Algorithmic and Probabilistic Aspects of the Bipartite Traveling Salesman Problem

Dissertation
zur Erlangung des Doktorgrades
der Mathematisch-Naturwissenschaftlichen Fakultät
der Christian-Albrechts-Universität
zu Kiel
vorgelegt von

Andreas Baltz
Kiel
2001

Referent/in: .....

Korreferent/in: .....

Tag der mündlichen Prüfung: .....

Zum Druck genehmigt: Kiel, ...

Der Dekan

Contents

Kurzfassung

Ursprünglich aus der Tourenplanung für Bestückungsroboter stammend, ist folgende Variante des berühmten Handlungsreisendenproblems von eigenständigem theoretischen Interesse.

Alternierendes Traveling-Salesman-Problem (TSP): Die $k \cdot n$ Knoten eines vollständigen Graphen mit Kantengewichtsfunktion $w : E \to \mathbb{R}_{\geq 0}$ seien in k Klassen der Mächtigkeit n eingeteilt. Man finde einen Hamiltonkreis minimalen Gesamtgewichts, der die Klassen in fester Reihenfolge abwechselnd besucht.

In der vorliegenden Arbeit werden einige algorithmische und probabilistische Aspekte dieses Problems untersucht. Das Hauptaugenmerk liegt dabei auf dem Fall $k = 2$ des **bipartiten TSP**. Ziele der Arbeit sind das Finden effizienter Approximationsalgorithmen beweisbarer Güte, sowie die Herleitung von Abschätzungen für die typische Länge optimaler Touren.

Nach einer kurzen Einführung in grundlegende graphentheoretische Begriffe weisen wir in Kapitel 1 nach, daß bereits der Fall zweier Klassen und euklidischer Kantengewichte NP-schwer ist.

Kapitel 2 stellt einige bekannte Approximationsresultate für diesen Fall vor und präsentiert erstmalig Beispielkonfigurationen, die zeigen, daß die bewiesenen Abschätzungen nicht verbessert werden können. Außerdem wird ein neuer effizienter Approximationsalgorithmus vorgeschlagen, der in der Praxis bessere Touren liefert als die bekannten Verfahren. Wir diskutieren zwei Versionen der neuen Approximationsmethode und beweisen eine Güteabschätzung, die für die erste Version scharf ist.

Experimentelle Resultate, die das Verhalten der vorgestellten Algorithmen in der Praxis beleuchten, werden in Kapitel 3 vorgestellt.

In Kapitel 4 wenden wir uns dem allgemeinen euklidischen alternierenden TSP für zufällige Punkte im Einheitsquadrat zu. Unter Verwendung des Satzes von Beardwood, Halton und Hammersley und eines tiefliegenden Resultats von Ajtai, Komlós und Tusnády bestimmen wir die Größenordnung einer optimalen alternierenden Tour durch unabhängig und gleichmäßig verteilte zufällige Punkte als fast sicher $k\sqrt{n \log n}$. Ferner geben wir einen sequentiellen Algorithmus an, der die effiziente Konstruktion (fast sicher) optimaler Touren ermöglicht und stellen einen parallelen Approximationsalgorithmus mit polylogarithmischer Laufzeit vor. Eine kurze Behandlung poissonverteilter Punkte beschließt dieses Kapitel.

Für das asymmetrische bipartite TSP ist schon das Finden einer Tour, deren Länge um einen konstanten Faktor vom Optimum abweicht, ein NP-schweres Problem. In Verallgemeinerung eines Ansatzes von Frieze, Karp und Reed untersuchen wir in Kapitel 5 das Verhältnis zwischen dem asymmetrischen bipartiten TSP und dem nahe verwandten, effizient lösbaren Assignment-Problem. Insbesondere geben wir einen konstruktiven Beweis dafür an, daß sich aus einem optimalen Assignment des Gesamtgewichts OPT^* eine alternierende Tour der Länge $(1+\alpha)\cdot \mathrm{OPT}^*$ für jedes $\alpha \geq 0$ effizient konstruieren läßt, falls die Kantengewichte w_{ij} unabhängig und zufällig aus einer Wahrscheinlichkeitsverteilung gewählt wurden, für die mit n auch $n \cdot \mathbb{P}[w_{ij} \leq \frac{\alpha \cdot \mathrm{OPT}^*}{2n}]$ gegen unendlich geht.

Das abschließende Kapitel 6 stellt einige offene Fragen zur Diskussion.

Introduction

Originally arising from applications involving pick- & place robots, the following variant of the famous graph theoretical traveling salesman problem is of independent interest.

Alternating Traveling Salesman Problem (TSP).
Given the complete graph $G = (V, E)$ on kn vertices, a weight function $w : E \to \mathbb{R}_{\geq 0}$, and a partition of V into k classes of size n, find a Hamiltonian cycle of minimum weight that visits the classes in a fixed alternating order.

This text is mainly concerned with the case $k = 2$ of the **bipartite TSP**. The aim is to find efficient approximation algorithms of provable quality for worst-case instances and to derive bounds on the typical length of optimal tours for random point configurations.

In Chapter 1 we briefly review some graph theoretical notions and give a simple argument why the bipartite TSP is NP-hard even in the basic case where the weights are drawn from the Euclidean metric.

Two constant factor approximation algorithms for the bipartite Euclidean TSP are known so far. The matching based algorithm of Anily and Hassin [3], and Michel, Schroeter and Srivastav [21] produces for every $\epsilon > 0$ a tour of length at most $(2 + \varepsilon) \cdot \mathrm{OPT}$, where OPT denotes the length of an optimal tour. Chalasani, Motwani and Rao [8], and Frank, Korte, Triesch and Vygen [13] showed how to turn a degree constrained minimum spanning tree computed via weighted matroid intersection into a tour of length at most $2 \cdot \mathrm{OPT}$. Chapter 2 for the first time presents worst-case examples, proving that the approximation guarantees of these algorithms are tight. We propose a new algorithm based on the cycle cover and minimum spanning tree relaxations which allows for an efficient tour computation by linear programming. Two versions of this algorithm are given. We prove a worst-case approximation guarantee of $3 \cdot \mathrm{OPT}$ and show that this bound is tight for the first version of the algorithm. The significance of the proposed new method lies in a superior average case behavior.

Computational results in Chapter 3 provide a comparison of the presented approximation algorithms in practice. We consider random instances of n red and n blue points, where n ranges from 10 to 80. Three different kinds of configurations in the grid $\{1,\ldots,500\}^2$ are studied:

1. uniformly distributed points,
2. configurations with all blue points on the left and all red points on the right,
3. configurations where the red points are confined in a central region which the blue points surround.

The last two kinds of situations appear quite often in practical applications. While the known algorithms for uniformly distributed points produce tours that are typically of length $1.12 \cdot \mathrm{OPT}$ (Matching) and $1.14 \cdot \mathrm{OPT}$ (Matroid), the tours computed by new algorithm are of average length at most $1.02 \cdot \mathrm{OPT}$. An even better average case behavior shows up for the non-uniform configurations studied.

The analysis of random instances is an important and well-studied part in the theory of the TSP. Chapter 4 treats the general Euclidean alternating TSP in the unit square for random point configurations. Based on the theorem of Beardwood, Halton and Hammersley, and a deep result of Ajtai, Komlós and Tusnády on the length of optimal matchings, we perform a probabilistic analysis, showing that the length of an optimal tour through $k \cdot n$ points distributed independently and uniformly at random is of order $k\sqrt{n \log n}$ almost surely. This is quite surprising since it proves that the typical length of alternating tours excels the length of non-alternating ones not only by a factor of k but by an additional factor of $\sqrt{\log n}$. We give an efficient sequential optimal algorithm and a parallel approximation algorithm with polylogarithmic running time. The parallel algorithm is based on a divide and conquer strategy where the unit square is successively partitioned into sub-squares in which maximal matchings are constructed until a perfect matching is obtained. By joining the matching edges we get a tour of length at most $O(\log n) \cdot \mathrm{OPT}$. The chapter is concluded with a brief discussion of Poisson distributed points. As the first non-trivial result on the length of optimal TSP tours through n red and n blue random points having Poisson distribution with parameter $\lambda \in \mathbb{N}$ we prove that $12\sqrt{2}n\sqrt{\lambda \ln \lambda + \ln \lambda + \frac{1}{\sqrt{\ln \lambda}}}$ is a valid upper bound with probability $1 - 2\left(\frac{1}{\lambda}\right)^{\frac{n}{3\lambda}}$.

While Euclidean instances allow for polynomial time algorithms to generate a constant factor approximation, already this moderate task is NP-hard in the case of the asymmetric bipartite TSP studied in Chapter 5. Generalizing ideas of Frieze, Karp and Reed, we examine how well the alternating TSP can be approximated by its assignment relaxation. We give a constructive proof that an optimum assignment of weight OPT^* can be turned into an alternating tour of length

$\leq (1+\alpha)\cdot \mathrm{OPT}^*$ almost surely, for arbitrary $\alpha \geq 0$, if the weights w_{ij} are independently drawn from a probability distribution satisfying $n \cdot \mathbb{P}[w_{ij} \leq \frac{\alpha \mathrm{OPT}^*}{2n}] \to \infty$ as $n \to \infty$. Thus we have a constant factor approximation almost surely. On the other hand, we show that the assignment bound is strictly smaller than the length of an optimal bipartite tour if the weights are uniformly drawn from $\{0, 1, \ldots, \lfloor c_n n \rfloor\}$, where $c_n \to \infty$ as $n \to \infty$.

In the final Chapter 6 some open questions are stated concerning possible improvements of the presented results.

Acknoledgements

I owe sincere thanks to Prof. Dr. Anand Srivastav for introducing me to the interesting subject of the bipartite TSP and for many friendly and helpful discussions. Dr. Tomasz Schoen has carefully read several versions of the manuscript and greatly helped to improve it. Last, but not least I would like to thank the DFG for granting me a scholarship of the graduate school "Effiziente Algorithmen und Mehrskalenmethoden".

Chapter 1

Preliminaries

1.1 It is not only salesmen who travel

Numerous books and papers deal with the problem of finding shortest tours traveling salesmen or postmen can use in order to serve their customers. Astonishingly, other professional travelers have been almost entirely neglected – a fact we are bitterly coming aware of in the age of e-mail and Aldi. To make up for this deficit we devote this text to a concept that satisfies more general needs and covers various applications including the planning of Robin-Hood-tours and finding short routes for traveling gastronomy critics. As the crucial additional feature we distinguish between different classes of destinations that have to be visited in alternating order.

1.2 The alternating TSP

Consider the following tasks:

- **Robin Hood tour planning:** To efficiently take from the rich and give to the poor, Robin Hood and his gang want to follow a fast route that visits wealthy and needy clients in turn. Since Robin's men are not too well-off themselves, the aim is to find a short strictly alternating tour starting and ending at Sherwood Forest.

- **The traveling gastronomy critic's problem:** A gastronomy critic has to evaluate an equal number of bistros, restaurants, cafés, and bars according to their breakfast, lunch, tea-time, and dinner services, respectively. Starting from his home place how can he find a round tour that minimizes his fuel expenses?

It is not difficult to see that these are special instances of a graph theoretical problem we denote as the alternating traveling salesman problem.

Alternating TSP: Given the complete graph $G = (V, E)$ on kn vertices, a weight function $w : E \to \mathbb{R}_{\geq 0}$, and a partition $\{P_0, P_1, \ldots, P_{k-1}\}$ of V into classes P_j of equal size $|P_j| = n$, find a Hamiltonian cycle $C = (v_1, \ldots, v_n, v_1)$ of minimum weight, such that

$$v_i \in P_j \Rightarrow v_{i+1} \in P_{(j+1) \bmod k} \text{ for all } i \in \{1, \ldots, kn-1\},\ j \in \{0, \ldots, k-1\}.$$

As a basis for the following considerations, let us shortly review some graph theoretical notions.

1.3 Basic definitions and notations

For a set S we denote by $\mathcal{P}(S) := \{T \mid T \subseteq S\}$ the set of all subsets of S and by $\mathcal{P}_2(S) := \{\{x, y\} \mid x, y \in S,\ x \neq y\}$ the set of all subsets of S of cardinality two. Let V be a finite set of elements called **vertices**. A (simple) **graph** on V is a pair $G = (V, E)$ where either $E \subseteq \mathcal{P}_2(V)$ or $E \subseteq V \times V \setminus \mathrm{id}_V$. In the former case we call G **undirected**, in the latter we view each $(x, y) \in E$ as pointing from x to y and call G a **directed** graph (or **digraph**, for short). The elements of E are the **edges** of the graph G. A set M of disjoint edges of an undirected graph G is a called a **matching**. M is **perfect** if it covers all vertices, i.e. if $\bigcup M = V$. A graph is **complete** if it contains all possible edges (i.e. if $E = \mathcal{P}_2(V)$ or $E = V \times V \setminus \mathrm{id}_V$). To allow multiple edges between identical vertices we define a **multigraph** on V as a triple $G = (V, E, I)$ where V and E are the finite sets of vertices and edges and $I : E \to \mathcal{P}_2(V)$ indicates which vertices belong to which edge. The **neighbors** of a vertex v are collected in the set $N_{(G)}(v)$ of all vertices forming an edge with v. The cardinality of $N(v)$ is called the **degree** of v, $\deg(v)$. In the case of directed graphs we distinguish neighbors x, y of v forming edges (x, v) and (v, y) of opposite direction by calling x an **in-neighbor** and referring to y as an **out-neighbor** of v. The corresponding sets and their cardinalities are denoted by $N^-(v)$, $N^+(v)$ and $\deg^-(v)$, $\deg^+(v)$, respectively. G is **connected** if, for all distinct $x, y \in V$, we can find a sequence of vertices $(x = v_1, v_2, \ldots, v_t = y)$ starting at x and ending at y such that neighboring vertices in the sequence are neighbors in the graph. We call such a sequence a path from x to y when all of its vertices are distinct; if G is directed we additionally require that all edges point into the right directions, i.e. $(v_i, v_{i+1}) \in E$ for all $i \in \{1, \ldots, t-1\}$. In the undirected case we say that G is **Eulerian** if there is a sequence $(v_1, e_1, v_2, e_2, \ldots, e_{m-1}, v_m = v_1)$, called **Eulerian tour**, such that each edge is listed exactly once and $e_i = \{v_i, v_{i+1}\}$ for all $i \in \{1, \ldots, m-1\}$. It can be shown that an undirected connected graph is Eulerian if and only if each vertex has even degree. A **subgraph** H of G is a graph whose components are subsets of the components of G. H is an **induced subgraph** if it contains a maximal number of edges from G. A sequence $C = (v_1, \ldots, v_t, v_{t+1} = v_1)$ where

$v_1, \ldots, v_t$ are $t \geq 3$ distinct vertices and $v_{i+1} \in N^{(+)}(v)$ for all $i \in \{1, \ldots, t\}$ determines a cyclic subgraph with vertices $V(C)$ and edges $E(C)$. We call both C and the corresponding subgraph a **cycle** of **size** t. A directed graph or a multigraph may also contain cycles of size two induced by edges $e_1 = (x, y)$ and $e_2 = (y, x)$ of opposite directions or by $e_1, e_2 \in E$ with $I(e_1) = I(e_2)$. A cycle C is **Hamiltonian** if $V(C) = V$. G is called a **forest** if none of its subgraphs is a cycle. A **tree** is a connected forest. Vertices of degree 1 are the **leaves** of the tree. It is useful to observe that every tree has at least two leaves and that a tree on n vertices has exactly $n - 1$ edges. For every two vertices x, y of a tree $T = (V(T), E(T))$ there is at most one path from x to y since T is free of cycles. If we mark some vertex r as the **root** of T we can assign to every $v \in V(T)$ a **depth** as the length of the unique path from r to v (if there is no such path we assign to v the depth ∞). A **spanning tree** of G is a tree containing all vertices of G. If T is a spanning tree of the undirected graph G we can find a Hamiltonian cycle in G by **short-cutting** a **depth first traversal** of G: pick an arbitrary $r \in V$ as the root of T, then turn T into a Eulerian multigraph $\tilde{T}$ by doubling all its edges. We can now traverse the edges of the multigraph starting at r by following paths in $\tilde{T}$ from leaf (in T) to leaf which try to avoid vertices of small depth as long as possible. Let $(r = v_1, e_1, v_2, e_2, \ldots, e_m, v_{m+1} = r)$ be the sequence of vertices and edges thus traversed. We short-cut this sequence by eliminating all e_i and all but the first occurrence of each vertex. Adding r to the short-cut sequence yields a Hamiltonian cycle. Let $w : E \to \mathbb{R}_{\geq 0}$ be a **weight** function. For a subset S of E we define $w(E) := \sum_{e \in S} w(e)$ to be the weight of S. The weight of a cycle C is $w(C) := w(E(C))$. An optimal **traveling salesman tour** is a Hamiltonian cycle of minimum weight. Let $k, n \in \mathbb{N}$ and suppose that V is partitioned into k (**color**-) classes $P_0, \ldots, P_{k-1}$ of cardinality n. G is called k**-partite** (for $k = 2$: **bipartite**) if there is no monochromatic edge in G. An **alternating traveling salesman tour** is a traveling salesman tour $C = (v_1, \ldots, v_{kn}, v_{kn+1} = v_1)$ that visits the classes P_j in a fixed alternating order, so that we have $v_i \in P_j \Rightarrow v_{i+1} \in P_{(j+1) \bmod k}$ for all $i \in \{1, \ldots, kn-1\}$ and all $j \in \{0, \ldots, k-1\}$. When considering a traveling salesman tour we will use the term **"length of the tour"** as a synonym for its weight. The length of an optimal tour will be abbreviated as **OPT**. A graph algorithm has **polynomial running time** if the number of steps the algorithm needs to terminate, for a graph on n vertices, can be bounded by a polynomial in n. We use the notations $\mathbf{O}(n^k)$ and $\mathbf{o}(n^k)$, $k \geq 0$, for respectively abbreviating terms $t(n)$ that can be upper bounded by $c \cdot n^k$ for some positive constant c or that satisfy $\frac{t(n)}{n^k} \to 0$ as $n \to \infty$. When we want to give a lower bound instead of an upper bound we write Ω instead of O.

As a tool from probability theory we will frequently use the well-known **Markov inequality**.

Theorem (Markov's inequality). *If X is a non-negative random variable with mean $\mathbb{E}[X]$ and $\alpha > 0$, then*

$$\mathbb{P}[X \geq \alpha \mathbb{E}[X]] \leq 1/\alpha.$$

Moreover, the following two large deviation **inequalities of Angluin and Valiant** [2] will be most useful.

Theorem (Angluin-Valiant). *Let $X_1, \ldots, X_n$ be independent random variables with $0 \leq X_i \leq 1$ for all i and set $X := \sum_{i=1}^{n} X_i$. Then for every $0 < \beta \leq 1$:*

(i) $\mathbb{P}[X > (1+\beta)\mathbb{E}[X]] \leq e^{-\frac{\beta^2 \mathbb{E}[X]}{3}}$,

(ii) $\mathbb{P}[X < (1-\beta)\mathbb{E}[X]] \leq e^{-\frac{\beta^2 \mathbb{E}[X]}{2}}$.

Chapter 2

The bipartite TSP with Euclidean weights

As a simple but non-trivial class of instances of the alternating TSP let us consider configurations of an equal number of red and blue points in the plane (corresponding to the rich and poor people in the Robin Hood problem) with distances measured according to the Euclidean metric:

Let $n \in \mathbb{N}$, let $R := \{r_1, \ldots, r_n\}$, $B := \{b_1, \ldots, b_n\}$ represent n red and n blue points $\in \mathbb{R}^2$, and define a weight function $w : \mathcal{P}_2(R \cup B) \to \mathbb{R}_{\geq 0}$ as the Euclidean distance between points from $R \cup B$. We want to find short alternating TSP tours in the complete Graph $G = (V, E)$ where $V = R \cup B$ and $E = \mathcal{P}_2(V)$. [One might wonder, why we do not restrict our considerations to the complete *bipartite* graph $(V, \{\{r, b\} \mid r \in R, b \in B\})$. The reason is that even "forbidden" edges $\in \mathcal{P}_2(R) \cup \mathcal{P}_2(B)$ can serve us to construct short alternating tours.]

2.1 Complexity

How difficult is finding an optimal alternating tour in G? We will show that this problem is at least as hard as finding an optimal non-alternating TSP tour.

Suppose, we are given n red points in the Euclidean plane. If we know how to construct optimal alternating tours we can find an optimal TSP tour easily: we simply add a blue twin to each red point at a distance of zero, compute an optimal alternating tour, and delete the blue points thereby shortcutting the tour edges [i.e. we replace each subsequence (r_i, b_j, r_{i+1}) of the optimal alternating tour by (r_i, r_{i+1}). The triangle inequality guarantees that $w(\{r_i, r_{i+1}\}) \leq w(\{r_i, b_j\}) + w(\{b_j, r_{i+1}\})$]. Hence the tour obtained is at most as long as the alternating tour. However, any optimal TSP tour can be turned into an alternating tour of equal length by connecting each red point to a blue one via an edge of zero weight. Thus the constructed tour is optimal.

Since the Euclidean TSP is known to be NP-hard [15], we have proved

Theorem 1 *Finding optimal alternating tours through two classes of points in the Euclidean plane is NP-hard.*

On the other hand, it is not clear at all how to construct optimal alternating tours through given points if we know how to find optimal non-alternating ones. Note that, for example, we cannot consider equally colored points as being infinitely far apart, since this makes the problem non-Euclidean.

