アルゴリズム 74－7
（2000．9．21）

平面巡回セールスマン問題の高速な近似アルゴリズム

久保典弘[1] 村本勝洋[2] 下薗真一[3]

[1] 九州工業大学大学院情報科学専攻

[2] 岩下エンジニアリング（株） FA 事業部

[3] 九州工業大学情報工学部知能情報工学科

概要

平面上に与えられた点をすべてを通過し最初の点に戻ってくる最も短い巡回路を求める問題の，非常に高速なアルゴリズムについて述べる．本アルゴリズムは非常に単純なアルゴリズムなので実装が簡単である．また，n 個の点が与えられたときに $O(n\log n)$ 時間，$O(n)$ 領域で計算を行う．アルゴリズムは，平面を分割し，点をソートして分割領域内の順路を求め，各領域内の順路をつなぎ合わせる構成をとる．ランダムに点が分布するときは確率的に定数近似アルゴリズムである．このアルゴリズムを Karp の分割アルゴリズム，Lin-Kernighan の局所探索，Arora の近似スキームなどと比較した結果，実行時間は最も速く精度は同じかより良いという結果が得られた．

A Simple and Quick Approximation Algorithm for Traveling Salesman Problem in the Plane

Norihiro Kubo[1] Katsuhiro Muramoto[2] Shinichi Shimozono[3]

[1]Graduate School of Computer Science and Systems Enginieering, Kyushu Inst. of Tech.

[2]FAD, Iwashita Engineering Inc.

[3]Dept. of Artificial Intelligence, Kyushu Inst. of Tech.

Abstract

We present a quite simple, fast and practical algorithm to find a short cyclic tour that visits a set of points distributed on the plane. The algorithm runs in $O(n\log n)$ time with $O(n)$ space, and is simple enough to easily implement on resource restricted machines. It constructs a tour essentially by axis-sorts of the points and takes a kind of the 'fixed dissection strategy.' We show that the algorithm is a 'probabilistic' constant-ratio approximation algorithm for uniform random distributions. We made computational comparisons of our algorithm, Karp's partitioning , Lin–Kernighan local search, Arora's PTAS, etc. The results indicate that in running time our algorithm overwhelms existing ones, and the average approximation ratio is better or competitive.

1 はじめに

巡回セールスマン問題（Traveling Salesman Problem）とは点の集合とそれらの間の距離が与えられたときに，すべての点を一度ずつ訪れて最初の点に戻ってくる巡回路の中でもっとも距離の短いものを求める問題である．この問題は最適化問題の中ではよく知られている問題であり，理論と実際の両方で多くの研究が行われている [11]．TSP は NP- 困難であり [9]，P=NP でない限り多項式時間で最適解を求めることはできない．TSP の部分問題で 2 点間の距離が三角不等式を満たすものを Δ-TSP(または metric-TSP) とよぶ．この問題は APX- 完全であることが証明されており，多項式時間で定数倍以内の評価値の解を求めることができる [2, 5, 11]．本研究で取り扱うのは平面巡回セールス

マン問題である．この問題は Δ-TSP の部分集合であり，NP- 困難である [6, 13]．この問題の問題例は 2 次元のユークリッド平面上の点の集合と各点の間のユークリッド距離で与えられる．現実の問題の多くがこの問題に帰着できる．例えば，この問題のアルゴリズムはプリント基板の自動組み立てロボットの動作順序などに応用できる．

Karp は平面を再帰的に分割する方法を用いた平面 TSP の近似アルゴリズムを考案した [10]．このアルゴリズムは n 個の点に対して $O(n\log n + n\cdot c^t)$ 時間で計算を行う．ここで t は最後の分割でできた領域内の点の最大の数である．c^t(ただし$c > 2$) は分割領域内の最適な巡回路を求めるのに必要な時間である．このアルゴリズムによって導かれる解は最悪な場合でも最適解との差が $O(\sqrt{n})$ 以内の評価値の解を求めることが可能である．さらに，このアルゴリズムは問題例の点の分布がポアソン分布に従うときは多項式時間で“確率的”に $(1+\epsilon)$- 近似スキームである．ここで ϵ は最悪の場合の相対誤差で t によって決まる．また，ごく最近 Arora は d- 次元ユークリッド空間上の TSP の多項式時間近似スキーム (PTAS) を考案した [1]．このアルゴリズムは任意の $\epsilon(>0)$ に対して，近似率が $1+\epsilon$ 以内の解を多項式時間で求めることができる．このアルゴリズムの実装は実用性と実行時間の両面からみて難しいが，平面上の TSP が PTAS に属することを初めて証明した．

今まで TSP の近似アルゴリズムの多くは，できるだけ精度の良い解を求めることを重視してきた．それは CAD の最適化のような一度求めた解を何度も使う大量生産的な分野では，多くのコストをかけてでも精度の良い解を求めることが要求されるからである．しかし，多品種少量生産の分野では多種多様の要求を処理するために，計算時間と消費メモリに重点を置いたアルゴリズムが必要とされる．今まで多くの時間を必要としたプロセスを瞬時に処理し，かつある程度の満足できる精度の解を保証することができれば，作業ラインの効率化ができる．そこで，本研究では，平面上の巡回セールスマン問題の単純で高速な *divide-and-sort* アルゴリズムを考案した．

