-hard even when the optimal value for its objective function is very small, then we can derive immediately that the problem has no polynomial time approximation scheme (based on the assumption $\mathsf{P} \neq \mathsf{NP}$). For example, observing for the BIN PACKING problem that deciding whether the minimum number of bins for a given set of items is 2 is NP-hard, we derive immediately that there is no polynomial time approximation algorithm of ratio less than $3/2$ for the BIN PACKING problem. Based on the fact that deciding whether the edges of a graph can be colored with at most 3 colors is NP-hard, we derive that the GRAPH EDGE COLORING problem has no polynomial time approximation algorithm for ratio less than $4/3$. These, of course, exclude immediately the possibility of existence of polynomial time approximation schemes for the problems.

For certain optimization problems, trivial modifications in input instances can change the approximation ratio dramatically. For this kind of problems, one may prove the nonexistence of polynomial time approximation schemes. Consider the general TRAVELING SALESMAN problem:

> TRAVELING SALESMAN
>
> Given a weighted complete graph G, construct a traveling salesman tour in G with the minimum weight.

Theorem 7.3.1 *If $\mathsf{P} \neq \mathsf{NP}$, then for any function $f(n) = O(c^n)$, where c is a constant, the* TRAVELING SALESMAN *problem has no polynomial time approximation algorithm of ratio bounded by $f(n)$.*

PROOF. We first reduce the NP-hard problem HAMILTONIAN CIRCUIT to the TRAVELING SALESMAN problem. Recall that the HAMILTONIAN CIRCUIT problem is, for each given graph G, decides if there is a simple cycle in G containing all the vertices in G (such a cycle is called a *Hamiltonian circuit*). For an instance G of n vertices for the HAMILTONIAN CIRCUIT problem, we construct an instance G' for the TRAVELING SALESMAN problem, which is a weighted complete graph, as follows. The graph G' has the same set of vertices as G. For each pair of vertices u and v, if $[u, v]$ is an edge in G, then the edge $[u, v]$ in G' has weight 1, and if there is no edge between u and v in G, then the edge $[u, v]$ in G' has weight $nf(n)$. It is easy to see that if the graph G has a Hamiltonian circuit then the minimum traveling salesman tour in G' has weight n, while if the graph G has no Hamiltonian circuit then the minimum traveling salesman tour in G' has weight at least $nf(n) + n - 1$. Also note that the condition $f(n) = O(c^n)$ ensures that the transformation from the (unweighted) graph G to the weighted complete G' can be done in polynomial time.

If the TRAVELING SALESMAN problem had a polynomial time approximation algorithm A of ratio bounded by $f(n)$, then we would be able to use this algorithm A to solve the HAMILTONIAN CIRCUIT problem, as follows. Applying the approximation algorithm A to the instance G'. Since the approximation ratio of A is bounded by $f(n)$, in case the graph G has a Hamiltonian circuit (i.e., the minimum salesman tour in G' has weight n), the algorithm A returns a salesman tour of weight at most $nf(n)$, while in case the graph G has no Hamiltonian circuit (so the minimum salesman tour in G' has weight at least $nf(n) + n - 1 > nf(n)$), the algorithm A returns a salesman tour of weight larger than $nf(n)$ (we assume $n > 1$). Therefore, based on the weight of the salesman tour returned by the algorithm A, we can directly decide if the graph G has a Hamiltonian circuit. This would solve the NP-hard problem HAMILTONIAN CIRCUIT in polynomial time, which in consequence would imply that $\mathsf{P} = \mathsf{NP}$. $\square$

Note that Theorem 7.3.1 is actually much stronger than saying that the TRAVELING SALESMAN problem has no polynomial time approximation schemes.