2.2 The trouble with good TSP approximations

We have seen that the Euclidean bipartite TSP is at least as hard as the Euclidean TSP, so we cannot hope to come up with an optimal solution in polynomial time, unless $P = NP$. However, for the Euclidean TSP several approximation algorithms are known which are guaranteed to produce a solution of length smaller than a constant multiple of the optimal value in polynomial time. The best constant factor approximation algorithm was given by Christofides [9] in 1976. It computes a TSP tour of length smaller than $\frac{3}{2}$OPT in time $O(n^3)$. In 1996 and 1997 Arora [4],[5] was able to prove a polynomial time approximation scheme for constructing a tour of length at most $(1+\varepsilon)$OPT. The randomized version of his PTAS runs in time $O\left(n(\ln n)^{1/\varepsilon}\right)$ while derandomization increases the running time to $O\left(n^3(\ln n)^{1/\varepsilon}\right)$. Rao and Smith [26] improved the running time to $O(\varepsilon^{-O(1/\varepsilon)}n + n\log n)$. (Although theoretically most efficient, Arora's approximation scheme is of limited practical value, since the constants hidden in the O-notation are considerably large.) Unfortunately, neither Christofides' nor Arora's construction seems to generalize to the alternating case.

Christofides' algorithm is based on the observation that the number of vertices of odd degree in each graph is even. Hence any spanning tree of minimum weight can be turned into a graph where each vertex has even degree by connecting the vertices of odd degree via a minimum weight perfect matching. We can obtain a TSP tour by traversing all the edges of this graph, using the above described short-cut technique whenever we visit a vertex for the second time. By the triangle inequality a tour thus obtained is of length at most w(edges of minimum spanning tree) $+ w$(minimum weight perfect matching) $\leq$OPT$+\frac{1}{2}$OPT, since any tour contains a spanning tree and short-cutting the tour via eliminating all but the above vertices of originally odd degree yields a shorter tour consisting of two perfect matchings. Trying to mimic this construction for the alternating case we encounter two obstacles.

1. What to do if the odd vertices in our minimum spanning bipartite tree do not split into an equal number of red and blue points?

2. How to perform short-cutting?

There is an elegant way to overcome the first problem. Let V_0 denote the set of red and blue vertices of odd degree. We compute a minimum weight perfect matching M in the complete graph on V_0, where each edge $\{x, y\} \in V_0$ is assigned the weight of a shortest path from x to y in our bipartite tree. Since an optimal alternating tour induces two perfect "matchings" consisting of alternating paths joining the vertices of V_0, we can double the paths corresponding to M in our tree to obtain a Eulerian graph of overall weight at most $\frac{3}{2}$OPT. The second obstacle is severe. Michel [20] considers a straight-forward generalized short-cutting procedure involving separate short-cut lists for R and B and proves that no constant approximation factor can be given.

Arora starts his construction by successively dissecting the Euclidean plane into rectangular boxes until each box contains at most one point. He shows how to modify an optimal tour – without significantly increasing its length – in such a way that for each box the number of edges crossing its boundary is smaller than a constant. By slightly bending the edges he forces them to pass through prespecified portals. Thus he reduces the original task to a problem that can be solved efficiently by complete enumeration (see the book of Korte and Vygen [17] for a detailed description of Arora's algorithm). The trouble with this approach for the alternating TSP is how to perform crossing elimination if we forbid monochromatic edges.

2.3 The matching method

Although Christofides' and Arora's techniques may not be suitable to produce alternating tours directly, we still can profit by them. Anily and Hassin [3], and, independently, Michel, Schroeter and Srivastav [21] observed that inserting a perfect matching into a TSP tour yields an alternating tour whose length is bounded by the triangle inequality to be at most the length of the TSP tour plus twice the weight of the matching. More formally, if we have a tour $C = (r_1, r_2, \ldots, r_n, r_1)$ through the red points and a permutation $\pi \in S_n$ determining a perfect matching $M := \big\{\{r_i, b_{\pi(i)}\} \mid i \in \{1, \ldots, n\}\big\}$, both, $C_1 := (r_1, b_{\pi(1)}, r_2, b_{\pi(2)}, \ldots, r_n, b_{\pi(n)}, r_1)$ and $C_2 := (r_1, b_{\pi(1)}, r_n, b_{\pi(n)}, r_{n-1}, b_{\pi(n-1)}, \ldots, r_2, b_{\pi(2)}, r_1)$ are alternating tours of length at most $w(C) + 2w(M)$, since for each subsequence $(r_i, b_{\pi(i)}, r_{i+1})$ the triangle inequality gives

$$\begin{aligned} w(\{r_i, b_{\pi(i)}\}) + w(\{b_{\pi(i)}, r_{i+1}\}) &\le w(\{r_i, b_{\pi(i)}\}) + w(\{b_{\pi(i)}, r_i\}) + w(\{r_i, r_{i+1}\}) \\ &= 2w(\underbrace{\{r_i, b_{\pi(i)}\}}_{\in M}) + w(\underbrace{\{r_i, r_{i+1}\}}_{\text{in } C}). \end{aligned}$$

Any alternating tour consists of an even number of edges, hence we can decompose it into two perfect matchings. Choosing M as a matching of minimum

weight (which can be found in time $O(n^3)$ by the "Hungarian Algorithm" of Kuhn and Munkres, see for example the book of Cook et al. [10]) gives the estimate $w(M) \leq \frac{1}{2}\mathrm{OPT}$. Taking into account the complexity and quality of Arora's and Christofides' constructions we get the following theorem.

Theorem 2 *Given n red and n blue points in the Euclidean plane we can obtain an alternating tour of length at most*

a) $2.5OPT$ *in time* $O(n^3)$

b) $(2+\varepsilon)OPT$ *in time* $O(n^3 + \varepsilon^{-O(1/\varepsilon)}n)$ *for all* $\varepsilon > 0$

via

1. *Computing a TSP tour C through the red points according to a) Christofides' or b) Arora's method,*

2. *Running the Hungarian Algorithm to determine a minimum weight perfect matching M between red and blue points, and*

3. *Inserting M into C.*

One may conjecture that the worst case bound of $(2+\varepsilon)\mathrm{OPT}$ is fairly pessimistic. Indeed, computational experiments with uniformly distributed random points suggest that typically the matching method produces alternating tours which are of length smaller than $1.21 \cdot \mathrm{OPT}$ (worst case) and $1.12 \cdot \mathrm{OPT}$ (average case), respectively. (Our implementation of the matching method uses the Nearest Neighbor- and full 3-OPT heuristics to gain a near optimal tour through the red points. Empirical studies of Reinelt [25] indicate that this combination even with limited 3-OPT achieves solutions typically deviating by no more than 4% from the optimum value.) However, it is impossible to generally improve the upper bound. Even if we restrict ourselves to configurations where all red points are confined in some central region of space which all blue points surround, we can give examples for which an approximation factor of 2 is approached arbitrarily closely. It is easy to see that the matching method performs poorly if in the insert step of the algorithm we do not consider both possible directions for traversing the TSP tour. Figure 1 shows an example where inserting a matching into a clockwise traversal of the TSP tour results in an alternating tour about $\frac{3}{2}$ times as long as obtained when choosing the counterclockwise direction.

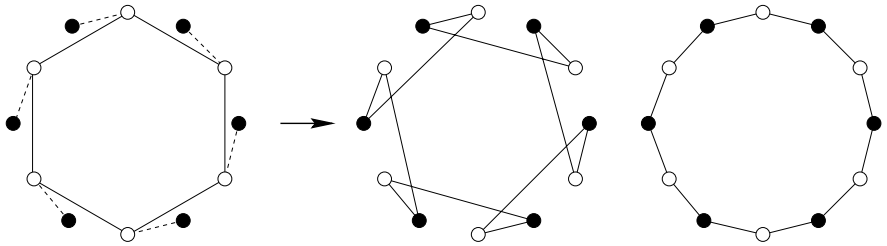


Figure 1: Alternating tours obtained by inserting matching edges in clockwise and counterclockwise order.

Here is a configuration of centered red points surrounded by blue points for which either traversal of an optimal tour through the red points induces an alternating tour of length almost 2·OPT.

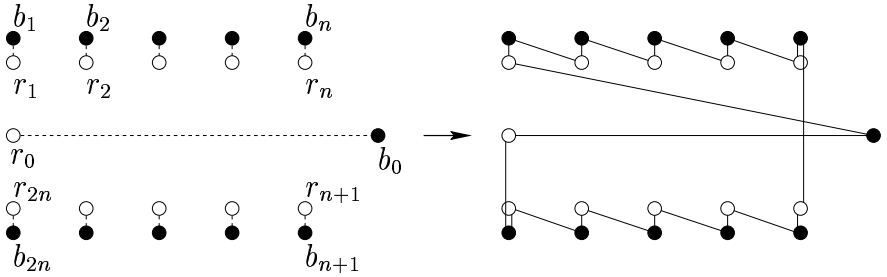


Figure 2: The unique minimum weight perfect matching consists of the edges indicated by dotted lines. Insertion into an optimal TSP tour through the red points $r_0, \ldots, r_{2n}$ yields a tour of length about $2 \cdot \mathrm{OPT}$.

Lemma 1 *For $n \in \mathbb{N}$ let $R(n) = \{r_0, r_1, \ldots, r_{2n}\}$, $B(n) = \{b_0, b_1, \ldots, b_{2n}\}$ be sets of $2n+1$ red and blue points, respectively, where*

$$r_i := \begin{cases} (1,2), & i = 0 \\ (i,3), & 1 \le i \le n \\ (2n+1-i,1), & i > n \end{cases}, b_i := \begin{cases} (n+1,2), & i = 0 \\ (i, 3+\frac{1}{2n^2}), & 1 \le i \le n \\ (2n+1-i, 1-\frac{1}{2n^2}), & i > n \end{cases}.$$

Any alternating tour $C(n)$ through $R(n) \cup B(n)$ produced by the matching method satisfies

$$\frac{w(C(n))}{OPT} > 2 - o(1) \xrightarrow{n\to\infty} 2.$$

Proof. The points are chosen to lie on the boundaries of two concentric rectangles (see figure 2), so for an alternating tour it is optimal to visit them in sequence $r_0, b_1, r_1, b_2, r_2, \ldots, b_n, r_n, b_0, r_{n+1}, b_{n+1}, \ldots, r_{2n}, b_{2n}, b_0$, and an optimal TSP tour through the red points is given by $(r_0, r_1, \ldots, r_{2n}, r_0)$. Both tours have total length of order $2n + o(n)$. We want to show that

(∗) For all $n \in \mathbb{N}$ the only minimum weight perfect matching is

$$M_0(n) := \{\{r_i, b_i\} \mid i \in \{0, \ldots, 2n\}\}.$$

Inserting $M_0(n)$ into an optimal red tour yields an alternating tour $C(n)$ as shown in figure 2. Its length is easily estimated to be of order $4n - o(n)$, so that

$$\frac{w(C(n))}{\mathrm{OPT}} > \frac{4n - o(n)}{2n + o(n)} = 2 - o(1) \xrightarrow{n\to\infty} 2.$$

To prove (∗) let $\pi : \{0, \ldots, 2n\} \longrightarrow \{0, \ldots, 2n\}$ be a permutation such that $M := \{\{r_0, b_{\pi(0)}\}, \ldots, \{r_{2n}, b_{\pi(2n)}\}\}$ is a perfect matching of minimum weight. It suffices to show $\{r_0, b_0\} \in M$, because $M_0(n)$ clearly is the unique minimum of all perfect matchings containing $\{r_0, b_0\}$. Suppose for a contradiction that $\{r_0, b_0\} \notin M$ and let t be the smallest positive integer such that $\pi^t(0)(:= \underbrace{\pi \circ \cdots \circ \pi}_{t \text{ times}}(0)) = 0$. Since 0 is no fix-point of π, the sequence $(0, \pi(0), \ldots, \pi^{t-1}(0))$ consists of at least two components. All subsequent components $\pi^i(0), \pi^{i+1}(0)$ represent an edge $\{r_{\pi^i(0)}, b_{\pi^{i+1}(0)}\}$ in M which is of weight larger than the Euclidean distance $\|r_{\pi^i(0)} - r_{\pi^{i+1}(0)}\|_2$ of the points $r_{\pi^i(0)}, r_{\pi^{i+1}(0)}$. Hence $w(M)$ is greater than the Euclidean length of the path $r_0, r_{\pi(0)}, \ldots, r_{\pi^{t-1}(0)}, b_{\pi^t(0)} = b_0$ from r_0 via some r_i to b_0. But now

$$\begin{aligned} w(M) &> \|r_0 - r_{\pi(0)}\|_2 + \|r_{\pi(0)} - r_{\pi^2(0)}\|_2 + \ldots + \|r_{\pi^{t-1}(0)} - b_0\|_2 \\ &\geq \|r_0 - r_{\pi(0)}\| + \|r_{\pi(0)} - b_0\| \\ &\geq \min\{\sqrt{(x-1)^2 + 1} + \sqrt{(n+1-x)^2 + 1} \mid 1 \leq x \leq n\} \\ &\geq 2\sqrt{\left(\frac{n}{2}\right)^2 + 1} > n + \frac{1}{n} = w(M_0), \end{aligned}$$

which contradicts the choice of M. □

2.4 The spanning tree strategy

A straightforward way to find a TSP tour of length at most $2 \cdot \mathrm{OPT}$ is the following:

- Compute a spanning tree T of minimum weight – this can be done in time $O(|E| \log |E|)$ by Kruskal's [18] algorithm or in time $O(|V|^2)$ by the method of Prim [24].
- Choose an arbitrary vertex r as the root of T, then double the edges and perform a depth first traversal to obtain a Eulerian tour.
- Construct a TSP tour by short-cutting the Eulerian tour.

How about using a bipartite spanning tree of minimum weight in the same way to get an alternating tour of length $2 \cdot \mathrm{OPT}$? Figure 3 shows an example given by Chalasani, Motwani, and Rao [8] of a bipartite tree for which no depth first traversal can be short-cut to yield an alternating tour.

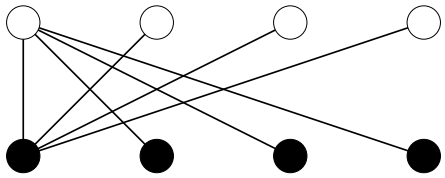


Figure 3: A bipartite tree for which short-cutting of depth first Eulerian tours never results in an alternating tour.

We see that short-cutting is a problem if in our bipartite tree there is a red vertex with more than one successive branch ending in a blue leaf. One the other hand, we are out of trouble if we have a bipartite tree where each red vertex is of degree at most two.

Definition 1 *Let G be a bipartite graph on $R \cup B$ with Euclidean edge weights. A minimum weight spanning tree of G is called* B2-tree *if* $\deg(r) \leq 2$ *for all* $r \in R$.

Lemma 2 *Let G be a Euclidean bipartite Graph on $R \cup B$ with $|R| = n = |B|$ and let T be a B2-tree of G.*

a) There is exactly one $r_0 \in R$ with $\deg(r_0) = 1$.

b) Viewing T as being rooted at its unique read leaf r_0 any depth first traversal can be short-cut to yield an alternating tour of length $\leq 2 \cdot OPT$.

Proof. a) Since T is a spanning tree of G it contains $2n - 1$ edges. Hence

$$2n - 1 = \sum_{r \in R} \deg(r) = \sum_{\substack{r \in R, \\ \deg(r)=2}} 2 + \sum_{\substack{r \in R, \\ \deg(r)=1}} 1,$$

and we can conclude that

$$|\{r \in R \mid \deg(r) = 2\}| = n - 1, \ |\{r \in R \mid \deg(r) = 1\}| = 1.$$

b) We prove that the short-cut tour is alternating by showing that this claim holds for every subtree T' of T such that T' consists of an equal number of red and blue vertices, and $r_0 \in V(T')$. So, let T' be a subgraph of T such that T' is a tree with $r_0 \in V(T')$ and $|V(T') \cap R| = |V(T') \cap B|$. We argue by induction on $|E(T')|$. Obviously, the claim is true if $|E(T')| = 1$. For $|E(T')| \geq 3$ consider an arbitrary depth first traversal of T' and let b_0 denote the last leaf

it visits before returning to the root r_0. Since r_0 root is the only red leaf we deduce that $b_0 \in B$ and, in view of a), b_0 is the unique descendent of a vertex $r_1 \in R$. Short-cutting the depth first traversal yields a tour of the form $C = (r_0, v_1, \ldots, v_{2l}, r_1, b_0, r_0)$. We can remove r_1 and b_0 from T' to obtain a tree T'' satisfying the premises of our induction assumption. Short-cutting a similar traversal of T'' gives the tour $(r_0, v_1, \ldots, v_{2l}, r_0)$ which is alternating by induction assumption. Hence C is an alternating tour, too. By deleting an arbitrary edge from an optimal alternating tour we get a bipartite tree satisfying $\deg(r) \leq 2$ for all $r \in R$. Thus $w(T) \leq \mathrm{OPT}$ and we conclude that $w(C) \leq 2 \cdot \mathrm{OPT}$ as claimed.□

Chalasani, Motwani and Rao [8] and, independently, Frank, Korte, Triesch and Vygen [13] proved that the spanning tree strategy is a polynomial 2-factor approximation algorithm for the alternating TSP since a B2-tree can be found in polynomial time via the weighted matroid intersection algorithm of Edmonds [11]. With Frank's [12] refined version of Edmonds' algorithm a B2-tree is guaranteed to be computed in time $O(n^7)$. Clearly, this term dominates the time needed for constructing an alternating tour, and we get the following theorem.

Theorem 3 *An alternating tour of length $\leq 2 \cdot OPT$ can be constructed in time $O(n^7)$ by determining a B2-tree T and performing a depth first traversal starting at the unique red leaf of T followed by short-cutting.*

Lemma 3 *The worst case bound of $2 \cdot OPT$ for the length of a B2-tour is tight.*

In order not to bother the reader with intransparent notations, we abstain from a formal description of our worst case example and refer to illustrations instead.

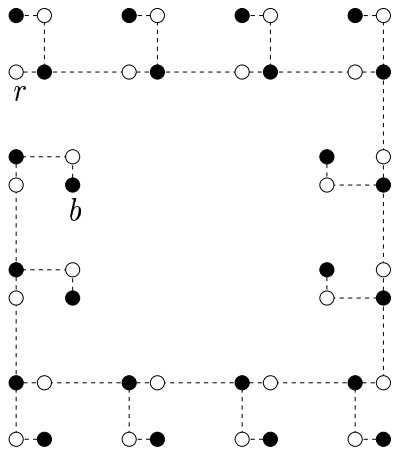


Figure 4: Structure of a B2-tree for a configuration of k small 4-point rectangles forming a large rectangle.

Figure 4 depicts a configuration of points consisting of $k = 12$ small congruent 4-point rectangles arranged to lie on the boundary of a large rectangle The dotted

lines indicate the structure of a B2-tree. Actually, the bipartite tree is even minimum spanning, since it consists of shortest edges only. Clearly, a depth first traversal initially choosing a path from r to b results in a tour as shown in figure 5a) of length about twice as large as the tour illustrated in figure 5b).

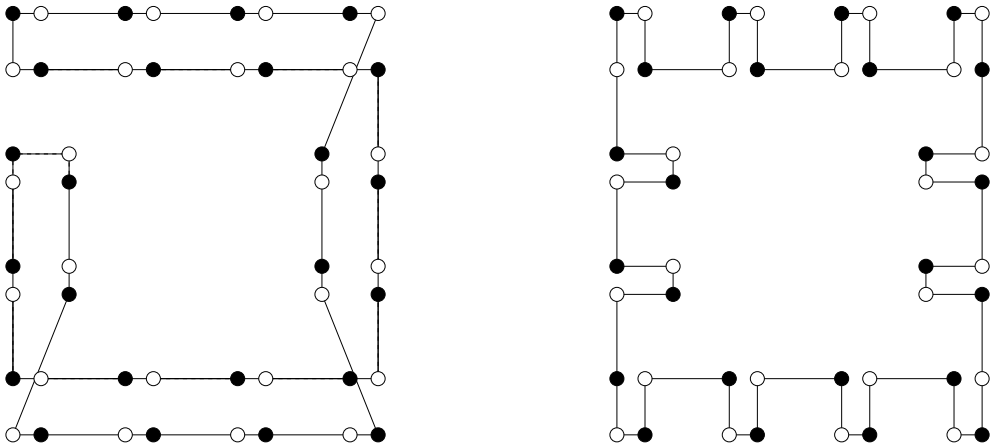


Figure 5: a) B2-tour b) optimal tour

By reducing the size and increasing the number k of small rectangles we can approach the worst case bound arbitrarily closely. □

Michel [20] observed that there is another way a B2-tree can be used to obtain an alternating tour of length at most $2 \cdot \mathrm{OPT}$. It is easily seen by induction that a B2-tree can be decomposed into a perfect matching by successively marking edges incident to leaves and then removing the vertices incident to a marked edge. Doubling the matching edges yields a set of subtours $C_1, \ldots, C_n$, we can combine into an alternating tour as follows.

- Choose tours C_i, C_j such that there is a "join edge" $\{r_i', b_j'\}$ in the B2-tree connecting C_i and C_j. We find $b_i' \in V(C_i)$ and $r_j' \in V(C_j)$ with $\{r_i', b_i'\} \in E(C_i)$, $\{r_j', b_j'\} \in E(C_j)$.

- Select a join edge $\{r_i, b_j\}$ and $b_i \in N(r_i) \cap V(C_i), r_j \in N(b_j) \cap V(C_j)$ such that $w(\{b_j, r_i\}) + w(\{b_i, r_j\}) - w(\{b_j, r_j\}) - w(\{b_i, r_i\})$ is minimum. Merge C_i and C_j by adding the edges $\{r_i, b_j\}$, $\{b_i, r_j\}$ and deleting $\{r_i, b_i\}$, $\{b_j, r_j\}$.

- Repeat these steps until all subtours are merged.

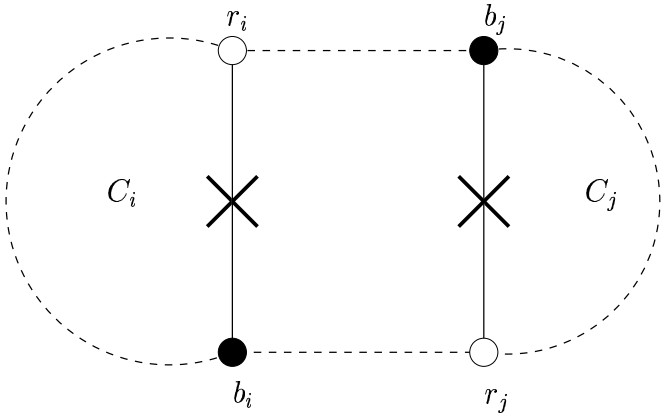


Figure 6: The merging step.

Lemma 4 *The merging procedure yields an alternating tour of length at most* $2 \cdot OPT$.

Proof. From the triangle inequality we deduce

$$w(\{b_i, r_j\}) \leq w(\{b_i, r_i\}) + w(\{r_i, b_j\}) + w(\{b_j, r_j\}). \qquad (*)$$

This relation is sometimes referred to as the **quadrangle** or **square inequality**. Anyway, we see that by connecting the tours C_i, C_j in the described way the overall weight increases by at most twice the weight of the join edge. Hence the weight of the alternating tour obtained is bounded by

$$2w(\text{matching}) + 2w(\text{join edges in B2-tree}) \leq 2w(\text{B2-tree})$$

which is smaller than $2 \cdot \text{OPT}$. □

Note that we may as well consider monochromatic join edges such as $\{b_i, b_j\}$ for estimating the merging costs, since

$$\begin{aligned} w(\{r_i, b_j\}) + w(\{b_i, r_j\}) &\leq w(\{r_i, b_i\}) + w(\{b_i, b_j\}) + w(\{b_i, b_j\}) + w(\{b_j, r_j\}) \\ &= w(\{r_i, b_i\}) + w(b_j, r_j\}) + 2w(\{b_i, b_j\}). \end{aligned}$$

So, instead of constructing a B2-tree we could try to find a (not necessarily bipartite) spanning tree of minimum weight containing a perfect matching between red and blue vertices. Unfortunately, it is not clear how to construct such a tree in polynomial time.

Michel [20] conjectures that in practice the above merging procedure should yield much better tours than the depth-first-search strategy. We implemented both algorithms and found that indeed, for uniformly distributed random points, merging tours drops the typical tour length from $1.27 \cdot$OPT (worst case) and $1.14 \cdot$ OPT (average) to $1.15 \cdot \text{OPT}$ (worst case) and $1.06 \cdot \text{OPT}$ (average), respectively. Still, it is impossible to prove a better general upper bound, since the worst case example shown in figure 4 also applies to the merging procedure.

2.5 Cycle covers

Let $G = (R \cup B, \mathcal{P}_2(R \cup B))$ denote the complete graph on n red and n blue vertices with Euclidean weight function w. A straight-forward relaxation of the bipartite TSP is the problem of determining a minimum weight bipartite graph on $R \cup B$, where each vertex is of degree two (clearly, by "bipartite" we mean "bipartite with respect to the classes R and B"). Obviously, any such graph decomposes $R \cup B$ into cycles.

Definition 2 *A* cycle cover *of $R \cup B$ is a set $\mathcal{C} = \{C_1, \ldots, C_l\}$ of bipartite cycles $C_i = (V(C_i), E(C_i))$, $E(C_i) \subseteq \{\{r, b\} \mid r \in R, b \in B\}$ for all $i \in \{1, \ldots, l\}$, such that $V(C_i) \cap V(C_j) = \emptyset$ for $i \neq j$ and $\bigcup_{C \in \mathcal{C}} V(C) = R \cup B$.*

We want to use a similar merging procedure as in the last section to combine cycles of a cycle cover into an alternating tour.

Definition 3 *Let $\mathfrak{C}$ denote the set of all bipartite cycles in G.*

(i) We define a weight function d on $\mathcal{P}_2(\mathfrak{C})$ as $d : \mathcal{P}_2(\mathfrak{C}) \longrightarrow \mathbb{R}_{\geq 0}$,

$$\{C, C'\} \mapsto \begin{cases} 0, & \text{if } V(C) \cap V(C') \neq \emptyset \\ \min\{w(e) \mid e \in E(G), & \\ \ e \cap V(C) \neq \emptyset, e \cap V(C') \neq \emptyset\} & \text{otherwise.} \end{cases}$$

(ii) Let $C_1, C_2 \in \mathfrak{C}$ with $V(C_1) \cap V(C_2) = \emptyset$. $C \in \mathfrak{C}$ is a merger *of C_1 and C_2 if $V(C) = V(C_1) \cup V(C_2)$ and $|E(C) \setminus (E(C_1) \cup E(C_2))| = 2$. A* least weight merger *is a merger of minimum weight.*

Consider the following new algorithm for finding an alternating tour in G.

1. Determine a cycle cover $\mathcal{C} = (C_1, \ldots, C_l)$ of minimum total weight

$$\mathrm{OPT}^* := \sum_{i=1}^{l} w(C_i).$$

2. Construct a minimum spanning tree T of the graph $(\mathcal{C}, \mathcal{P}_2(\mathcal{C}))$ with weight function d defined as above.

3. Successively replace adjacent cycles by least weight mergers.

Theorem 4 *The above algorithm can be implemented to produce an alternating tour of length at most*

$$OPT^* + 2d(E(T)) \leq 3OPT$$

in time $O(n^3 \log n)$.

Proof. The first estimate is proven similar to Lemma 4. The second inequality follows from the fact that an alternating tour is a cycle cover containing a spanning tree. The running time of the algorithm is bounded by $O(n^3 \log n)$ since a minimum weight cycle cover may be computed in $O(n^3 \log n)$-time using a capacitated transportation algorithm [23]. □

Another way to efficiently determine a cycle cover $\mathcal{C}$ of minimum weight is linear programming. Linear programming problems can be polynomially solved by the Ellipsoid Method or interior point algorithms (see [17]). For practical computations the Simplex Algorithm, despite its exponential worst-case running time, is most effective. Interestingly, the problem of computing $\mathcal{C}$ allows for a very simple simplex solution.

Theorem 5 *$\mathcal{C}$ can be constructed via solving a linear program such that the simplex method demands only storing entries from $\{0, 1, -1\}$ in the simplex tableau throughout the whole algorithm (objective row excluded).*

Hence the simplex tableau allows for succinct storage. Besides, updating the tableau can be done very quickly, since no multiplication or division operations are needed.

Proof. Let $A = (a_{ij})$ be the *vertex-edge incidence matrix* of the complete bipartite graph $H = (V, E')$ on $V = R \cup B$. Labeling the vertices and edges of H as $v_1, \ldots, v_{2n}$ and $e_1, \ldots, e_{n^2}$, respectively, A is given as

$$(a_{ij}) := \begin{cases} 1, & v_i \in e_j \\ 0 & \text{otherwise} \end{cases} \quad \text{for all } i \in \{1, \ldots, 2n\},\ j \in \{1, \ldots, n^2\}.$$

We define $H_x := (R \cup B, \{e_i \in E' \mid x_i = 1\})$ to be the subgraph of H induced by a vector $x \in \{0,1\}^{n^2}$ and notice that Ax is a vector, listing the vertex degrees of H_x. A cycle cover is determined by $x \in \{0,1\}^{n^2}$ such that all components of Ax are equal to 2. Hence our objective is to find $x \in \{0,1\}^{n^2}$ minimizing $\sum_{e_i \in E} w(e_i)x_i$ subject to the constraint $Ax = \mathbf{2}$ (where $\mathbf{2}$ abbreviates the n^2-dimensional vector $(2, \ldots, 2)^T$). It is important to recognize that A is *totally unimodular*, i.e. the determinant of any square submatrix of A is 0, 1 or -1. In fact, we claim that the vertex-edge incidence matrix m of *any* bipartite undirected graph is totally unimodular. To prove this, consider an arbitrary square submatrix M' of size $l \times l$. If M' contains columns with at most one nonzero entry we can successively develop M' with respect to these, ending up with

a) an $l' \times l'$ matrix where each column has exactly two nonzero entries,

b) a 1×1 matrix or

c) calculating $\det M'$ to be zero.

In the latter cases we are done, so we can assume that our original matrix M' is described by situation a). This means that M' corresponds to a graph G' with l vertices and l edges. Since any tree in G' has at most $l-1$ edges, G' contains a cycle. G' is bipartite, so this cycle is of even length, and its edges split into two perfect matchings M_1 and M_2. But now the column vectors corresponding to M_1 and M_2 add up to equal sums, and we see that M' is singular. Hence $\det M' = 0$ and the claim is proved. The following Lemma provides ways of constructing totally unimodular matrices from a given totally unimodular matrix (see [23], page 540 for a proof of the last point in the Lemma).

Lemma 5 *Let A be a matrix. The following statements are equivalent.*

1. *A is totally unimodular.*
2. *The transpose of A is totally unimodular.*
3. *(A, I) is totally modular.*
4. *A matrix obtained by deleting a row (column) of A is totally unimodular.*
5. *A matrix obtained by multiplying a row (column) of A by -1 is totally unimodular.*
6. *A matrix obtained by interchanging two rows (columns) of A is totally unimodular.*
7. *A matrix obtained by duplicating rows (columns) of A is totally unimodular.*
8. *A matrix obtained by a pivot operation on A is totally unimodular.*

Let us consider the set $P := \left\{ x \in \mathbb{R}^{n^2}_{\geq 0} \mid Ax = \mathbf{2}, x_i \leq 1 \text{ for all } i \in \{1, \ldots, n^2\} \right\}$. We introduce non-negative slack variables $y_1, \ldots, y_{n^2}$ to replace the inequality constraints by $x_i + y_i = 1$ and define $M := \begin{pmatrix} A & \mathbf{0} \\ I & I \end{pmatrix}$, where I denotes the $n^2 \times n^2$ unity matrix. Now P can be written as

$$P = \{ x \in \mathbb{R}^{n^2}_{\geq 0} \mid M \tbinom{x}{y} = (2, \ldots, 2, 1, \ldots, 1)^T \}.$$

We can use the Simplex Algorithm to find $x \in P$ minimizing $\sum_{e_i \in E} w(e_i) x_i$. Recall that the Simplex Algorithm works by scanning *basic feasible solutions* $\tbinom{x}{y}$ of $M\tbinom{x}{y} = (2, \ldots, 2, 1, \ldots, 1)^T$ until the optimum is attained. This is done by selecting a maximum number of linear independent columns of M as a matrix M', such that the unique solution $\tbinom{x'}{y'}$ to the subsystem $\tbinom{x'}{y'} = M'^{-1}(2, \ldots, 2, 1, \ldots, 1)^T$ is a 0/1-vector. Adding zero-components to the vector of *basic components* $\tbinom{x'}{y'}$ completes the basic feasible solution. We observe that M and M' can be obtained

from A via operations described in Lemma 5. Hence M and M' are totally unimodular matrices, and from Cramer's rule it follows that M'^{-1} is integral. But this means that all basic feasible solutions – including the optimal – are vectors of integers, too. Thus, $\mathcal{C}$ can indeed be constructed as the graph H_x induced by an optimal solution to the linear program minimize $\sum_{e_i \in E} w(e_i)x_i$ subject to $Ax = \mathbf{2}$, $0 \leq x_i \leq 1$ for all $i \in \{1, \ldots, n^2\}$.
For the second part of the proof we note that M is turned into a simplex tableau by the following steps.

- Delete an arbitrary of the first $2n$ rows of M to obtain a matrix M' of full row rank (A is of rank $2n-1$ since we can pick at most $2n-1$ edges of our bipartite graph H without creating a cycle).
- Choose a basic feasible solution $\binom{x}{y}$ of $M'\binom{x}{y} = (2, \ldots, 2, 1, \ldots, 1)^T$. Then turn the corresponding linear independent columns into unit vectors by pivoting. Deleting these unit vectors yields a matrix M''. Lemma 5 ensures that M'' is totally unimodular.
- Complete the tableau by adding the objective row and a column containing the values of the basic components of the basic feasible solution. Due to feasibility, this value column consists of 0/1-entries only.

Since the Simplex Algorithm works by pivoting, M'' stays totally unimodular throughout the whole procedure. Feasibility ensures that the value column stays a 0/1-vector. Thus all entries of the tableau – except for those of the objective row – are $\in \{0, 1, -1\}$. □

In computational experiments with random points, the cycle cover algorithm shows a superior behavior with respect to the relative quality of the constructed alternating tours. We implemented two different ways of choosing an order of merging. Version 1 is similar to Kruskal's algorithm: the edges of G are sorted according to their weights. We then go through them one by one, starting from the lightest edge, and perform a merging step whenever we find that the edge under consideration connects two distinct cycles. The second version is inspired by Prim's algorithm: successively the largest cycle is combined with the other cycles via lightest edges. (Actually, in both versions we did not consider all edges of G but only those connecting a red vertex to a blue one.) The alternating tours proved to be typically shorter than a maximum of $1.08 \cdot \mathrm{OPT}$. The average tour lengths were less than $1.02 \cdot \mathrm{OPT}$.

There are some reasons why the cycle cover algorithm should typically show a much better behavior than guaranteed by the worst case bound of $3 \cdot \mathrm{OPT}$.

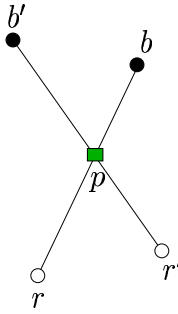
- If the cycles are large, the number of edges connecting them is small. So the length of these edges should also be small. On the other hand, if the cycles are small, our procedure is similar to the nearest insert algorithm

for the ordinary Euclidean TSP, which is proven to construct tours shorter than $2 \cdot \mathrm{OPT}$.

- Let S be an optimal alternating tour and denote by $\tilde{S}$ the multigraph obtained by replacing each vertex by the cycle from an optimum cycle cover $\mathcal{C}$ it lies in. Each cycle contains at least 4 vertices. The minimum spanning tree T constructed in step 2 thus has less than $\lfloor \frac{n}{2} \rfloor$ edges. Since $\tilde{S}$ contains a spanning tree we can find an injective mapping $\varphi : E(T) \to E(S)$ with $d(E(T)) \leq w(\varphi(E(T))$. So, for instance, $d(E(T)) > \frac{1}{2} \cdot \mathrm{OPT}$ would imply that less than a quarter of the edges in an optimal tour are responsible for more than half of its total length.

- Call a cycle $C \in \mathcal{C}$ "good" with respect to an optimal tour S if S traverses C, i.e. if $|E(C) \cap E(S)| \geq |E(C)| - 1$. We collect all good cycles in the set $\mathcal{C}_g$ and define $\mathcal{C}_b$ as the set of "bad" cycles. Since S passes through a spanning tree of $(\mathcal{C}, \mathcal{P}_2(\mathcal{C}))$ and all edges – except for one – of each cycle from $\mathcal{C}_g$, we have $d(E(T)) + \frac{1}{2}\sum_{C \in \mathcal{C}_g} w(C) < \mathrm{OPT}$. Hence, we get a tour of length at most $2 \cdot \mathrm{OPT}$ if $\mathcal{C}$ consists of good cycles only. On the other hand, every bad cycle has degree at least 4 in $\tilde{S}$, thus S traverses at least $|\mathcal{C}_b|$ additional edges, giving $d(E(T)) + \frac{1}{2}\sum_{C \in \mathcal{C}_g} w(C) + w(A) < \mathrm{OPT}$ for a subset A of $E(S)$ with $|A| = |\mathcal{C}_b|$.

- Edges of distinct cycles are geometrically non-intersecting.



For if there are edges $\{r, b\} \in C$, $\{r', b'\} \in C'$ such that the corresponding line segments in the plane intersect in a point p, the triangle inequality gives

$$w(\{r, b'\}) + w(\{r', b\}) \leq w(\{r, p\}) + w(\{p, b'\}) + w(\{r', p\}) + w(\{p, b\}) = w(\{r, b\}) + w(r', b'\}).$$

So C and C' can be united into a single cycle of weight $\leq w(C) + w(C')$. The inequality is strict if $p \notin \{r, b, r', b'\}$.

Moreover, it is impossible that other than very small cycles are contained in another cycle. Clearly, the fact that cycles are confined in distinct regions of the plane should boost their probability of being "good", especially if $d(E(T))$ is large.

Led by the deceptive intuition that it is impossible for cycles and join edges from T to simultaneously be of length about OPT we made several fruitless attempts to prove a worst case bound of 2OPT. However, the example showing that the worst case factor of 3 can indeed be attained looks pretty familiar.

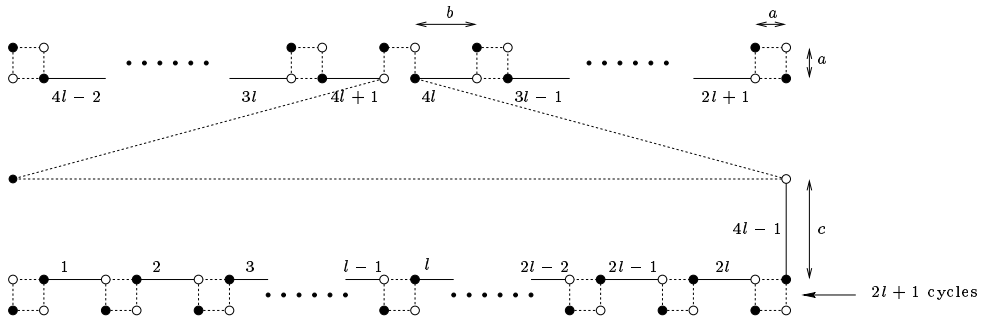


Figure 7: Optimal cycle cover and join edges; the numbers indicate the order of merging.

Lemma 6 *The worst case bound of* $3 \cdot OPT$ *in Theorem 4 is tight.*

Proof. Picture 3 parallel line segments of equal length, separated by a distance c. On each of the outer segments we arrange $n-1$ red and $n-1$ blue points as $2l+1$ 4-point squares of sidelength a equidistantly b length units apart from each other. The remaining single red and blue point are placed at a distance $2l(a+b)+a$ on the middle line segment. For appropriate choices of a, b and c, $a := \frac{1}{l}$, $b := 1$ and $c := 3$, say, an optimal cycle cover looks as indicated by the dotted lines in figure 7. If we merge the cycles in the order specified by the numbers in the figure, after $4l-2$ steps the cycles constructed will obviously look as shown in figure 8.

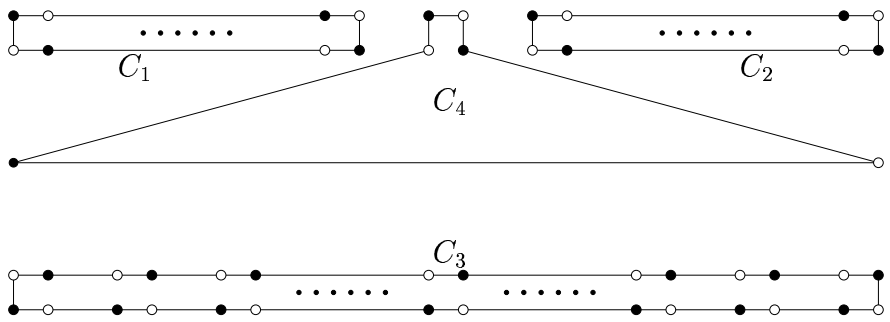


Figure 8: Situation right before the last three merging steps.

We can calculate the weight of the depicted cycles as

$$\begin{aligned} w(C_1) &= w(C_2) = (l-1)\cdot 2a + (l-2)\cdot 2b + 2a > 2lb - 4b \\ w(C_3) &= (2l+2)\cdot 2a + 2l\cdot 2b > 4lb \\ w(C_4) &> 2((2l+1)\cdot a + 2lb) > 4lb, \end{aligned}$$

hence $\sum_{i=1}^{4} w(C_i) > 12lb - 8b = 12l - 8$ by our choice of b. The following observation is an easy consequence of the triangle inequality.

Observation: *If C is a least weight merger of C_i and C_j and e_i, e_j are the heaviest edges of C_i and C_j, respectively, then*

$$w(C) \geq w(C_i) + w(C_j) - 2\min\{w(e_i), w(e_j)\}.$$

Hence, in each of the remaining three merging steps the overall length can decrease by at most $2b$, so the length of the constructed tour C can be estimated as $w(C) > 12l - 14$. The alternating tour S illustrated in figure 9, on the contrary has length

$$\begin{aligned} 2 \cdot ((2l+1) \cdot 3a + 2lb + 2c) &= 4lb + 4c + (2l+1) \cdot 6a \\ &= 4l + 24 + 6/l \end{aligned}$$

by the choice of a, b and c.

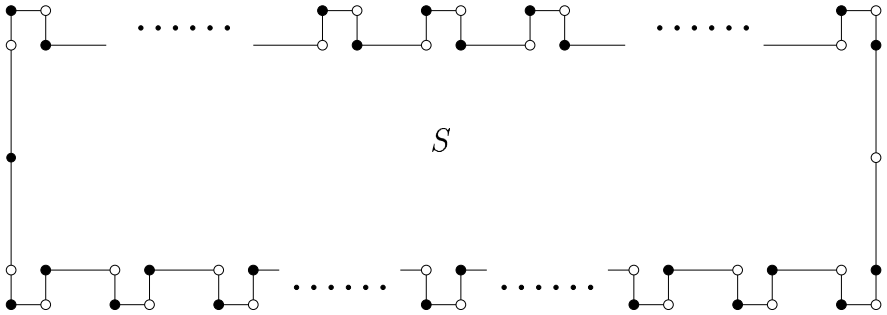


Figure 9: A short tour.

Thus

$$\frac{w(C)}{\mathrm{OPT}} \geq \frac{w(C)}{w(S)} > \frac{12l-14}{4l+24+6/l} = 3 - o(1) \xrightarrow{l \to \infty} 3,$$

so the worst case factor can be approached arbitrarily closely. □

Note that only version 1 of the cycle cover algorithm can choose a merging order according to figure 7. The second version will successively enlarge C_4 and thus is likely to construct an optimal tour.

Chapter 3

Computational results

The following tables list some experimental data we computed to test the algorithms presented in this chapter in practice. We generated random instances of $20 \leq 2n \leq 180$ points with coordinates chosen from the integers in $[1, 500]$. The tour lengths are not given in absolute but as relative values with respect to some OPT* which is the maximum of the values of an optimal cycle cover, a 1-tree and a B2-tree. The numbers in brackets indicate how many times an algorithm produced the shortest tour.

3.1 Uniformly distributed points

No. of points	20	40	60	80
No. of instances	100	100	100	100
Max. Cycle-tour V.1	1.122	1.117	1.097	1.070
Max. Cycle-tour V.2	1.121	1.141	1.160	1.067
Max. B2-tour (Merge)	1.156	1.168	1.118	1.136
Max. Matching-tour	1.296	1.224	1.195	1.213
Max. B2-tour (Double tree)	1.249	1.370	1.302	1.278
Ave. Cycle-tour V.1	1.025 (70)	1.024 (77)	1.020 (74)	1.021 (60)
Ave. Cycle-tour V.2	1.026 (72)	1.026 (62)	1.023 (60)	1.022 (57)
Ave. B2-tour (Merge)	1.051 (10)	1.062 (5)	1.063 (1)	1.067 (0)
Ave. Matching-tour	1.095 (6)	1.111 (1)	1.112 (0)	1.113 (0)
Ave. B2-tour (Double tree)	1.105 (2)	1.136 (1)	1.144 (0)	1.150 (0)
Max. No. of cycles in a cycle cover	5	9	11	15
Ave. No. of cycles in a cycle cover	2.7	4.2	5.9	7.5

No. of points	100	120	140	160
No. of instances	100	100	100	33
Max. Cycle-tour V.1	1.043	1.085	1.040	1.040
Max. Cycle-tour V.2	1.063	1.085	1.050	1.044
Max. B2-tour (Merge)	1.144	1.222	1.105	1.116
Max. Matching-tour	1.192	1.183	1.177	1.183
Max. B2-tour (Double Tree)	1.249	1.261	1.217	1.211
Ave. Cycle-tour V.1	1.019 (75)	1.019 (67)	1.018 (64)	1.019 (27)
Ave. Cycle-tour V.2	1.021 (43)	1.020 (48)	1.019 (45)	1.027 (8)
Ave. B2-tour (Merge)	1.067 (0)	1.067 (0)	1.064 (0)	1.071 (0)
Ave. Matching-tour	1.117 (0)	1.116 (0)	1.117 (0)	1.118 (0)
Ave. B2-tour (Double tree)	1.146 (0)	1.150 (0)	1.143 (0)	1.152 (0)
Max. No. of cycles in a cycle cover	15	19	22	23
Ave. No. of cycles in a cycle cover	8.9	10.3	12.2	14.3

We see that the cycle-tour algorithm shows a better worst-case and average case behavior than all the other algorithms we discussed. Moreover, version 1 of the algorithm seems slightly superior to version 2, although we know that it can attain a worst-case approximation factor of 3 which is not clear for version 2. Interestingly, the theoretically most efficient B2-tour algorithm (in the basic version which uses a depth first traversal of the B2-tree) yields the weakest results. Since the B2-tour algorithms take about 10 minutes per iteration (on a Pentium II processor with 400 MHz) in the case of 160 points (for a comparison, the matching and cycle cover based algorithms need less than 10 seconds) we chose to stop after 33 iterations. Note that the maximum number of 160 points is also due to the B2-tree algorithm, since this was the greatest number our implementation could handle within a memory of 128 MB.

3.2 Two non-uniform point distribution

Of special interest are the following two non-uniform configurations of random points. Firstly, let us consider instances where all of the blue points are in the left half and all of the red points are in the right half of our grid (see figure 10). This situation is interesting since the optimal tours extremely differ from optimum monochromatic tours.

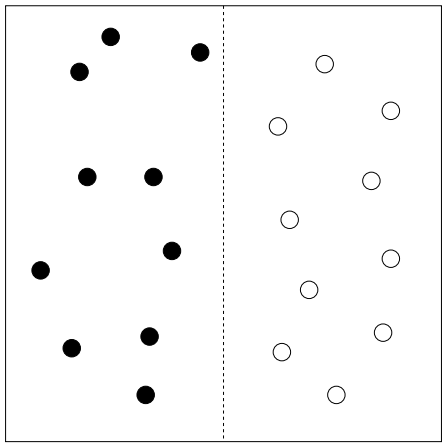


Figure 10: The first non-uniform configuration.

No. of points	20	40	60	80	100
No. of instances	50	50	50	50	10
Max. Cycle-tour V.1	1.0220	1.0067	1.0030	1.0017	1.0010
Max. Cycle-tour V.2	1.0220	1.0081	1.0082	1.0028	1.0015
Max. B2-tour (Merge)	1.0227	1.0104	1.0046	1.0030	1.0013
Max. Matching-tour	1.0464	1.0278	1.0131	1.0101	1.0056
Max. B2-tour (Double Tree)	1.1161	1.0445	1.0313	1.0269	1.0213
Ave. Cycle-tour V.1	1.0073 (44)	1.0026 (41)	1.0016 (49)	1.010 (46)	1.0007 (10)
Ave. Cycle-tour V.2	1.0084 (30)	1.0034 (16)	1.0021 (11)	1.0015 (8)	1.0011 (2)
Ave. B2-tour (Merge)	1.0088 (7)	1.0037 (6)	1.0023 (0)	1.0016 (4)	1.0011 (1)
Ave. Matching-tour	1.0233 (0)	1.0118 (0)	1.0079 (0)	1.0063 (0)	1.0045 (0)
Ave. B2-tour (Double Tree)	1.0367 (0)	1.0244 (0)	1.0206 (0)	1.0165 (0)	1.0158 (0)
Max. No. of cycles in a cycle cover	5	9	13	18	22
Ave. No. of cycles in a cycle cover	4.1	7.7	11.3	15.3	19.9

The relative behavior is similar to the case of uniformly distributed points. Still, the cycle-tour performs best, although the average number of cycles in a cycle cover has roughly doubled, compared to the case of uniformly distributed points. Note that all of the algorithms strongly tend towards the optimum as the number of points increases.

A prominent application of the alternating TSP is the automatic placement of of components on a printed circuit board [21]. Therefore we now consider con-

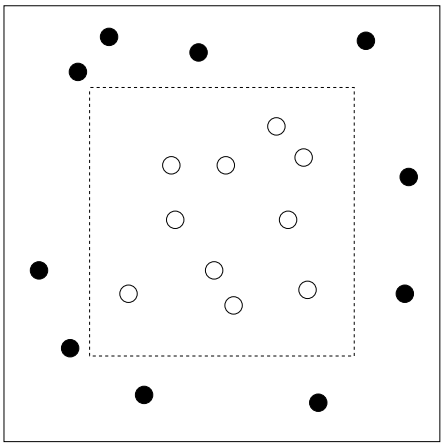


Figure 11: The second non-uniform configuration.

figurations where the red points are confined to coordinates in $\{125, \ldots, 375\}^2$ (representing legal positions on the printed circuit board) while the blue points (representing legal locations from which the components are to be picked) are restricted to the region outside (see figure 11).

No. of points	20	40	60	80	100
No. of instances	50	50	50	50	10
Max. Cycle-tour V.1	1.0652	1.0307	1.0244	1.0136	1.0069
Max. Cycle-tour V.2	1.0764	1.0417	1.0284	1.0157	1.0090
Max. B2-tour (Merge)	1.0903	1.0438	1.0276	1.0186	1.0105
Max. Matching-tour	1.1029	1.0528	1.0490	1.0274	1.0166
Max. B2-tour (Double Tree)	1.1178	1.1278	1.1082	1.0705	1.0686
Ave. Cycle-tour V.1	1.0140 (39)	1.0091 (36)	1.0063 (26)	1.032 (41)	1.0028 (7)
Ave. Cycle-tour V.2	1.0146 (37)	1.0100 (34)	1.0070 (32)	1.0038 (31)	1.0035 (5)
Ave. B2-tour (Merge)	1.0251 (12)	1.0215 (2)	1.0136 (1)	1.0103 (1)	1.0080 (1)
Ave. Matching-tour	1.0289 (10)	1.0191 (7)	1.0121 (11)	1.0115 (1)	1.0088 (0)
Ave. B2-tour (Double Tree)	1.0588 (1)	1.0625 (0)	1.0548 (0)	1.0486 (0)	1.0382 (0)
Max. No. of cycles in a cycle cover	5	9	14	18	20
Ave. No. of cycles in a cycle cover	2.7	4.6	7.5	8.0	10.1

Note that the matching-tour algorithm for these configurations achieves about the same quality as the merge-version of the B2-tour algorithm. However, the cycle-tour algorithm is still unbeaten.

Chapter 4

Random points in the unit square

Let us now turn to the more general setting of the traveling gastronomy critic's problem. Instead of studying the worst case behavior of approximation algorithms, we aim at the delicate task of estimating the typical length of optimum alternating tours to provide the critic with a hint of how much money he should be prepared to spend. Moreover, we want to devise algorithms which with high probability find (nearly) optimal tours. Since fuel prices are instable, we look for procedures that can be efficiently performed in parallel. To give some meaning to the described task, we assign the term "typical" to configurations of points distributed uniformly at random in the unit square. We are interested in how the tour length depends on the number of points almost surely (i.e. with probability tending to 1 as the number of points approaches infinity). So the problem can be stated as follows:

Let $k, n \in \mathbb{N}$ and let $P_0, \ldots, P_{k-1}$ be k sets of vertices, where each P_i consists of n points that are distributed independently and uniformly at random in $(0,1]^2$. Consider the complete graph G on $P_0 \cup \ldots \cup P_{k-1}$ with edge weights $w(e_i)$ given by the Euclidean distances of the endpoints of e_i. How does the length of an alternating tour in G depend on n and k almost surely? Can we give a (parallel) algorithm to construct a tour of optimal length almost surely?

The case $k = 1$ is included in the well-known theorem of Beardwood, Halton and Hammersley [7]. So we know that a monochromatic tour through n random points has length $\gamma \cdot \sqrt{n}$ almost surely for some absolute constant γ. The following procedure is easily analyzed to yield a directed cycle of length $O(\sqrt{n})$ which we will refer to as a BHH-tour (see figure 12).

Algorithm BHH-Tour.

- Partition the unit square into $\lceil \sqrt{n} \rceil$ horizontal segments of width $\frac{1}{\lceil \sqrt{n} \rceil}$. Number the segments starting with 0 in a bottom up fashion.
- Connect the points in each even numbered segment from left to right and the points in segments with odd number from right to left.

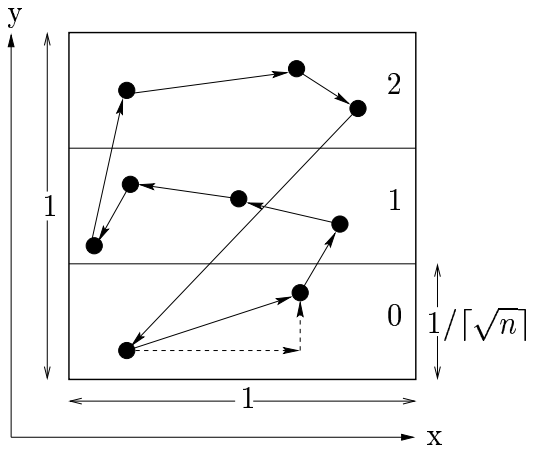


Figure 12: A BHH-tour.

- Join the paths in neighboring segments alternatingly via the rightmost and the leftmost points.

- Add a final edge to close the path to a tour.

Note that this algorithm always gives a tour of length $O(\sqrt{n})$ no matter how the points are distributed, so that $O(\sqrt{n})$ is a valid bound even in the worst case. On the contrary, for $k \geq 2$ it is easy to give a configuration of points forcing an alternating tour to be of length $\Omega(n)$ (see figure 13) whereas a typical tour should be much smaller. We will construct alternating TSP tours similar to the matching method in 2.3, with the matching replaced by a cycle cover and the Christofides- or Arora-tour substituted with a BHH-tour.

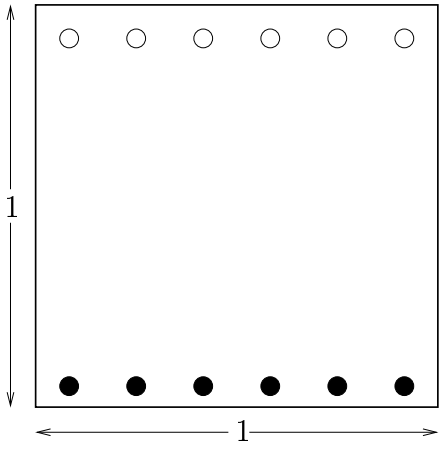


Figure 13: Bad configuration for the alternating TSP.

4.1 An optimal algorithm

Interestingly, we can give a simple sequential algorithm for computing alternating TSP tours of asymptotically optimum length:

Algorithm CYCLE INSERT.

- Find minimum weight perfect matchings M_i between the points of P_i and P_{i+1} for all $i \in \{0, \ldots, k-1\}$ ($P_k := P_0$). $M := \bigcup_{i=0}^{k-1} M_i$ is the edge set of a cycle cover $\mathcal{C} = (C_1, \ldots, C_n)$ of G (in the case $k = 2$ we view all edges of $M := M_0$ as being doubled so that $\mathcal{C}$ consists of 2-cycles).
- Choose one point $\in P_0$ from each $C \in \mathcal{C}$ and let P_0' be the set of selected points. Construct a directed BHH-tour C_0 through P_0'.
- Insert $\mathcal{C}$ into C_0 by replacing each edge $\{x, p'\} \in E(C_i)$, where $x \in P_{k-1}'$, $p' \in P_0'$, with $\{x, N_{C_0}^+(p')\}$, i.e. connect the neighbor of p' from P_{k-1} with the successor of p' in the BHH-tour. Denote the constructed tour by C_M.

Theorem 6 *The algorithm* CYCLE INSERT *produces an* optimal *alternating TSP tour of length* $O(k\sqrt{n \log n})$ *almost surely. It can be implemented to run in time* $O(kn^3)$.

For the proof we need the following result of Ajtai, Komlós, and Tusnády [1].

Theorem 7 (Ajtai,Komlós,Tusnády) *There are absolute constants* c_1, c_2 *such that any minimum weight perfect matching* M_i *between the points of* P_i *and* P_{i+1} *with probability* $1 - o(1)$ *satisfies* $c_1\sqrt{n \log n} < w(M) < c_2\sqrt{n \log n}$.

Thus, in contrast to the monochromatic situation, the average case behavior differs considerably from the worst case situation (confer figure 13). Moreover, the case $k \geq 2$ enforces an additional factor of $\sqrt{\log n}$ compared to the case $k = 1$.

Proof of Theorem 6. Obviously, C_M is alternating. By the triangle inequality we estimate

$$\begin{aligned} w(C_M) &\leq w(C_0) + w(M) \\ \text{and } w(C_0) &\leq w(\text{BHH-tour through } P_0) = O(\sqrt{n}). \end{aligned}$$

Since an optimal alternating tour contains k perfect matchings and the matchings M_i are of minimum weight, Theorem 7 gives

$$kc_2\sqrt{n \log n} > \text{OPT} \geq w(M) = \sum_{i=0}^{k-1} w(M_i) > kc_1\sqrt{n \log n}.$$

Hence $\frac{w(C_0)}{\text{OPT}} = o(1)$ and we deduce that $w(C_M) \leq \text{OPT}(1+o(1)) = O(k\sqrt{n \log n})$. Producing a BHH-tour takes time $O(n \log n)$, a minimum weight perfect matching can be computed in time $O(n^3)$. So the overall running time is $O(kn^3)$. □

4.2 A perfect matching algorithm

It is an open question whether or not the problem of finding a minimum-weight perfect matching in a graph with non-integral edge weights allows efficient parallelization. Therefore we cannot parallelize the CYCLE INSERT algorithm straightway, but we need some further ideas to tackle the problem. As a preparation for the solution presented in section 4.3 we will show that the following procedure for constructing a "light" perfect matching can be parallelized (for convenience of description we consider the points of P_0 as red and those of P_1 as blue):

Algorithm PERFECT MATCHING.

- Subdivide the unit square into small elementary squares and match as many red and blue points as possible inside these small squares. Clearly, some red and blue points in the elementary squares will remain unmatched.
- Construct larger squares by joining every 4 adjacent elementary squares and create new larger matching edges inside of these.
- Continue in this fashion until after t steps all points are matched.

For analyzing this procedure we will need the following large deviation inequality which is an easy consequence of the Angluin-Valiant inequalities [2].

Lemma 7 *Let $X_1, X_2, \ldots, X_n$ be independent random variables having the Bernoulli distribution with identical parameter and define $S := X_1 + X_2 + \ldots + X_n$. Then for every $\alpha > 0$*

$$\mathbb{P}\left[|S - \mathbb{E}(S)| > \alpha\sqrt{\mathbb{E}(S)}\right] \leq 2\exp(-\alpha^2/3).$$

Now, fix a square $Q \subseteq (0,1]^2$ and let $X_1, \ldots, X_n$ be random variables indicating which of the blue points belong to Q. Since all points are distributed uniformly at random, the variables X_i are independent Bernoulli variables with identical parameter proportional to the area of Q. Putting

$$S_Q := \sum_{i=1}^{n} X_i,$$

Lemma 7 gives

$$\mathbb{P}\left[|S_Q - \mathbb{E}(S_Q)| > \alpha\sqrt{\mathbb{E}(S_Q)}\right] \leq 2\exp(-\alpha^2/3).$$

A similar inequality holds for the red points. In step 0 we partition the unit square into q^2 equally sized squares $Q_1, \ldots, Q_{q^2}$, where q is a constant depending

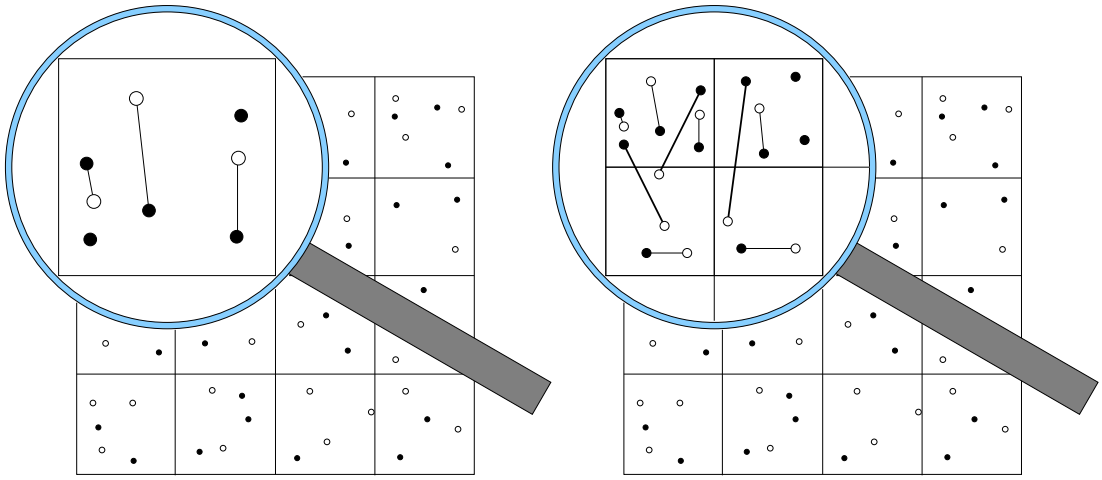


Figure 14: Step 0 and step 1 of the perfect matching procedure.

on n to be fixed later. Provided that $\alpha \to \infty$ as $n \to \infty$, Lemma 7 ensures that each square Q_i contains at least

$$\mathbb{E}(S_{Q_i}) - \alpha\sqrt{\mathbb{E}(S_{Q_i})} = \frac{n}{q^2} - \alpha\frac{\sqrt{n}}{q}$$

red and blue points that can be joined to form a perfect matching. The weight of each matching edge is bounded by the length of the square diagonal which is $\sqrt{2}/q$. Hence the overall weight of the matching is at most $q^2(\frac{n}{q^2} - \alpha\frac{\sqrt{n}}{q}) \cdot \frac{\sqrt{2}}{q} = \sqrt{2}(\frac{n}{q} - \alpha\sqrt{n})$. A maximum number of $n - q^2(\frac{n}{q^2} - \alpha\frac{\sqrt{n}}{q}) = \alpha q\sqrt{n}$ points of each class remain isolated. We successively combine 4 squares to one large square and proceed by induction. In the ith step we have $q^2/4^i$ squares, each containing at least $n \cdot 4^i/q^2 - \alpha\sqrt{n} \cdot 2^i/q$ points of both classes. By creating additional edges of weight $\leq 2^i\sqrt{2}/q$ we can reduce the number of isolated points to

$$n - \frac{q^2}{4^i}\left(\frac{n \cdot 4^i}{q^2} - \alpha\frac{\sqrt{n} \cdot 2^i}{q}\right) = \alpha q\sqrt{n}2^{-i}.$$

To estimate the contribution of the new matching edges we take the difference between the number of isolated points before and after the ith step, i.e. $\alpha q\sqrt{n}2^{-i+1} - \alpha q\sqrt{n}2^{-i} = \alpha q\sqrt{n}2^{-i}$, and multiply with the length of the current square diagonal $2^i\sqrt{2}/q$. The procedure terminates after t steps when $q^2 4^{-t} \leq 1$, i.e. after $\log_2 q$ iterations. The overall weight of the matching is $\leq \sqrt{2}(n/q + \log_2 q \cdot \alpha\sqrt{n})$. This is minimum if $n/q = \log_2 q \cdot \alpha\sqrt{n}$, i.e. if $q \log_2 q = \sqrt{n}/\alpha$. Note that for every square considered we have to apply Lemma 7 once for each color, this makes a total of k times for our k classes P_i. To ensure probability tending to 1 we thus need

$$k \cdot 2\exp(-\tfrac{\alpha^2}{3})q^2\left(1 + \tfrac{1}{4} + \tfrac{1}{4^2} + \ldots + \tfrac{1}{4^t}\right) \xrightarrow{n\to\infty} 0,$$

so $\exp(-\frac{\alpha^2}{3})kq^2 \to 0$, which is satisfied if $\alpha \geq \sqrt{3\ln(kq^2) + \ln\ln n}$. Putting

$$q := \left\lceil \frac{\sqrt{n}}{\ln n} \right\rceil \text{ and } \alpha := \sqrt{3\ln(kn)},$$

we get the following theorem.

Theorem 8 *Inserting a cycle cover obtained by k runs of the perfect matching procedure into a BHH-tour yields an alternating TSP tour of length bounded by $(1+o(1))k\sqrt{n\ln(kn)}\ln n$ almost surely.*

A comparison with theorem 6 yields an estimate of the approximation quality achieved by the PERFECT MATCHING algorithm.

Corollary 1 *The* PERFECT MATCHING *algorithm approximates a perfect matching of minimum weight within as factor of $O(\ln n)$.*

4.3 The parallel algorithm

Now we give a parallel counterpart of the procedure used in Theorem 8. Our algorithm will make use of concurrent reading of the same variables but will not simultaneously write to identical memory locations. So the computational model we refer to is a CREW PRAM. We will present two versions of our algorithm. The first one produces a tour of exactly the quality promised in Theorem 8 and requires k^2n^2 processors to run in time $O(\log^2(kn))$ or $k^2n^2/\log^2(kn)$ processors for a running time of $O(\log^3(kn))$. The length of the tour computed by the second version satisfies a slightly weaker bound of $(2+o(1))k\sqrt{n\ln(kn)}\ln n$. This is due to the fact that Version 2 inserts paths instead of cycles into the BHH-tour. In the worst case the tour length can be twice the weight of the paths and hence about twice the weight of the cycle cover. On the other hand, we will see that Version 2 terminates within $O(\log^2 n)$ steps on kn^2 processors and needs only n^2 processors to achieve a running time of $O(k\log^2 n)$. The parallel algorithm consists of two parts. We begin with a high-level description.

- **Algorithm** PARALLEL-TSP (high level).

(I) Matching Part:

  - Start with elementary squares.
  - In each square Q (in parallel):

    (I.i) Sort points $\in P_i$, $i \in \{0,\ldots,k-1\}$ w.r.t. their coordinates.

    (I.ii) Let $z_i := \min\{\#\text{points} \in P_i \cap Q, \#\text{points} \in P_{i+1} \cap Q\}$, $i \in \{0,\ldots,k-1\}$, Version 2: $i \in \{0,\ldots,k-2\}$.

(I.iii) Match the first z_i points from $P_i \cap Q$ and $P_{i+1} \cap Q$, $i \in \{0, \ldots, k-1\}$, Version 2: $i \in \{0, \ldots, k-2\}$.

– Enlarge the squares and repeat the above steps considering unmatched points only.

(II.) Tour Part:

– View the matching edges as forming the edge set of
  * a cycle cover (Version 1),
  * n alternating paths (Version 2).

(II.i) Version 1 only: Determine a matrix $M = (m_{ij})_{1 \le i,j \le kn}$ indicating whether points i and j lie on a common cycle.

(II.ii) Select one point $\in P_0$ from each cycle (path) into a set P_0'.

– Consider horizontal segments of width $\sqrt{n}$; in each segment (in parallel):

(II.iii) Sort points from P_0' in segments No. 0,2,4,... in ascending order w.r.t. their coordinates; sort points from P_0' in segments No. 1,3,5,... in descending order w.r.t. their coordinates.

(II.iv) Insert matching edges.

Let us study the parts is detail. We assume that all points are assigned distinct numbers from 1 to kn. At first we focus on sorting and counting the points in each square of side length s. Suppose, the points of some $P \in \{P_0, \ldots, P_{k-1}\}$ are numbered from 1 to n and the x and y coordinates of point i are denoted x_i and y_i, respectively. Let us further assume that no two points have identical coordinates. This is how we do the sorting and counting.

Algorithm PARALLEL TSP (I.i)

`in parallel: for all unmatched` $\{i,j\} \in \mathcal{P}_2(\{1,\ldots,n\})$ `do`

`if` $(\lceil \frac{x_i}{s} \rceil = \lceil \frac{x_j}{s} \rceil$ `and` $\lceil \frac{y_i}{s} \rceil = \lceil \frac{y_j}{s} \rceil)$ {i `and` j `belong to the same square`}

`then if` $(x_i > x_j)$ `or` $(x_i = x_j$ `and` $y_i > y_j)$ `then` $c_{ij} := 1$

`in parallel: for all unmatched` $i \in \{1,\ldots,n\}$ `do`

`begin`

$Q_i := (\lceil \frac{x_i}{s} \rceil, \lceil \frac{y_i}{s} \rceil)$ {`point` i `belongs to square` Q_i}

$p_i := 1 + \sum_{j=1}^{n} c_{ij}$ {p_i :=`sorted position of point` i}

$l_{Q_i,p_i} := i$ {l_{Q_i} :=sorted list of points from P in square Q_i}

$z_{Q_i,i} := 1$

end

in parallel: for each square Q do

$z_Q := \sum_{i=1}^{n} z_{Q,i}$ {z_Q := # points from P in square Q}

In this way we obtain for each square Q a sorted list $(l_{Q,1}, l_{Q,2}, \ldots, l_{Q,z_Q})$ of the points from P it contains. For comparing the coordinates in parallel we need n^2 processors. The summation of n terms can be done in $\log_2 n$ steps on $\lceil n/2 \rceil$ processors. So the sorting and counting procedure takes time $O(\log n)$ on n^2 processors for one class of points. To get lists $(l^P_{Q,1}, \ldots, l^P_{Q,z^P_Q})$ for all $P \in \{P_0, \ldots, P_{k-1}\}$ we can either run steps (I.i) k times, thereby increasing the running time to $O(k \log n)$, or perform them in parallel on kn^2 processors. We proceed as follows to construct directed matching edges between points from P_j and P_{j+1}.

Algorithm PARALLEL TSP (I.ii) and (I.iii)

in parallel: for all Q do

begin

$z^{P_j} := \min\{z^{P_j}_Q, z^{P_{j+1}}_Q\}$

in parallel: for $i := 1$ to z^{P_j} do

$N^+(l^{P_j}_{Q,i}) := l^{P_{j+1}}_{Q,i}$; $N^-(l^{P_{j+1}}_{Q,i}) := l^{P_j}_{Q,i}$ {out- and in-neighbors}

end

To regard all classes of points we again can choose to perform (I.ii) and (I.iii) k times or to use kn^2 processors. The parts (I.i), (I.ii) and (I.iii) have to be repeated for all side lengths of elementary squares, i.e. $O(\log n)$ times. Hence the overall running time required for the matching part of our parallel algorithm is $O(k \log^2 n)$ on n^2 processors and $O(\log^2 n)$ on kn^2 processors, respectively.

Version 1 of the tour part starts with computing a matrix M indicating whether points belong to a joint cycle. To achieve polylogarithmic running time we first determine an auxiliary matrix $A = (a_{x,i}) \in V^{kn \times \lceil \log(kn) \rceil}$, where for $x \in V$ and $i \in \{0, \ldots, \lceil \log(kn) \rceil\}$ we have $a_{x,i} = y$ if and only if x and y lie on the same cycle and we can reach y from x by traversing 2^i edges in the prespecified direction. Here is how A and M can be constructed.

Algorithm PARALLEL TSP (II.i)

```
in parallel: for all x ∈ V do

    a_{x,0} := N^+(x)

for i := 1 to ⌈log(kn)⌉ do

    in parallel: for all x ∈ V do

        a_{x,i} := a_{a_{x,i-1},i-1}

in parallel: for all x ∈ V do

    m_{x,x} := 1

for i := ⌈log(kn)⌉ downto 0 do

    in parallel: for all x, y ∈ V do

        if m_{x,y} = 1 then m_{x,a_{y,i}} := 1
```

The time required for (II.i) is $O(\log(kn))$ on $(kn)^2$ processors. Kindervater, Lenstra and Shmoys [16] study a similar procedure for implementing the nearest addition heuristic for the Euclidean non-alternating TSP in parallel. Using their ideas it is possible to compute M with only $(kn)^2/\log^2(kn)$ processors on the cost of a weaker bound of $O(\log^2(kn))$ on the running time. Note that the entries of A are not distinct, so that several processors may simultaneously try to assign to the same entry of M the value 1. We can prevent this by replacing the last loop with

```
for i := ⌈log(kn)⌉ downto 0 do

begin
    in parallel: for all x, y ∈ V do

        if m_{x,y} = 1 then m^y_{x,a_{y,i}} := 1

    in parallel: for all x ∈ V do

        if Σ_{y∈V} m^y_{x,a_{y,i}} > 0 then m_{x,a_{y,i}} := 1
end
```

This increases the running time by a factor of $O(\log(kn))$. Now we want to find the set P_0'. In Version 2 of our algorithm this is easy since $P_0' = P_0$. For Version 1 we do the following.

Algorithm PARALLEL TSP (II.ii)

`in parallel: for all` $x, y \in P_0$ `do`

`if` $x < y$ `then if` $m_{x,y} = 1$ `then` $q_y^x := 1$

`in parallel: for all` $y \in P_0$ `do`

`if` $\sum_{x \in P_0} q_y^x = 0$ `then` $q_y := 1$ $\{P_0' := \{y \mid q_y = 1\}\}$

The steps (II.ii) take $O(\log n)$ time on n^2 processors. The sorting procedure (II.iii) for the points from P_0' is similar to the sorting procedure (I.i) in the matching part. After computing

$S_i :=$ No. of segment point $i \in P_0'$ belongs to

$z_{S_i} :=$ number of points from P_0' in segment S_i

$$c_{x,y} := \begin{cases} 1, & \text{if point } x \in S_i \cap P_0' \text{ is to be visited before point } y \in S_i \cap P_0' \\ 0 & \text{otherwise} \end{cases}$$

we can determine the edges of an alternating tour as follows.

Algorithm PARALLEL TSP (II.iv)

`in parallel: for all` $x \in P_0'$ `do`

`begin`

$p_x := \sum_{i=0}^{S_x - 1} z_i + \sum_{y \in P_0'} c_{x,y}$ $\{p_x :=$`sorted position of point` $x\}$

$t_{p_x} := x$ $\{t :$ `tour through` $P_0'\}$

$N^+(N^-(t_{p_x})) := t_{p_x+1}$

`end`

Part (II.iv) runs in time $O(\log n)$ on n processors. All that is left to do is to output the tour edges $\{x, N^+(x)\}$, for all $x \in V$, which can be done in time $O(1)$ on kn processors or in time $O(k)$ on n processors. Thus we have proved the following theorem.

Theorem 9 *Version 1 of the described parallel algorithm runs on a CREW PRAM in time* $O(\log^2(kn))$ *on* $(kn)^2$ *processors and in time* $O(\log^3(kn))$ *on* $(kn)^2/\log^2(kn)$ *processors, respectively and produces an alternating TSP tour of length* $(1 + o(1))k\sqrt{n \ln(kn)} \ln n$ *almost surely. Version 2 needs time* $O(\log^2 n)$ *on* kn^2 *processors and time* $O(k \log^2 n)$ *on* n^2 *processors, respectively to compute an alternating TSP tour of length* $(2 + o(1))k\sqrt{n \ln(kn)} \ln n$ *almost surely.*

4.4 Poisson distributed points

Let us leave the unit square and study alternating tours through other than uniformly distributed random points. We consider n red and n blue random points $p_1, \ldots, p_{2n}$ as being distributed in the 2-dimensional grid according to the Poisson distribution with parameter $\lambda \in \mathbb{N}$ if

$$\mathbb{P}[p_i = (x,y)] = \exp(-\lambda)\frac{\lambda^x}{x!} \cdot \exp(-\lambda)\frac{\lambda^y}{y!}$$

for all $(x,y) \in \mathbb{N}_0^2$ and all $i \in \{1, \ldots, 2n\}$. To be able to estimate the length of a tour, we must restrict our attention to points in a finite region of the grid. The Poisson distribution is sharply concentrated around the expectation (λ, λ). Hence it makes sense to ignore all points with x- or y-coordinate greater than 2λ. Moreover, since the number r of red points can differ from the number b of blue points in $\{0, \ldots, 2\lambda\} \times \{0, \ldots, 2\lambda\}$, we have to be content with alternating tours through $2n' := 2 \cdot \min\{r, b\}$ points. As a trivial bound we observe that the length of such a tour can be no greater than $2n \cdot 2\lambda\sqrt{2}$. Our aim is to show that $12\sqrt{2}n\sqrt{\lambda \ln \lambda + \ln \lambda + \frac{1}{\sqrt{\ln \lambda}}}$ is a valid upper bound. In the book of Barbour, Holst and Janson [6] we find the following estimate.

Lemma 8 *Let X be a random variable having Poisson distribution with parameter $\lambda > 0$. For $m \in \mathbb{N}_0$ we have*

(i) $\mathbb{P}[X = m] \leq \frac{1}{\sqrt{2\pi m}} \exp\left(-(\sqrt{\lambda} - \sqrt{m})^2\right)$,

(ii) $\mathbb{P}[X \geq m] \leq \frac{m+1}{m+1-\lambda}\mathbb{P}[X = m]$, *if* $m + 1 > \lambda$,

(iii) $\mathbb{P}[X < m] \leq \frac{\lambda}{\lambda+1-m}\mathbb{P}[X = m-1]$, *if* $m - 1 < \lambda$.

Theorem 10 *The optimum length of a bipartite Poisson tour is smaller than $12\sqrt{2}n\sqrt{\lambda \ln \lambda + \ln \lambda + \frac{1}{\sqrt{\ln \lambda}}}$ with probability $\geq 1 - 2\left(\frac{1}{\lambda}\right)^{\frac{n}{3c}}$.*

Proof. Let X be a Poisson distributed random variable with parameter $\lambda \in \mathbb{N}$. We choose $d := 2\sqrt{\lambda \ln \lambda} + \ln \lambda - 1$, $\alpha := \sqrt{\frac{n}{\lambda} \ln \lambda}$, and define

$$p := \mathbb{P}[X \in \{\lambda - d, \ldots, \lambda + d\}].$$

By Lemma 7, the square $[\lambda - d, \lambda + d]^2$ with probability $\geq 1 - 2\left(\frac{1}{\lambda}\right)^{\frac{n}{3c}}$ contains at least $p^2 n - \alpha p\sqrt{n}$ points of both colors. So there is an alternating tour of length at most

$$\begin{aligned} &2\left((n' - p^2 n + \alpha p\sqrt{n})2\lambda\sqrt{2} + (p^2 n - \alpha p\sqrt{n})2d\sqrt{2} + 2\lambda\sqrt{2}\right) \\ &\leq 4\sqrt{2}\left(n'\lambda - p^2(n\lambda - nd) + \alpha p\sqrt{n}\lambda\right) \end{aligned}$$

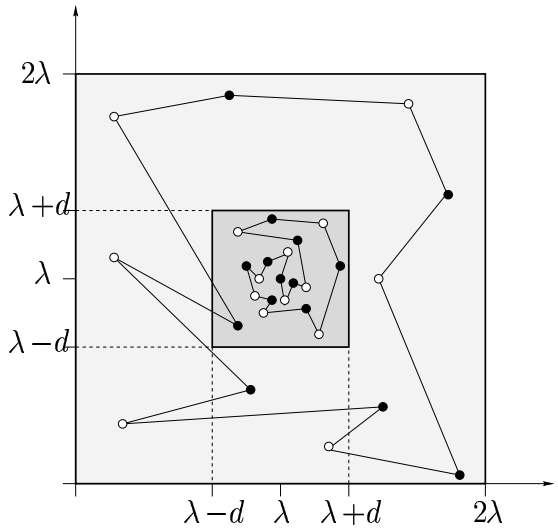


Figure 15: Joining the depicted cycles yields a tour of length as claimed.

(see figure 15). Let us now estimate p. By Lemma 8,

$$\begin{aligned} p &= \mathbb{P}[X \le \lambda + d] - \mathbb{P}[X < \lambda - d] \\ &= 1 - \mathbb{P}[X \ge \lambda + d + 1] - \mathbb{P}[X < \lambda - d] \\ &\ge 1 - \frac{\lambda + d + 2}{d+2} \cdot \frac{\exp\left(-(\sqrt{\lambda} - \sqrt{\lambda + d + 1})^2\right)}{\sqrt{2\pi(\lambda + d + 1)}} \\ &\quad - \frac{\lambda}{d+1} \cdot \frac{\exp\left(-(\sqrt{\lambda} - \sqrt{\lambda - d - 1})^2\right)}{\sqrt{2\pi(\lambda - d - 1)}}. \end{aligned}$$

The following implications hold:

$$\begin{aligned} & -(d+1)^2 < 0 \\ \Rightarrow\ & \lambda^2 - (d+1)^2 < \lambda^2 \\ \Rightarrow\ & 2\sqrt{\lambda^2 - (d+1)^2} < 2\lambda \\ \Rightarrow\ & 2\sqrt{\lambda^2 - (d+1)^2} + \lambda + d + 1 + \lambda - d - 1 < 4\lambda \\ \Rightarrow\ & \sqrt{\lambda + d + 1} + \sqrt{\lambda - d - 1} < 2\sqrt{\lambda} \\ \Rightarrow\ & -\sqrt{\lambda} + \sqrt{\lambda + d + 1} < \sqrt{\lambda} - \sqrt{\lambda - d - 1} \\ \Rightarrow\ & (\sqrt{\lambda} - \sqrt{\lambda + d + 1})^2 < (\sqrt{\lambda} - \sqrt{\lambda - d - 1})^2 \\ \Rightarrow\ & \exp\left(-(\sqrt{\lambda} - \sqrt{\lambda + d + 1})^2\right) > \exp\left(-(\sqrt{\lambda} - \sqrt{\lambda - d - 1})^2\right). \end{aligned}$$

By the choice of d, $-(d+1)^2 = -(2\sqrt{\lambda \ln \lambda} + \ln \lambda)^2 < 0$ and

$$\sqrt{\lambda} - \sqrt{\lambda + d + 1} = \left(\sqrt{\lambda} - \sqrt{\lambda + 2\sqrt{\lambda \ln \lambda} + \ln \lambda}\right)$$

$$\begin{aligned} &= \left(\sqrt{\lambda} - \sqrt{(\sqrt{\lambda} + \sqrt{\ln\lambda})^2}\right) \\ &= \sqrt{\ln\lambda}. \end{aligned}$$

Hence $\exp\left(-(\sqrt{\lambda} - \sqrt{\lambda - d - 1})^2\right) < \exp\left(-(\sqrt{\lambda} - \sqrt{\lambda + d + 1})^2\right) = \frac{1}{\lambda}$, so that

$$p > 1 - \frac{1}{\lambda}\left(\frac{\lambda + d + 2}{(d+2)\sqrt{2\pi(\lambda + d + 1)}} + \frac{\lambda}{(d+1)\sqrt{2\pi(\lambda - d - 1)}}\right).$$

Moreover

$$\begin{aligned} \frac{\lambda + d + 2}{(d+2)\sqrt{2\pi(\lambda + d + 1)}} &= \frac{\sqrt{\lambda + d + 1}}{\sqrt{2\pi}(d+2)} + \frac{1}{(d+2)\sqrt{2\pi(\lambda + d + 1)}} \\ &= \frac{\sqrt{\lambda} + \sqrt{\ln\lambda}}{\sqrt{2\pi}(2\sqrt{\lambda\ln\lambda} + \ln\lambda + 1)} \\ &\quad + \frac{1}{\sqrt{2\pi}(2\sqrt{\lambda\ln\lambda} + \ln\lambda + 1)(\sqrt{\lambda} + \sqrt{\ln\lambda})} \\ &\leq \frac{\sqrt{\lambda} + \sqrt{\ln\lambda}}{\sqrt{2\pi}2\sqrt{\lambda\ln\lambda}} \\ &= \frac{1}{2\sqrt{2\pi}}\left(\frac{1}{\sqrt{\ln\lambda}} + \frac{1}{\sqrt{\lambda}}\right). \end{aligned}$$

Since $\frac{\lambda}{4} > \left(\frac{41}{16}\right)^2 \ln\lambda$, it follows that $\frac{1}{2}\sqrt{\lambda\ln\lambda} > \frac{41}{16}\ln\lambda$, giving $-2\sqrt{\lambda\ln\lambda} - \ln\lambda > -\frac{5}{2}\sqrt{\lambda\ln\lambda} + \frac{25}{16}\ln\lambda$. Hence

$$\begin{aligned} \lambda - d - 1 &= \lambda - 2\sqrt{\lambda\ln\lambda} - \ln\lambda \\ &> \lambda - \frac{5}{2}\sqrt{\lambda\ln\lambda} + \frac{25}{16}\ln\lambda \\ &= \left(\sqrt{\lambda} - \frac{5}{4}\sqrt{\ln\lambda}\right)^2 \end{aligned}$$

and

$$\begin{aligned} \frac{\lambda}{(d+1)\sqrt{2\pi(\lambda - d - 1)}} &\leq \frac{\lambda}{(2\sqrt{\lambda\ln\lambda} + \ln\lambda)\sqrt{2\pi}(\sqrt{\lambda} - \frac{5}{4}\sqrt{\ln\lambda})} \\ &= \frac{\lambda}{2\sqrt{2\pi}\lambda\sqrt{\ln\lambda} + \sqrt{2\pi\lambda}\ln\lambda - \frac{5}{2}\sqrt{\lambda}\ln\lambda - \frac{5}{4}\ln\lambda\sqrt{\ln\lambda}} \\ &\leq \frac{1}{2\sqrt{2\pi\ln\lambda}}. \end{aligned}$$

So we can conclude that

$$p > 1 - \frac{1}{2\lambda\sqrt{2\pi}}\left(\frac{2}{\sqrt{\ln\lambda}} + \frac{1}{\sqrt{\lambda}}\right) > 1 - \frac{1}{2\lambda\sqrt{\ln\lambda}}.$$

Now

$$\begin{aligned} p^2(n\lambda - nd) &> n\left(1 - \frac{1}{\lambda\sqrt{\ln\lambda}} + \frac{1}{4\lambda^2\ln\lambda}\right)(\lambda - 2\sqrt{\lambda\ln\lambda} - \ln\lambda + 1) \\ &> n\left(1 - \frac{1}{\lambda\sqrt{\ln\lambda}}\right)(\lambda - 2\sqrt{\ln\lambda} - \ln\lambda) \\ &> n\left(\lambda - 2\sqrt{\lambda\ln\lambda} - \ln\lambda - \frac{1}{\sqrt{\ln\lambda}}\right). \end{aligned}$$

So the overall length is smaller than

$$\begin{aligned} & 4\sqrt{2}\left(n'\lambda - n\lambda + 2n\sqrt{\lambda\ln\lambda} + n\ln\lambda + \frac{n}{\sqrt{\ln\lambda}} + \sqrt{\frac{n}{\lambda}\ln\lambda}\sqrt{n\lambda}\right) \\ < \; & 4\sqrt{2}n\left(2\sqrt{\lambda\ln\lambda} + \ln\lambda + \frac{1}{\sqrt{\ln\lambda}} + \sqrt{\lambda\ln\lambda}\right) \\ = \; & 12\sqrt{2}n\left(\sqrt{\lambda\ln\lambda} + \ln\lambda + \frac{1}{\sqrt{\ln\lambda}}\right). \end{aligned}$$

□

Apart from the fact that Theorem 10 gives the first non-trivial result for bipartite TSP tours through Poisson distributed points, the upper bound we have proven is not too impressive. This indicates that we need refined methods for constructing short bipartite tours which probably require stronger estimates than those of Lemma 8. So it seems to be a rather involved task to considerably improve the upper bound. Moreover, it is not clear to us how to obtain a non-trivial lower bound.

Chapter 5

Asymmetric bipartite TSP

In the previous chapters our analyses heavily relied on the fact that we chose edge weights to be the Euclidean distances of certain point configurations, so that weights were symmetric and satisfied the triangle inequality. Now let us see how far we can get without either of these assumptions and consider a complete directed graph G on n red and n blue vertices with arbitrary non-negative edge weights. How to find an optimal alternating tour in G?

By the same arguments as in Chapter 2 we can deduce that this problem is no easier than the monochromatic asymmetric TSP, for which it even is NP-hard to construct a tour whose length is within a constant factor of the optimal length. Though this looks frustrating, there is a very simple polynomially solvable relaxation that can often be exploited to obtain near optimal tours.

5.1 The assignment relaxation

For a directed graph $G = (V, E)$, $V = \{1, \ldots, t\}$, $E \subseteq V \times V$, with weight function $w : E \to \mathbb{R}_{\geq 0}$ we define an associate bipartite graph $G' = (V', E')$, as $V' = \{x_1, \ldots, x_t, y_1, \ldots, y_t\}$, $\{x_i, y_j\} \in E' \Leftrightarrow (i, j) \in E$ with edge weights $w'(\{x_i, y_j\}) := w(i, j)$. Since any Hamiltonian cycle in G corresponds to a perfect matching in G', whereas each perfect matching in G' corresponds to a cycle cover of G, we see that the *Assignment Problem (AP)* of finding a minimum weight perfect matching in G' is a relaxation of the (alternating) Asymmetric Traveling-Salesman Problem. An instance on n vertices of the above problems for the monochromatic case can be specified by an $n \times n$ matrix $W = (w_{ij})$, where w_{ij} represents the weight of the edges $(i, j) \in E$ and $\{x_i, y_j\} \in E'$ respectively. In terms of W we can give the following problem formulations:

- **Assignment Problem**: Find a permutation $\varphi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ minimizing $\sum_{i=1}^{n} w_{i\varphi(i)}$.

- **Asymmetric Traveling-Salesman Problem**: Find a cyclic permutation $\varphi : \{1, \ldots, n\} \to \{1, \ldots, n\}$, minimizing $\sum_{i=1}^{n} w_{i\varphi(i)}$.

Note that there are exactly $n!$ assignments, $(n-1)!$ of which are cyclic. So for a matrix W whose entries are drawn independently and uniformly at random from some interval there is a $1/n$ chance of solving the TSP by computing a solution to the AP. In the book on the traveling salesman problem edited by Lawler, Lenstra, Rinnooy Kan and Shmoys [19], Balas and Toth report of 400 computational experiments where they solved both the TSP and its assignment relaxation for problems with $50 \leq n \leq 250$ vertices and edge weights drawn independently from a uniform distribution of the integers over the intervals $[1, 100]$ and $[1, 1000]$. As the outcome of these experiments the average weight of the assignment was found to be 99.2% of the average optimal TSP tour length. Miller and Pekny [22] managed to obtain optimal solutions to random TSP instances on up to 500,000 vertices by scanning the optimal solutions of the assignment problem for a cyclic permutation. In 1995 Frieze, Karp and Reed [14] proved that, given an $n \times n$ weight matrix W such that each entry has probability p_n of being zero, the optimal values $\mathrm{ATSP}(W)$ and $\mathrm{AP}(W)$ are equal almost surely whenever np_n tends to infinity with n. On the other hand, they showed that, if np_n tends to some constant, then $\mathbb{P}[\mathrm{ATSP}(W) \neq \mathrm{AP}(W)] > \varepsilon > 0$ and, for $np_n \to 0$, $\mathbb{P}[\mathrm{ATSP}(W) \neq \mathrm{AP}(W)] \to 1$ almost surely. Let us prove similar results for the bipartite case.

5.2 A positive result

Suppose our n red and n blue vertices are numbered from 1 to n and from $n+1$ to $2n$, respectively. Since we want all edges to connect a red vertex to a blue one we forbid monochromatic edges by viewing the entries w_{ij} of our $2n \times 2n$ weight matrix W as being ∞ for $(i, j) \in \{1, \ldots, n\}^2 \cup \{n+1 \ldots, 2n\}^2$. Thus the problems to study are these:

- **Assignment Problem**:
  Find bijective mappings
  $\varphi_1 : \{1, \ldots, n\} \to \{n+1, \ldots, 2n\}$, $\varphi_2 : \{n+1, \ldots, 2n\} \to \{1, \ldots, n\}$ such that $\varphi := \varphi_1 \cup \varphi_2$ minimizes $\sum_{i=1}^{2n} w_{i\varphi(i)}$.

- **Bipartite Asymmetric Traveling-Salesman Problem**:
  Find bijective mappings
  $\varphi_1 : \{1, \ldots, n\} \to \{n+1, \ldots, 2n\}$, $\varphi_2 : \{n+1, \ldots, 2n\} \to \{1, \ldots n\}$ such that $\varphi := \varphi_1 \cup \varphi_2$ is a cyclic permutation minimizing $\sum_{i=1}^{2n} w_{i\varphi(i)}$.

Denote by $\mathrm{AP}(W)$ and $\mathrm{BATSP}(W)$ the optimal values of solutions to instances specified by W and let $I := \{1, \ldots, n\} \times \{n+1, \ldots, 2n\} \cup \{n+1, \ldots, 2n\} \times \{1, \ldots, n\}$ be the set of indices for which the entries of W are finite.

Theorem 11 *Let (Z_n) be a sequence of random variables over the nonnegative reals and let $\alpha \geq 0$. Let $W = (w_{ij})$ be a $2n \times 2n$ matrix such that $w_{ij} = \infty$ for $(i,j) \notin I$ and w_{ij} is drawn independently from the same distribution as Z_n for $(i,j) \in I$. If $p_n := \mathbb{P}[Z_n \leq \frac{\alpha AP(W)}{2n}]$ satisfies $n \cdot p_n \to \infty$ as $n \to \infty$ then $BATSP(W) \leq (1+\alpha)AP(W)$ almost surely.*

Proof. Let G denote the complete bipartite digraph on $\{1,\ldots,n\} \cup \{n+1,\ldots,2n\}$ with edge-weights given by W. We will use the abbreviation $\beta := \frac{\alpha \mathrm{AP}(W)}{2n}$ and distinguish between "light" and "heavy" edges depending on whether or not their weights exceed β. The proof relies on two facts.

- The number of light edges is large, since $np_n \to \infty$.
- The weights are independently and identically distributed, hence an optimal assignment may be viewed as consisting of two random permutations which induce only $O(\log n)$ cycles (a.s.).

We will show how to construct a (near) optimal assignment with a small number of cycles and how to combine these into a tour via light edges.
Consider for each $(i,j) \in I$ an indicator variable

$$z_{ij} := \begin{cases} 1, & w_{ij} \leq \beta \\ 0 & \text{otherwise.} \end{cases}$$

The z_{ij} are independent, and each z_{ij} is equal to 1 with probability $p := p_n$. The light edges indicated by the z_{ij} are crucial for our construction. Note that once we observed the weight of an edge it may no longer be considered random. To make our construction work we thus have to be careful not to observe too many light edges too early, since otherwise we would loose randomness and fail to perform the last steps. Therefore we decompose the graph induced by the z_{ij} into separate random subgraphs we can consider independently of each other. For the sake of this, let $h \in \mathbb{R}_{>0}$ be such that $(1-h)^5 = 1-p$. For all $k \in \{1,\ldots,5\}$, $(i,j) \in I$ let z_{ij}^k be independent indicator random variables satisfying $\mathbb{P}[z_{ij}^k = 1] = h$ and define $\hat{z}_{ij} := \max\left\{z_{ij}^k \mid k \in \{1,\ldots,5\}\right\}$. Then the $\hat{z}_{ij}$ are independent and

$$\mathbb{P}[\hat{z}_{ij} = 1] = 1 - \mathbb{P}[\hat{z}_{ij} = 0] = 1 - \prod_{k=1}^{5} \mathbb{P}[z_{ij}^k = 0] = 1 - (1-h)^5 = p.$$

So we can identify z_{ij} with $\hat{z}_{ij}$ and view the z_{ij} as being generated by the above construction. For $k \in \{1,\ldots,5\}$ let G_k be the bipartite digraph on $\{1,\ldots,2n\}$ with $(i,j) \in E(G_k) \Leftrightarrow (i,j) \in I$ and $z_{ij}^k = 1$. We call the edges of G_3, G_4, and G_5 out-, in-, and patch-edges respectively. Let $D := (X \cup Y, F)$, $X = \{x_1,\ldots,x_{2n}\}$, $Y = \{y_1,\ldots,y_{2n}\}$ be a bipartite digraph where

$$F := \{(x_i,y_j) \mid \{x_i,y_j\} \in E(G_1')\} \cup \{(y_i,x_j) \mid \{y_i,x_j\} \in E(G_2')\}.$$

The random graphs D, G_3, G_4, G_5 are completely independent. For $v \in X \cup Y$ we denote by

$$N^+(v) := \{w \in X \cup Y \mid (v,w) \in F\}$$

the set of "out-neighbors" of v, and we call $\deg^+(v) := |N^+(v)|$ the "out-degree" of v. Then $\delta := nh$ is the expected out-degree of each vertex in G_i, $i \in \{1,\ldots,5\}$. Our first aim is to cover those vertices by a minimum weight perfect matching that are likely to be incident with an edge of nonzero weight in an optimal assignment for G'. Clearly, the vertices of small out-degree in D are candidates for this "troublesome" set $T' \subseteq X \cup Y$. Therefore we put $T'_{-1} := \{v \in X \cup Y \mid \deg^+(v) \leq \delta/2\}$, choose a minimum-weight matching M_0 in G' that covers T'_{-1} and set $T'_0 := \bigcup M_0$. We also consider vertices as troublesome if many of their neighboring edges have been observed. Hence we proceed to construct T' inductively as follows: for $i \geq 1$ we choose $v \in X \cup Y \setminus T'_{i-1}$ such that $|N^+(v) \cap T'_{i-1}| \geq \delta/4$ and we let M_i be a minimum-weight matching (in G') covering $T'_{i-1} \cup \{v\}$. Since T'_{i-1} is the vertex set of a matching there is a vertex $w \in X \cup Y \setminus (T'_{i-1} \cup \{v\})$ covered by M_i, and we can define $T'_i := T'_{i-1} \cup \{v,w\}$. Our construction terminates after r steps with a set $T' := T'_r$ and a perfect matching $M := M_r$ in $G_{|_{T'}}$, the subgraph of G induced by T'.

Lemma 9 *$|T'| \leq 3ne^{-\delta/8}$ (a.s.).*

Proof Let us assume that we have already shown the following two claims:

- **Claim 1:** $|T'_{-1}| \leq ne^{-\delta/8}$
- **Claim 2:** For $S \subseteq X \cup Y$ with $|S| \leq 3ne^{-\delta/8}$ we have

$$|\,(E(G'_1) \cup E(G'_2)) \cap \mathcal{P}_2(S)| < 2|S| \text{ (a.s.).}$$

Then $|T'_0| \leq 2|T'_{-1}| \leq 2ne^{-\delta/8}$ by Claim 1. By construction of T'_i we have for each $i \in \mathbb{N}_{\leq r}$: $|T'_i| = |T'_0| + 2i$ and $|\,(E(G'_1) \cup E(G'_2)) \cap \mathcal{P}_2(T'_i)| \geq i\delta/4$. Assume for a contradiction that $|T'| > 3ne^{-\delta/8}$. It follows that $|T' \setminus T'_0| > ne^{-\delta/8}$, so $r \geq r_0 := \lfloor ne^{-\delta/8}/2 \rfloor$. But now, $|T'_{r_0}| \leq 2ne^{-\delta/8} + ne^{-\delta/8} = 3ne^{-\delta/8}$ and $|\,(E(G'_1) \cup E(G'_2)) \cap \mathcal{P}(T'_{r_0})| \geq r_0\delta/4 = \Omega(\delta ne^{-\delta/8}) > 6ne^{-\delta/8}$ for n large (since $\delta = nh$ grows with n) $\geq 2|T'_{r_0}|$, which contradicts Claim 2.

Proof of Claim 1. Let $v \in X \cup Y$. We have

$$\begin{aligned}\mathbb{P}[\deg^+(v) \leq \delta/2] &= \mathbb{P}[\deg^+(v) - \delta \leq -\delta/2] = \mathbb{P}[\delta - \deg^+(v) \geq \delta/2] \\ &= \mathbb{P}[e^{\delta - \deg^+(v)} \geq e^{\delta/2}].\end{aligned}$$

By Markov's inequality

$$\mathbb{P}[e^{\delta-\deg^+(v)} \geq e^{\delta/2}] \leq \mathbb{E}[e^{\delta-\deg^+(v)}]/e^{\delta/2} = e^{\delta/2}\mathbb{E}[e^{-\sum_{i=1}^n z_{1i}^1}]$$

(assuming w.l.o.g. that v corresponds to vertex 1 in G_1). Hence

$$\begin{aligned}\mathbb{P}[\deg^+(v) \leq \delta/2] &\leq e^{\delta/2} \prod_{i=1}^{n} \mathbb{E}[e^{-z^1_{1i}}] = e^{\delta/2} \left((1-h) + h \cdot e^{-1}\right)^n \\ &= e^{\delta/2} \left(1 + \frac{\delta(e^{-1}-1)}{n}\right)^n \leq e^{\delta/2} e^{\delta(e^{-1}-1)} = e^{\delta(1/e-1/2)},\end{aligned}$$

since $\delta = hn$ and $\left(1+\frac{a}{n}\right)^n \leq e^a$ for all $a \in \mathbb{R}$. Hence

$$\mathbb{E}[|T_{-1}|] \leq 4ne^{\delta(1/e-1/2)}$$

and

$$P[|T_{-1}| > ne^{-\delta/8}] \leq \frac{\mathbb{E}[|T_{-1}|]}{ne^{-\delta/8}} \leq 4e^{\delta(1/e-1/2+1/8)} = O(e^{-\frac{\delta}{141}}) \to 0,$$

as $n \to \infty$.

Proof of Claim 2. Let $S \subseteq X \cup Y$ with $|S| \leq 3ne^{-\delta/8}$ and let Z abbreviate $|\,(E(G_1') \cup E(G_2')) \cap \mathcal{P}_2(S)|$. Then

$$\mathbb{E}[Z] < \binom{|S|}{2} \cdot 2h < |S|^2 h$$

and

$$P[Z \geq 2|S|] < \frac{|S|^2 h}{2|S|} = |S|h/2 \leq \frac{3}{2} nhe^{-\delta/8} = \frac{3}{2}\delta e^{-\delta/8} = \frac{3}{2} e^{\ln\delta - \delta/8} \xrightarrow{n\to\infty} 0.$$

□

Let $T'_X := T' \cap X$, $T'_Y := T' \cap Y$ be the subsets of troublesome vertices in the partition classes X and Y, and consider the subgraph of D induced by $(X \setminus T'_X) \cup (Y \setminus T'_Y)$. We claim that this subgraph contains a perfect matching $\hat{M}$ (a.s.). Since all edges in D are light, we obtain an assignment of weight $\leq \mathrm{AP}(W) + |\hat{M}| \cdot \beta \leq (1+\alpha) \cdot \mathrm{AP}(W)$ by combining M and $\hat{M}$.

Lemma 10 *$D_{|_{(X\setminus T'_X)\cup(Y\setminus T'_Y)}}$ contains a perfect matching $\hat{M}$ (a.s.).*

Proof. By construction of T', each $x \in \{x_1, \ldots, x_n\} \setminus T'_X$ has at least $\delta/4$ out-neighbors in $\{y_{n+1}, \ldots, y_{2n}\} \setminus T'_Y$, and each $x \in \{x_{n+1}, \ldots, x_{2n}\} \setminus T'_X$ has at least $\delta/4$ out-neighbors in $\{y_1, \ldots, y_n\} \setminus T'_Y$. Moreover, these neighbors can be considered as chosen independently at random. Clearly, similar statements hold for the $y \in Y \setminus T'_Y$. By a theorem of Walkup [27] $D_{|_{\{x_1,\ldots,x_n\}\setminus T'_X \cup \{y_{n+1},\ldots,y_{2n}\}\setminus T'_Y}}$ and $D_{|_{\{x_{n+1},\ldots,x_{2n}\}\setminus T'_X \cup \{y_1,\ldots,y_n\}\setminus T'_Y}}$ contain perfect matchings $\hat{M}_1$, $\hat{M}_2$. Hence $\hat{M} :=$

$\hat{M}_1 \cup \hat{M}_2$ is a perfect matching in $D_{|(X \setminus T'_X) \cup (Y \setminus T'_Y)}$. □

Our next aim is to show that the subgraph of G corresponding to $M \cup \hat{M}$ contains only few cycles and is thus not too far away from a salesman tour. In G the matching $M \cup \hat{M}$ corresponds to a set σ of pairs $\in \{1, \ldots, 2n\}^2$ which we can view as a mapping on $\{1, \ldots, 2n\}$. We claim that σ can be considered as the union of two random bijections $\sigma_1 : \{1, \ldots, n\} \to \{n+1, \ldots, 2n\}$ and $\sigma_2 : \{n+1, \ldots, 2n\} \to \{1, \ldots, n\}$. To see this, we define the equivalence class $[W]$ of a matrix W to consist of all matrices obtained by permuting the first n and the last n columns of W. Since the probability that W has two identical columns is negligibly small we can assume that $[W]$ is a set of $(n!)^2$ distinct and equally likely matrices $W^\pi = (w^\pi_{ij})$, where $\pi = \pi_1 \cup \pi_2$ for permutations $\pi_1 : \{1, \ldots, n\} \to \{1, \ldots, n\}$, $\pi_2 : \{n+1, \ldots, 2n\} \to \{n+1, \ldots, 2n\}$, and $w^\pi_{ij} := w_{i\pi(j)}$. If σ is an assignment for W obtained by the above algorithm then $\pi^{-1} \circ \sigma$ is an assignment of the same weight for W^π. Clearly, $\pi_1^{-1} \circ \sigma_{|\{n+1,\ldots,2n\}}$ and $\pi_2^{-1} \circ \sigma_{|\{1,\ldots,n\}}$ range over all bijections $\{n+1, \ldots, 2n\} \to \{1, \ldots, n\}$ and $\{1, \ldots, n\} \to \{n+1, \ldots, 2n\}$ if π_1, π_2 range over all permutations. Hence it only remains to show that we can assume that $\pi \circ \sigma$ is constructed by the above algorithm for W^π. For this we slightly alter the procedure in the following way: we permute the first n and the last n columns of W into lexicographic order, perform the described steps on the ordered matrix to construct T'_X, T'_Y and σ, and permute back. Now we are ready to prove the following.

Claim 3:

(a) σ has at most $2 \ln n$ cycles (a.s.),

(b) for all $k \in \{1, \ldots, n\}$ there are less than $(np)^k$ cycles of length $2k$ (a.s.),

(c) there are at most $\frac{n}{\sqrt[3]{np}}$ vertices on cycles of length at most $\sqrt{\frac{n}{p}}$ (a.s.).

Proof of Claim 3. (a) σ can be seen as the union of two random bijections $\sigma_1 : \{1, \ldots, 2n\} \to \{n+1, \ldots, 2n\}$ and $\sigma_2 : \{n+1, \ldots, 2n\} \to \{1, \ldots, 2n\}$ generated by a succession of $2n$ decisions: first we choose $\sigma_1(1)$; next we select $\sigma_2(\sigma_1(1))$; if $\sigma_2(\sigma_1(1)) \neq 1$ we proceed by choosing $\sigma_1(\sigma_2(\sigma_1(1)))$, otherwise we select $\sigma_1(2)$ and continue with $\sigma_2(\sigma_1(2))$, and so forth. In each second step we have the chance of completing a cycle. Let

$$J_i := \begin{cases} 1, & \text{if a cycle is completed in step } 2i, \\ 0 & \text{otherwise.} \end{cases}$$

The J_i, $i \in \{1, \ldots, n\}$, are independent random variables with $\mathbb{P}[J_i = 1] = 1/(n-i+1)$, $\mathbb{P}[J_i = 0] = 1 - 1/(n-i+1)$, and $\sum_{i=1}^n J_i$ counts the overall

number of cycles in σ. Hence we have

$$\begin{aligned}\mathbb{E}[\#\text{ cycles in }\sigma] &= \mathbb{E}\left[\sum_{i=1}^{n} J_i\right] = \sum_{i=1}^{n}\mathbb{P}[J_i=1] = \sum_{i=1}^{n}\frac{1}{n-i+1}\\ &= \sum_{i=1}^{n}\frac{1}{i} = (1+o(1))\ln n.\end{aligned}$$

Using the first Angluin-Valiant inequality we conclude

$$\mathbb{P}[\#\text{ cycles in }\sigma > 2\ln n] = O(e^{-\ln n/3}) \xrightarrow{n\to\infty} 0.$$

(b) Let $k \in \{1,\dots,n\}$. For each cycle γ of length $2k$ let

$$J_\gamma := \begin{cases}1, & \text{if } \gamma \subseteq \sigma,\\ 0 & \text{otherwise.}\end{cases}$$

We have

$$\begin{aligned}\mathbb{E}[\#\text{ cycles of length } 2k \text{ in }\sigma] &= \mathbb{E}\left[\sum_{\gamma} J_\gamma\right]\\ &= (\#\text{ cycles of length } 2k \text{ in } I)\cdot\mathbb{P}[J_\gamma=1]\\ &= \binom{n}{k}^2 (k-1)!k!\cdot\frac{((n-k)!)^2}{(n!)^2} = \frac{1}{k}.\end{aligned}$$

Markov's inequality yields $\mathbb{P}[\#\text{cycles of length } 2k > (pn)^k] \le \frac{1}{k(np)^k} \xrightarrow{n\to\infty} 0$.

(c) Similar to (b) we calculate

$$\mathbb{E}[\#\text{ vertices on cycles of length } \le \sqrt{n/p}] = \sum_{k=1}^{\frac{1}{2}\sqrt{n/p}} \frac{1}{k}\cdot 2k = \sqrt{n/p},$$

so, by Markov's inequality

$$\mathbb{P}[\text{(c) is false}] \le \frac{1}{\sqrt[6]{np}} \xrightarrow{n\to\infty} 0.$$

□

Recall that

$$R = \{r_1,\dots,r_n\} := \{1,\dots,n\} \text{ and } B = \{b_1,\dots,b_n\} := \{n+1,\dots,2n\}$$

denote the vertices of G. Let $T := \{i \in R\cup B \mid x_i \in T' \text{ or } y_i \in T'\}$, $T_R := T\cap R$, $T_B := T\cap B$ be the sets of troublesome vertices in G. So far we did not consider

the weight of any edge with both its endpoints in $R \cup B \setminus T$. Hence each such edge still has the probability h of being an out-, in-, or patch-edge. We will use these edges to eliminate the cycles of σ. To succeed in this we have to make sure that no cycle has too many vertices in T.

Lemma 11 *No cycle of σ has more than $\frac{1}{20}$ of its vertices in T (a.s.).*

Proof. By Lemma 9, $|T| \leq 3ne^{-\delta/8}$ (a.s.), so $t := |T_R| \leq \frac{3}{2}ne^{-\delta/8}$ (a.s.). Let Z denote the number of cycles in σ with at least $\frac{1}{40}$ of their vertices in T_R. Similar to the proof of Claim 3 (b) we estimate

$$\begin{aligned}
\mathbb{E}[Z] &= \sum_{k=1}^{n} \binom{t}{\lceil \frac{2k}{40} \rceil} \binom{n - \lceil \frac{2k}{40} \rceil}{k - \lceil \frac{2k}{40} \rceil} \binom{n}{k} (k-1)!k! \cdot \frac{((n-k)!)^2}{(n!)^2} \\
&= \sum_{k=1}^{n} \frac{t(t-1) \cdot \ldots \cdot (t - \lceil k/20 \rceil + 1)}{n(n-1) \cdot \ldots \cdot (n - \lceil k/20 \rceil + 1)} \cdot \frac{(k-1) \cdot \ldots \cdot (k - \lceil k/20 \rceil + 1)}{\lceil k/20 \rceil (\lceil k/20 \rceil - 1) \cdot \ldots \cdot 1} \\
&\leq \sum_{k=1}^{n} \left(\frac{t}{n - \lceil \frac{n}{20} \rceil + 1} \right)^{\lceil k/20 \rceil} \cdot \frac{k^{\lceil k/20 \rceil}}{\lceil k/20 \rceil !} \\
&\leq \sum_{k=1}^{n} \left(\frac{20t}{19n} \right)^{\lceil k/20 \rceil} \cdot \frac{k^{\lceil k/20 \rceil}}{\lceil k/20 \rceil !},
\end{aligned}$$

and by Stirling's formula we have

$$\begin{aligned}
\mathbb{E}[Z] &\leq \sum_{k=1}^{n} \left(\frac{20t}{19n} \right)^{\lceil k/20 \rceil} \cdot \frac{k^{\lceil k/20 \rceil}}{\sqrt{2\pi \lceil k/20 \rceil} \, (\lceil k/20 \rceil / e)^{\lceil k/20 \rceil}} \\
&\leq \sum_{k=1}^{n} \left(\frac{20t}{19n} \right)^{\lceil k/20 \rceil} \cdot (20e)^{\lceil k/20 \rceil} \\
&\leq \sum_{k=1}^{n} \left(86e^{-\delta/8} \right)^{\lceil k/20 \rceil} = o(1).
\end{aligned}$$

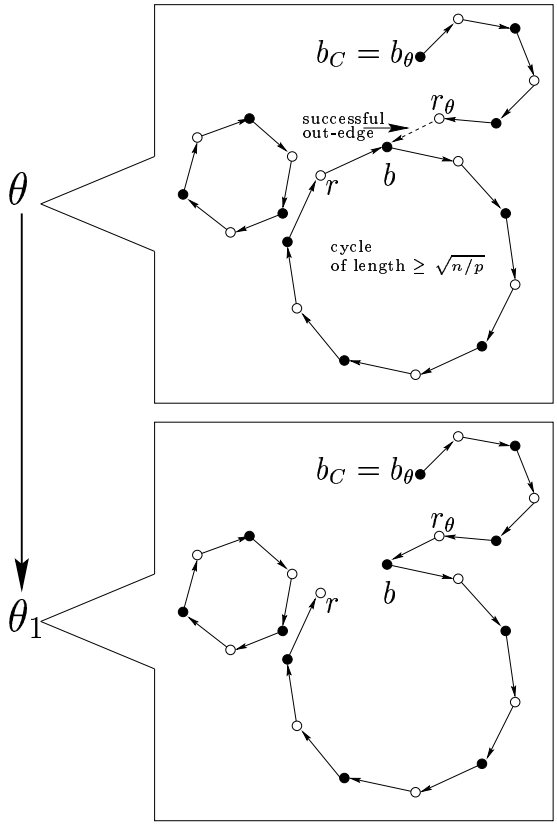
By the same argument $\mathbb{E}[\#$ cycles with $> \frac{1}{40}$ of their vertices in $T_B] = o(1)$. $\square$

Let us start with eliminating small cycles of σ. A cycle is called "small" if it contains less than $\sqrt{n/p}$ vertices. Let C be a small cycle of length l in σ. We describe an algorithm for removing C via the out- and in-edges. Once the in- or out-edges incident with a vertex have been observed, they may no longer be considered random. Therefore we mark such a vertex as "dirty". At the beginning of our cycle elimination algorithm all vertices apart from those in T are clean. Let r_C, b_C be two clean vertices from C forming an edge. W.l.o.g. we can assume that this edge is directed from r_C to b_C. By a "near-cycle-cover" we

mean a digraph θ consisting of a directed path P_θ between two clean vertices and a cycle cover extending over those vertices that are not on P_θ. We grow a rooted tree with near-cycle-covers as nodes. As the root we take the near-cycle-cover obtained by deleting (r_C, b_C) from $(R \cup B, M \cup \hat{M})$. The tree is grown as follows: let θ be any node of the tree with path $b_\theta \to r_\theta$. We consider out-edges of the form (r_θ, b) such that b and its predecessor r in θ are clean. (r_θ, b) is "successful"

- if b lies on a cycle of length $\geq \sqrt{n/p}$ or
- if b is on P_θ and the paths $b_\theta \to r$ and $b \to r_\theta$ both have length $\geq \sqrt{n/p}$.

For each successful edge we create a successor of θ by removing (r, b) and inserting (r_θ, b). b is marked dirty immediately afterwards. When all successful edges have been considered r_θ is marked dirty, too. We stop growing the tree when we have produced $\sqrt{n \ln n}$ leaves. This construction implies that throughout the algorithm the starting vertex of our paths stays the same, namely b_C.



Claim 4. Provided that we marked at most $o(n)$ vertices as dirty, we succeed to produce a tree with $\sqrt{n \ln n}$ leaves, thereby marking at most $O(\sqrt{n \ln n})$ additional vertices as dirty (a.s.).

Proof. We have proved as Claim 3 (c) that the number of vertices on small cycles is at most $\frac{n}{\sqrt[3]{np}}(= o(n))$. Since for each node θ in our tree the path in the associated cycle cover ends in some clean vertex, we can view the number of successors of θ as a random variable with binomial distribution $B(n - o(n), h)$. The marking steps in our algorithm ensure that paths belonging to distinct nodes θ_1, θ_2 end in distinct vertices. Thus the corresponding random variables are independent. Let Z_t denote the number of nodes in the t^{th} level of the tree, and let $\hat{\delta}$ abbreviate $(n - o(n))h$. We prove by induction on t that

$$\mathbb{P}\left[Z_t \notin [\hat{\delta}^{t-1}/2, 2\hat{\delta}^{t-1}]\right] < 2\sum_{j=1}^{t-1} e^{-\frac{\hat{\delta}^j}{8}} \text{ for } t \geq 2.$$

We start the induction by observing that $\mathbb{E}[Z_2] = \hat{\delta}$. The Angluin-Valiant inequalities yield $\mathbb{P}[Z_2 < \hat{\delta}/2] \leq e^{-\hat{\delta}/8}$ and $\mathbb{P}[Z_2 > 2\hat{\delta}] \leq e^{-\hat{\delta}/3}$, hence

$$\mathbb{P}\left[Z_2 \notin [\hat{\delta}/2, 2\hat{\delta}]\right] \leq e^{-\hat{\delta}/8} + e^{-\hat{\delta}/3} < 2e^{-\hat{\delta}/8}.$$

For the induction step we write

$$\begin{aligned}
\mathbb{P}\left[Z_t \notin [\hat{\delta}^{t-1}/2, 2\hat{\delta}^{t-1}]\right] &= \mathbb{P}\left[Z_t \notin [\hat{\delta}^{t-1}/2, 2\hat{\delta}^{t-1}] \mid Z_{t-1} = \hat{\delta}^{t-2}\right] \\
&\quad \cdot \mathbb{P}\left[Z_{t-1} = \hat{\delta}^{t-2}\right] \\
&\quad + \mathbb{P}\left[Z_t \notin [\hat{\delta}^{t-1}/2, 2\hat{\delta}^{t-1}] \mid Z_{t-1} \neq \hat{\delta}^{t-2}\right] \\
&\quad \cdot \mathbb{P}\left[Z_{t-1} \neq \hat{\delta}^{t-2}\right] \\
&\leq \mathbb{P}\left[Z_t \notin [\hat{\delta}^{t-1}/2, 2\hat{\delta}^{t-1}] \mid Z_{t-1} = \hat{\delta}^{t-2}\right] \cdot 1 \\
&\quad + 1 \cdot \mathbb{P}\left[Z_{t-1} \neq \hat{\delta}^{t-2}\right] \\
&\leq \mathbb{P}\left[Z_t \notin [\hat{\delta}^{t-1}/2, 2\hat{\delta}^{t-1}] \mid Z_{t-1} = \hat{\delta}^{t-2}\right] \\
&\quad + \mathbb{P}\left[Z_{t-1} \notin [\hat{\delta}^{t-2}/2, 2\hat{\delta}^{t-2}]\right].
\end{aligned}$$

Since $\mathbb{E}[Z_t \mid Z_{t-1} = \hat{\delta}^{t-2}] = \hat{\delta}^{t-1}$, the Angluin-Valiant inequalities give

$$\mathbb{P}\left[Z_t \notin [\hat{\delta}^{t-1}/2, 2\hat{\delta}^{t-1}] \mid Z_{t-1} = \hat{\delta}^{t-2}\right] < 2e^{-\hat{\delta}^{t-1}/8}.$$

So we get by induction assumption

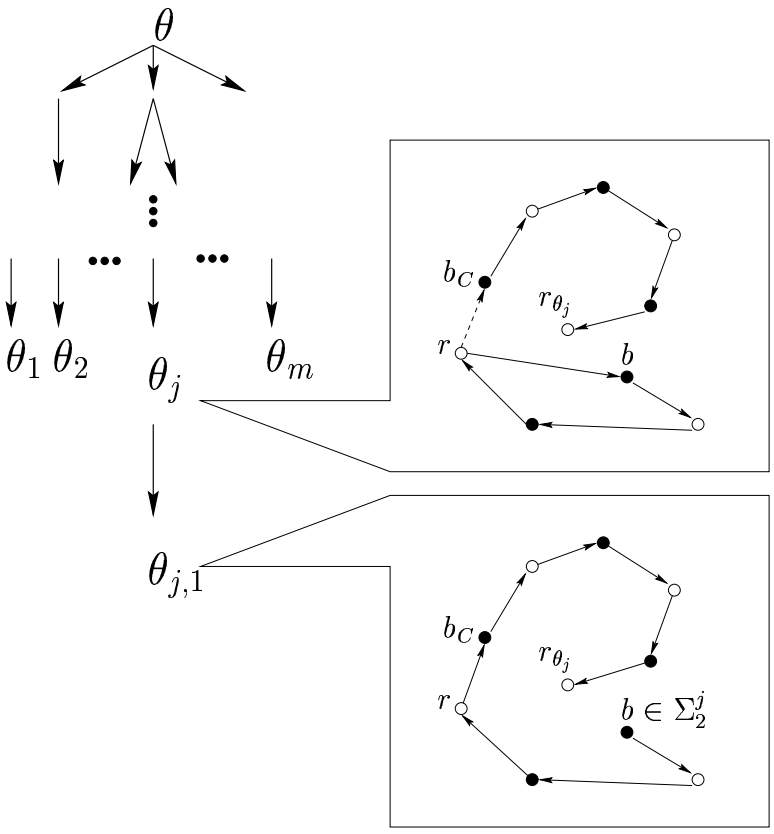
$$\mathbb{P}\left[Z_t \notin [\hat{\delta}^{t-1}/2, 2\hat{\delta}^{t-1}]\right] < 2e^{-\frac{\hat{\delta}^{t-1}}{8}} + \sum_{j=1}^{t-2} 2e^{-\frac{\hat{\delta}^j}{8}} = \sum_{j=1}^{t-1} 2e^{-\frac{\hat{\delta}^j}{8}}.$$

Let $k \in \mathbb{N}$ be such that $\frac{\hat{\delta}^k}{2} \geq \sqrt{n \ln n} \geq \frac{\hat{\delta}^{k-1}}{2}$. The probability that we fail to produce $\sqrt{n \ln n}$ leaves is bounded by

$$\begin{aligned} \mathbb{P}\left[Z_{k+1} \notin [\hat{\delta}^k/2, 2\hat{\delta}]\right] &< 2\sum_{j=1}^{k+1} e^{-\frac{\hat{\delta}^j}{8}} < 2\sum_{j=1}^{\infty} e^{-\frac{\hat{\delta}^j}{8}} \\ &= \frac{2e^{-\hat{\delta}/8}}{1 - e^{-\hat{\delta}^8}} = o(1). \end{aligned}$$

□

Let $\theta_1, \ldots, \theta_m$ denote the $\sqrt{n \ln n}$ leaves of the tree thus constructed. Each θ_j is a near-cycle-cover containing a path which starts at the clean vertex b_C and ends at some vertex r_{θ_j}. We take each leaf θ_j as the root of a new tree T_{θ_j}. These trees T_{θ_j} are grown simultaneously in the following fashion using in-edges: initially b_C is the single starting vertex of all paths corresponding to the roots θ_j. Let $\Sigma_1 := \{b_C\}$. If (r, b) is an edge in G such that r and b both are clean and (r, b_C) is an in-edge, we construct a successor for every θ_j by inserting (r, b_C) and deleting (r, b). r is immediately marked as dirty; b_C is marked dirty when all in-edges ending at b_C have been considered.



Note that this construction ensures that for each tree T_{θ_j} the sets Σ_2^j of starting vertices of the new paths in level 2 are identical. Now suppose we have grown the trees up to a level $t-1$ and Σ_{t-1} is the single set of starting vertices for the paths belonging to the current level of each tree. Determining the direct successors for each node in level $t-1$ in the same way as above clearly maintains the condition $\Sigma_t^j = \Sigma_t = \Sigma_t^{j'}$ for all j, j'. We stop growing the trees when we have reached a depth k with $\frac{\hat{\delta}^k}{2} \geq \sqrt{n \ln n} \geq \frac{\hat{\delta}^{k-1}}{2}$ as above. By the same calculations as in the proof of Claim 4 we obtain that every tree has $\sqrt{n \ln n}$ leaves almost surely. Moreover, since the edges we used for growing the trees are identical for each tree, a total number of at most $O(\sqrt{n \ln n})$ additional vertices is marked dirty. So the overall number of dirty vertices is $O(\sqrt{n \ln n}) = o(n)$. Note that the above construction did not care about the sizes of the cycles and paths. Therefore we now delete those nodes from each T_{θ_j} that contain a small path or a small cycle. We call T_{θ_j} "good" if it still has $\sqrt{n \ln n}$ leaves and "bad" otherwise. Since the number of vertices that can cause the creation of small cycles or paths is $o(n)$ we can repeat the proof of Claim 4 to see that $\mathbb{P}\left[T_{\theta_j} \text{ is bad}\right] \leq \frac{2e^{-\hat{\delta}/8}}{1-e^{-\hat{\delta}/8}}$. Thus the expected number of bad trees is at most $\sqrt{n \ln n} \cdot \frac{2e^{-\hat{\delta}/8}}{1-e^{-\hat{\delta}/8}}$ and by Markov's inequality

$$\mathbb{P}\left[\text{there exist more than } \frac{\sqrt{n \ln n}}{2} \text{ bad trees}\right] \leq \frac{4e^{-\hat{\delta}/8}}{1-e^{-\hat{\delta}/8}} = o(1).$$

So with probability $1 - o(1)$ more than $\sqrt{n \ln n}/2$ trees are good. Each good tree T_{θ_j} has $\sqrt{n \ln n}$ leaves each of which consists of cycles plus a long path from distinct starting vertices b to r_{θ_j}. Since for distinct θ_j, θ_j' the ending vertices $r_{\theta_j}, r_{\theta_j'}$ are distinct, too we have created an overall number of $(n \ln n)/2$ distinct long paths. Note that neither the out-edges of the ending vertices nor the in-edges of the starting vertices have been observed so far. Hence the probability that we cannot close any path via an in- or out-edge is smaller than

$$(1-h)^{2\frac{n \ln n}{2}} = \left(1 + \frac{-hn}{n}\right)^{n \ln n} \leq e^{-hn \ln n}.$$

But this means that with probability at least $1 - e^{-hn \ln n}$ we can eliminate the small cycle C. Since the number of cycles is less than $2 \ln n$, the probability that we fail to remove all small cycles is at most $2 \ln n \cdot e^{-hn \ln n} = o(1)$. Having got rid of the small cycles we are left with cycles $C_1, C_2, \ldots, C_l$ each having at least $\sqrt{n/p}$ vertices, so that $l \leq 2\sqrt{np}$. We want to use the patch-edges to combine these cycles into a single tour. Cycles C, C' can be patched together if we find edges $(v, w) \in C$, $(v', w') \in C'$ such that (v, w') and (v', w) are patch-edges. All edges, except those incident with T have probability h to be patch edges. It cannot happen that a cycle C is fully contained in T since $|C| > \sqrt{n/p}$,

$|T| \leq 3ne^{-\delta/8} = 3ne^{-\frac{n}{8}(1-\sqrt[5]{1-p})}$, and $p < \frac{1}{9n}e^{\frac{n}{4}(1-\sqrt[5]{1-p})}$ which is easily checked by taking derivatives with respect to p. So the probability that C, C' cannot be patched together is at most

$$\begin{aligned}
&(1-h^2)^{\left(\frac{|C|}{2}-|T_R|\right)\cdot\left(\frac{|C'|}{2}-|T_B|\right)+\left(\frac{|C|}{2}-|T_B|\right)\cdot\left(\frac{|C'|}{2}-|T_R|\right)} \\
&= (1-h^2)^{\frac{|C||C'|}{2}-|T|\left(\frac{|C|}{2}+\frac{|C'|}{2}\right)+2|T_R||T_B|} \\
&\leq (1-h^2)^{\frac{1}{2}\left(\frac{n}{p}-3n^2e^{-\delta/8}\right)} \\
&= \left(1-\frac{\delta^2}{n^2}\right)^{n^2\left(\frac{1}{2np}-3e^{-\delta/8}\right)} \\
&\leq e^{-\frac{\delta^2}{2np}+3\delta^2e^{-\delta/8}} = e^{-\frac{nh^2}{2p}+3e^{2\ln\delta-\delta/8}} \\
&= e^{-\frac{np}{2}\cdot\left(\frac{h}{p}\right)^2+o(1)}.
\end{aligned}$$

Since $\frac{h}{p} = \frac{h}{1-(1-h)^5} > \frac{1}{5}$ for all h we can upper bound the above expression by $e^{-\frac{np}{50}+o(1)}$. Now the probability that all cycles can be patched together is at least $1 - le^{-\frac{np}{50}+o(1)} \geq 1 - 2\sqrt{np}e^{-\frac{np}{50}+o(1)} = 1 - o(1)$. We used $o(n)$ additional light edges to eliminate all cycles. Hence the weight of the tour is

$$\mathrm{AP}(W)(1+\alpha) + o(n)\cdot\frac{\alpha\mathrm{AP}(W)}{2n} \to \mathrm{AP}(W)(1+\alpha) \text{ for } n\to\infty,$$

concluding the proof of our theorem. □

Corollary 2 *Let (Z_n) be a sequence of random variables over the nonnegative reals and let $p_n := \mathbb{P}[Z_n = 0]$. Let $W = (w_{ij})$ be a $2n \times 2n$ matrix such that $w_{ij} = \infty$ for $(i,j) \notin I$ and w_{ij} is drawn independently from the same distribution as Z_n for $(i,j) \in I$. If $n \cdot p_n \to \infty$ as $n \to \infty$ then BATSP(W) =AP(W) (a.s.).*

5.3 A negative result

Theorem 12 *Let $W = (w_{ij})$ be a $2n \times 2n$-matrix where, for $(i,j) \in I$, w_{ij} is independently and uniformly drawn from $\{0, 1, \ldots, \lfloor cn \rfloor\}$ and $w_{ij} = \infty$ for $(i,j) \notin I$.*

(a) $\mathbb{P}[AP(W) \neq BATSP(W)] \not\longrightarrow 0$ as $n \to \infty$, if $c \in \mathbb{R}_{>0}$.

(b) $\mathbb{P}[AP(W) \neq BATSP(W)] \longrightarrow 1$ as $n \to \infty$, if $c \xrightarrow{n\to\infty} \infty$.

Proof. Let $\pi_1 : \{1, \ldots, n\} \to \{1, \ldots, n\}$ and $\pi_2 : \{n+1, \ldots, 2n\} \to \{n+1, \ldots, 2n\}$ be random permutations and define a $2n \times 2n$-matrix $U = (u_{ij})$ by $u_{ij} := w_{i\pi(j)}$, where $\pi := \pi_1 \cup \pi_2$. Clearly, for $(i,j) \notin I$ we have $u_{ij} = \infty$ and for each $(i,j) \in I$ u_{ij} is independently and uniformly drawn from $\{0, 1, \ldots, \lfloor cn \rfloor\}$.

Let G, H be complete bipartite digraphs on $\{1,\ldots,n\} \cup \{n+1,\ldots,2n\}$, where the edge-weights of G are specified by W, while the edge-weights of H are determined by U. Again, let G', H' denote the associated bipartite graphs on $\{x_1,\ldots,x_{2n}, y_1,\ldots,y_{2n}\}$. With the definition

$$\begin{aligned}\mathcal{B} \;:=\; \{\varphi \mid \varphi = \varphi_1 \cup \varphi_2,\ \varphi_1 : \{1,\ldots,n\} \to \{n+1,\ldots,2n\},\\ \varphi_2 : \{n+1,\ldots,2n\} \to \{1,\ldots,n\},\ \varphi_1, \varphi_2 \text{ bijective}\}\end{aligned}$$

we see that if M is an optimal matching for G' then the edges of the subgraph of H induced by the corresponding optimal matching for H' can be considered as a random bijection from $\mathcal{B}$. We call an edge of G "forced" if its corresponding edge in G' is contained in each optimal matching for G'. A "positive" edge of G is an edge of weight ≥ 1 whose corresponding edge in G' is contained in at least one optimal matching for G'. Since U is from a probabilistic point of view "indistinguishable" from W the following holds:

if G has l_1 forced edges in $\{1,\ldots,n\} \times \{n+1,\ldots,2n\}$ and l_2 forced edges in $\{n+1,\ldots,2n\} \times \{1,\ldots,n\}$ then the probability that these contain a "forced" cycle equals the probability that l_1 and l_2 *random* edges from I contain a cycle.

Lemma 12 *Provided that G contains l_1 forced edges $\in \{1,\ldots,n\}\times\{n+1,\ldots,2n\}$ and l_2 forced edges $\in \{n+1\ \ldots,2n\} \times \{1,\ldots,n\}$ there is a forced cycle of length $\leq 2n-2$ in G with probability at least*

$$\max_{t\in\{2,4,\ldots,2n-2\}} \left\{ \frac{t}{t+4} \cdot \frac{l_1(l_1-1)\cdot\ldots\cdot(l_1-\frac{t}{2}+1)\cdot l_2(l_2-1)\cdot\ldots\cdot(l_2-\frac{t}{2}+1)}{n(n-1)\cdot\ldots\cdot(n-\frac{t}{2}+1)\cdot n(n-1)\cdot\ldots\cdot(n-\frac{t}{2}+1)} \right\}$$

for all $t \in \{2,4,\ldots,2n-2\}$.

Proof. Let $Z_{n,t}$ count the number of bijections in $\mathcal{B}$ containing a cycle of length $\leq t$, $t \in \{2,4,\ldots,2n-2\}$. Then the probability that a random $\varphi \in \mathcal{B}$ contains such a cycle is $\frac{Z_{n,t}}{(n!)^2}$. We show by induction on n that

$$\frac{Z_{n,t}}{(n!)^2} \geq \frac{t}{t+4}. \qquad (*)$$

The induction starts with $n = 2$ and $t = 2$. Obviously, $Z_{2,2} = 2$ and thus $\frac{Z_{2,2}}{(2!)^2} = \frac{2}{4} \geq \frac{2}{6} = \frac{t}{t+4}$, as claimed. For the induction step we observe that there are $\binom{n-1}{i-1}\binom{n}{i} i!(i-1)!$ bipartite cycles of length $2i$ in G containing the vertex 1. Hence the number of bijections $\varphi \in \mathcal{B}$ containing a cycle of length $\leq t$ through 1 is

$$\sum_{i=1}^{\frac{t}{2}} \binom{n-1}{i-1}\binom{n}{i} i!(i-1)!\left((n-i)!\right)^2,$$

and the number of bijections containing a cycle of length $\leq t$ *not* passing through 1 is

$$\sum_{i=\frac{t}{2}+1}^{n-2} \binom{n-1}{i-1} \binom{n}{i} i!(i-1)! \cdot Z_{n-i,t}.$$

Thus

$$\begin{aligned}
\frac{Z_{n,t}}{(n!)^2} &= \frac{1}{(n!)^2}\left(\sum_{i=1}^{\frac{t}{2}} \binom{n-1}{i-1}\binom{n}{i} i!(i-1)!\left((n-i)!\right)^2 \right. \\
&\qquad \left. + \sum_{i=\frac{t}{2}+1}^{n-2} \binom{n-1}{i-1}\binom{n}{i} i!(i-1)!Z_{n-i,t}\right) \\
&= \frac{1}{(n!)^2}\left(\sum_{i=1}^{\frac{t}{2}} \frac{(n-1)!n!i!(i-1)!\left((n-i)!\right)^2}{(n-i)!(i-1)!(n-i)!i!} \right. \\
&\qquad \left. + \sum_{i=\frac{t}{2}+1}^{n-2} \frac{(n-1)!n!i!(i-1)!Z_{n-i,t}}{(n-i)!(i-1)!i!(n-i)!}\right) \\
&= \frac{t}{2n} + \frac{1}{n}\sum_{i=\frac{t}{2}+1}^{n-2} \frac{Z_{n-i,t}}{\left((n-i)!\right)^2} \\
&\geq \frac{t}{2n} + \frac{n-2-\frac{t}{2}}{n} \cdot \frac{t}{t+4} \text{ by induction assumption} \\
&= \frac{t^2+4t+2nt-4t-t^2}{2n(t+4)} = \frac{t}{t+4}.
\end{aligned}$$

Let F_1 and F_2 denote the sets of forced edges of G in $\{1,\ldots,n\}\times\{n+1,\ldots,2n\}$ and $\{n+1,\ldots,2n\}\times\{1,\ldots,n\}$, respectively. Let φ be an optimal assignment for G (corresponding to an optimal matching in G'). We can view φ as being randomly chosen from $\mathcal{B}$. By FC, FC_t, C_t, and F_{l_1,l_2} we denote the following events:

$$\begin{aligned}
FC &: \text{“}\varphi \text{ contains a forced cycle of length} \leq 2n-2\text{”} \\
FC_t &: \text{“}\varphi \text{ contains a forced cycle of length} \leq t\text{”} \\
C_t &: \text{“}\varphi \text{ contains a cycle of length} \leq t\text{”} \\
F_{l_1,l_2} &: \text{“}|F_1| = l_1 \text{ and } |F_2| = l_2\text{”}
\end{aligned}$$

Let $t \in \{2, \ldots, 2n-2\}$ be arbitrary. We estimate

$$\begin{aligned}\mathbb{P}[FC|F_{l_1,l_2}] &\geq \mathbb{P}[FC_t|F_{l_1,l_2}] \\ &= \mathbb{P}[FC_t \cap C_t|F_{l_1,l_2}] \\ &= \mathbb{P}[C_t|F_{l_1,l_2}] \cdot \mathbb{P}[FC_t|C_t \cap F_{l_1,l_2}] \\ &= \mathbb{P}[C_t] \cdot \mathbb{P}[FC_t \mid C_t \cap F_{l_1,l_2}],\end{aligned}$$

since C_t and F_{l_1,l_2} are independent events. Using $(*)$ we deduce

$$\begin{aligned}\mathbb{P}[FC|F_{l_1,l_2}] &\geq \frac{t}{t+4}\mathbb{P}[FC_t|C_t \cap F_{l_1,l_2}] \\ &\geq \frac{t}{t+4}\frac{l_1(l_1-1)\cdot\ldots\cdot(l_1-\frac{t}{2}+1)l_2(l_2-1)\cdot\ldots\cdot(l_2-\frac{t}{2}+1)}{n(n-1)\cdot\ldots\cdot(n-\frac{t}{2}+1)n(n-1)\cdot\ldots\cdot(n-\frac{t}{2}+1)}.\end{aligned}$$

□

If G contains a forced cycle of length $\leq 2n-2$ then clearly $AP(W) \neq BATSP(W)$. Hence $\mathbb{P}[AP(W) \neq BATSP(W)]$ is at least

$$\begin{aligned}\mathbb{P}[FC] &= \sum_{l_1,l_2=1}^{n} \mathbb{P}[FC \cap F_{l_1,l_2}] \\ &= \sum_{l_1,l_2} \mathbb{P}[FC|F_{l_1,l_2}] \cdot \mathbb{P}[F_{l_1,l_2}] \\ &\geq \frac{t}{t+4} \cdot \frac{\left(\sum_{l_1=1}^{n}(l_1)_{\frac{t}{2}}\mathbb{P}[|F_1|=l_1]\right)\left(\sum_{l_2=1}^{n}(l_2)_{\frac{t}{2}}\mathbb{P}[|F_2|=l_2]\right)}{n^2(n-1)^2\cdot\ldots\cdot(n-\frac{t}{2}+1)^2},\end{aligned}$$

where $(x)_k$ denotes $x(x-1)\cdot\ldots\cdot(x-k+1)$. In $\mathbb{N}$ the equation

$$x(x-1)\cdot\ldots\cdot(x-k+1) = s(s-1)\cdot\ldots\cdot(s-k+1)$$

has the unique solution $x = s$ (since the left-hand-side is strictly increasing with x). Thus $\mathbb{P}[|F_i| = l_i] = \mathbb{P}\left[(|F_i|)_{\frac{t}{2}} = (l_i)_{\frac{t}{2}}\right]$, $i \in \{1,2\}$, giving

$$\mathbb{P}[FC] \geq \frac{t}{t+4}\frac{\mathbb{E}\left[(|F_1|)_{\frac{t}{2}}\right]\mathbb{E}\left[(|F_2|)_{\frac{t}{2}}\right]}{n^2(n-1)^2\cdot\ldots\cdot(n-\frac{t}{2}+1)^2}.$$

Lemma 13 *The expected number of forced edges it at least the expected number of positive edges.*

Proof of Lemma 13. Let $\mathcal{F}_e$ and $\mathcal{P}_e$ be the sets of weight functions such that e is a forced or a positive edge, respectively. For $w' \in \mathcal{P}_e$ let

$$w''(x) := \begin{cases} w'(x), & \text{for } x \neq e, \\ w'(x)-1, & \text{for } x = e. \end{cases}$$

Then $w'' \in \mathcal{F}_e$ and hence $|\mathcal{F}_e| \geq |\mathcal{P}_e|$. $\square$

Let P_1 and P_2 denote the sets of positive edges in $\{1,\ldots,n\} \times \{n+1,\ldots,2n\}$ and $\{n+1,\ldots,2n\} \times \{1,\ldots,n\}$. Lemma 13 gives

$$\mathbb{E}\left[(|F_i|)_{\frac{t}{2}}\right] \geq \mathbb{E}\left[(|P_i|)_{\frac{t}{2}}\right], \ i \in \{1,2\}.$$

For every $v \in \{1,\ldots,n\}$ let X_v be a random variable indicating if v is the starting vertex of some zero weighted edge:

$$X_v := \begin{cases} 0, & \exists\ v' \in \{n+1,\ldots,2n\} :\ w_{vv'} = 0, \\ 1 & \text{otherwise.} \end{cases}$$

Then

$$\begin{aligned} \mathbb{E}[X_v] &= \mathbb{P}[X_v = 1] = \left(\frac{\lfloor cn \rfloor}{\lfloor cn \rfloor + 1}\right)^n \\ &= \left(1 + \frac{1}{\lfloor cn \rfloor}\right)^{-n} = \left(1 + \frac{1}{\lfloor cn \rfloor}\right)^{\lfloor cn \rfloor \cdot \left(-\frac{n}{\lfloor cn \rfloor}\right)} \\ &\geq e^{-\frac{n}{\lfloor cn \rfloor}} \\ &= (1+o(1))e^{-1/c} \end{aligned}$$

Clearly, $|P_1| \geq \sum_{v \in \{1,\ldots,n\}} X_v$ and hence

$$\begin{aligned} \mathbb{E}\left[(|P_1|)_{\frac{t}{2}}\right] &\geq \mathbb{E}\left[(\sum_v X_v)_{\frac{t}{2}}\right] \\ &= \mathbb{E}\left[(\sum X_v)(\sum X_v - 1) \cdot \ldots \cdot (\sum X_v - \frac{t}{2} + 1)\right] \\ &= \left(\frac{t}{2}\right)! \binom{n}{\frac{t}{2}} \mathbb{E}[X_1]^{\frac{t}{2}} \\ &= \left(\frac{t}{2}\right)! \binom{n}{\frac{t}{2}} (1+o(1))e^{-\frac{t}{2c}}, \end{aligned}$$

as can be easily verified by induction. Repeating the same calculation for P_2 we conclude that

$$\begin{aligned} \mathbb{P}[AP(W) \neq ATSP(W)] &\geq \mathbb{P}[FC] \\ &\geq \frac{t}{t+4} \cdot \frac{\mathbb{E}\left[(|F_1|)_{\frac{t}{2}}\right] \cdot \mathbb{E}\left[(|F_2|)_{\frac{t}{2}}\right]}{n^2(n-1)^2 \cdot \ldots \cdot (n-\frac{t}{2}+1)^2} \\ &\geq \frac{t}{t+4} \cdot \frac{\mathbb{E}\left[(|P_1|)_{\frac{t}{2}}\right] \cdot \mathbb{E}\left[(|P_2|)_{\frac{t}{2}}\right]}{n^2(n-1)^2 \cdot \ldots \cdot (n-\frac{t}{2}+1)^2} \end{aligned}$$

$$\begin{aligned} &\geq \frac{t}{t+4} \cdot \frac{\left(\left(\frac{t}{2}\right)!\right)^2 \binom{n}{\frac{t}{2}}^2 (1+o(1))e^{-\frac{t^2}{c}}}{n^2(n-1)^2 \cdot \ldots \cdot (n-\frac{t}{2}+1)^2} \\ &= \frac{t}{t+4}(1+o(1))e^{-\frac{t^2}{c}}, \end{aligned}$$

giving the claim of (a) for $t = 2$ and of (b) for $t = \sqrt[4]{c}$. □

Chapter 6

Open questions

The considerations in the previous chapters suggest a number of problems for further research.

• In Chapter 2 we saw that the bipartite Euclidean TSP is approximable within $2 \cdot \mathrm{OPT}$ in time $O(n^7)$. The dominating term in the complexity bound is due to the running time of the weighted matroid intersection algorithm needed to find a B2-tree. Is it possible to compute a B2-tree in less time? We observed that the construction does not actually need a B2-tree but that it would be sufficient to determine a minimum weight spanning tree which contains a perfect matching between red and blue vertices. How can this be done efficiently?

• The ordinary Euclidean TSP can be approximated within $\frac{3}{2} \cdot \mathrm{OPT}$ and in fact, it even allows for a polynomial time approximation scheme. We pointed out the difficulties of transferring the methods used for these results to the bipartite case. Still, it is a challenging open question whether or not it is possible to give an improved approximation algorithm or even a PTAS for the bipartite TSP.

• We gave a worst-case example proving that version 1 of our cycle cover algorithm can produce tours of length $3 \cdot \mathrm{OPT}$. Is there a similar example for the second version?

• Turning to Chapter 4, it would be of interest to improve the parallel perfect matching algorithm. Can we do with a smaller number of processors without loosing approximation quality? How to close the $O(\log n)$-gap between the approximate and the optimal tours?

• Almost nothing is known about bipartite TSP tours through random Poisson distributed points. We gave a non-trivial upper bound on the optimal tour length. How to improve it? How to find a non-trivial lower bound?

• Finally, concerning the results of Chapter 5, it seems to be worth studying what happens if we add symmetry to our weight matrix W. Clearly, a minimum weight assignment in this case would merely induce 2-cycles spoiling the described construction by increasing the number of cycles to n instead of $O(\log n)$. However, is it possible to save the situation by replacing the assignment with an optimal cycle cover where each cycle is of size ≥ 4?

Bibliography

[1] M. Ajtai, J. Komlós, G. Tusnády, *On optimal matchings.* Combinatorica **4** (1984), 259–264.

[2] D. Angluin, L.G. Valiant, *Fast probabilistic algorithms for Hamiltonian circuits and matchings.* J. Comp. Sys. Sci. **18** (1979), 155–193.

[3] S. Anily, R. Hassin, *The swapping problem,* Networks **22** (1992), 11–18.

[4] S. Arora, *Polynomial time approximation schemes for Euclidean TSP and other geometric problems.* Manuscript, March 30, 1996. Appears in Proc. 37th Annu. IEEE Sympos. Found. Comput. Sci. (1996), 2–12.

[5] S. Arora, *Nearly linear time approximation schemes for Euclidean TSP and other geometric problems.* In Proc. 38th Annu. IEEE Sympos. Found. Comput. Sci. (FOCS 97) (1997).

[6] A.D. Barbour, L. Holst, S. Janson. Poisson Approximation. Oxford Science Publications. Claredon Press (1992).

[7] J. Beardwood, J.H. Halton, J.M. Hammersley, *The shortest path through many points.* Proc. Cambridge Philos. Soc. **55** (1959), 299–327.

[8] P. Chalasani, R. Motwani, A. Rao, *Approximation Algorithms for Robot Grasp and Delivery.* In Proceedings of the 2nd International Workshop on Algorithmic Foundations of Robotics, Toulouse, France (1996).

[9] N. Christofides, *Worst-case analysis for a new heuristic for the Traveling Salesman problem.* In Symposium on New Directions and Recent Results in Algorithm and Complexity (Ed.: J.F. Traub). Academic Press (1976).

[10] W.J. Cook, W.H. Cunningham, W.R. Pulleyblank, A. Schrijver, *Combinatorial Optimization.* Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons, inc. (1998).

[11] J. Edmonds, *Matroid intersection.* In Discrete Optimization I, Annals of Discrete Mathematics **4** (Eds.: P.L. Hammer, E.L. Johnson, B.H. Korte), North-Holland, Amsterdam (1979), 39–49.

[12] A.Frank, *A weighted matroid intersection algorithm.* Journal of Algorithms **2** (1981), 328–336.

[13] A. Frank, B. Korte, E. Triesch, J. Vygen, *On the Bipartite Travelling Salesman Problem.* Report No. 98866-OR, Research Institute for Discrete Mathematics, University of Bonn (1998).

[14] A. Frieze, M. Karp, B. Reed, *When is the Assignment Bound Tight for the Asymmetric Traveling-Salesman Problem?* SIAM Journal on Computing **24** (1995), 484–493.

[15] M.R. Garey, D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, San Francisco (1979).

[16] G.A.P. Kindervater, J.K. Lenstra, D. Shmoys, *The Parallel Complexity of TSP Heuristics.* Journal of Algorithms **10** (1989), 249–270.

[17] B. Korte, J. Vygen, *Combinatorial Optimization.* Springer (2000).

[18] J.B. Kruskal, *On the shortest spanning tree of a graph and the traveling salesman problem.* Proceedings of the American Mathematical Society **7** (1956), 58–50.

[19] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys (editors), *The Traveling Salesman Problem – A Guided Tour of Combinatorial Optimization.* John Wiley & Sons (1985).

[20] C. Michel, *Optimierung des Fertigungsprozesses von Leiterplatinen durch approximative Algorithmen.* Diplomarbeit, Bonn (1993).

[21] C. Michel, H. Schroeter, A. Srivastav, *Alternating TSP and Printed Circuit Board Assembly.* submitted.

[22] D.L. Miller, J.F. Pekny, *Exact solution of large asymmetric traveling salesman problems.* Science **251** (1991), 754–762.

[23] G.L. Nemhauser, L.A. Wolsey, *Integer and Combinatorial Optimization.* Wiley, New York (1988).

[24] R.C. Prim, *Shortest connection networks and some generalizations.* Bell Systems Technical Journal **36** (1957), 1389–1401.

[25] G. Reinelt, *The Traveling Salesman – Computational Solutions for TSP Applications.* Lecture Notes in Computer Science **840**, Springer (1994).

[26] S.B. Rao, W.D. Smith, *Approximating geometric graphs via "spanners" and "banyans".* Proceedings of the 30th Annual ACM Symposium on the Theory of Computing (1998), 540–550.

[27] D.W. Walkup, *Matchings in random regular bipartite graphs.* Discrete Mathematics **31** (1980), 59-64.