Divide-and-sort アルゴリズムは n 個の点の入力に対して $O(n\log n)$ 時間で計算を行い点の分布がポアソン分布に従うときは，確率 1 で最適化の定数倍以内の解を求めることができる．アルゴリズムのポイントは点を座標軸に対してソートしそれを用いて解を求めるところにある．アルゴリズムではすべての点を囲んだ領域を，$2k$ 個の横長の領域と 1 つの縦長の領域に分割する．k の値は n の平方根と点を囲む領域の面積に比例する．座標軸に沿ってソートしたリストを用いて各領域内の順路を求める．そして，それぞれの順路を Halton らのアルゴルリズム [7] の様につなぎ合わせて巡回路を導く．*Divide-and-sort* の計算時間は他のアルゴリズムと比べ群を抜いて速い．分割結合型のアルゴリズムには全体の領域を $O(n)$ 個に分割し，それらの領域の最適解を求めるものがあるが，これは良い近似率を保持するために時間計算量に表われない要素が多く，決して計算時間は少なくない．また，Halton らによるアルゴリズムは最悪の場合，計算に点の数の指数時間かかる．しかし，*divide-and-sort* は実用的なアルゴリズムで，良い近似率でかつ最悪でも $O(n\log n)$ 時間で計算を行う．

本論文では，まず始めにいくつかの定義を行い，続いてアルゴリズムの説明をする．次に最悪の場合の近似率と，ポアソン分布に従う場合の近似率を求めた．その結果，点の分布がポアソン分布に従うときは”確率的”に定数近似アルゴリズムになることを証明した．また実験を行い，以下のことを確かめた．まず始めに，分割の数 k の最適な値を求めた．次に，Karp の分割アルゴリズム，Lin-Kernighan の局所探索 [12]，Arora の近似スキームなどのよく知られている近似アルゴリズムと計算時間，精度を比較した．その結果，実行時間はずば抜けて速かった．精度については，時間計算量が $O(n^2)$ 以下のアルゴリズムより良い解を求めることを確認した．また TSPLIB[16] の解との比較でも同様の結果が得られた．

2 定義

平面 Z^2 上の点 c_1, c_2 のユークリッド距離を $d(c_1, c_2)$ とする.
平面巡回セールスマン問題は次のように定義できる.

定義　平面巡回セールスマン問題 (平面 TSP)

$$\begin{aligned} \text{問題例} &: \text{平面上の点の集合} C = \{c_1, c_2, ..., c_n\} \subseteq Z^2 \\ \text{解} &: C\text{の巡回路}\pi. \text{すなわち,整数} 1, ..., n\text{の順列}\pi = \langle \pi(1), ..., \pi(n) \rangle. \\ \text{評価値} &: \text{巡回路}\pi\text{の長さ } d(c_{\pi(n)}, c_{\pi(1)}) + \sum_{i=1}^{n-1} d(c_{\pi(i)}, c_{\pi(i+1)}). \end{aligned}$$

問題例に対する解の中で評価値ができるだけ小さい解を求める問題が巡回セールスマン問題である. ここでは点の間の距離は一般的なユークリッド距離で定義しているが, 以後の議論では距離の評価をマンハッタン距離にしても同様な結果が得られる.

3 Divide-and-sort アルゴリズムの手順

1. 準備

   (a) 点をすべて囲み, 各辺が座標軸に平行な最も小さい長方形を求める. だたしその各角は Z^2 上の点で与えられ, さらに以後の議論を簡単にするため幅 l_H と高さ l_V は奇数でかつ $l_H \geq l_V$ とする.

   (b) 長方形の枠内を整数 $k(\geq 1)$ に対して次のように分割する.
   (i) 長方形の枠内の左端に縦が l_V 横が $\frac{1}{2k} l_V$ の領域をとる.
   (ii) 長方形の枠内から (i) の領域を除いた部分を縦が $\frac{1}{2k} l_V$ 横が $l_H - \frac{1}{2k} l_V$ の領域に $2k$ 等分する.

2. ソートとつなぎ合わせ.

   (a) 縦が l_V 横が $\frac{1}{2k} l_V$ の領域は縦方向に $\frac{1}{2k} l_V$ 横が $l_H - \frac{1}{2k} l_V$ の領域は横方向に領域内の点のソートを行う. そのソートした点の順序を領域内の順路とする.

   (b) 上で求めた順路を次のようにつなぎ合わせる. (図 1)
   (i) 一番上と一番下の横長の長方形内の順路の左端の点を縦長の長方形内の順路の上端と下端の点にそれぞれつなぎ合わせる.
   (ii) 上から奇数番目の横長の長方形は隣接する下の長方形と順路の右端同士を, 偶数番目の長方形は左端同士をつなぎ合わせる.

4 Divide-and-sort のアルゴリズムの計算量

上記のアルゴリズム手順では理解しやすいように分割を行ってからソート, 結合を行っているが, 時間計算量が分割数 k によらないようにするには以下の手順をとる. ます, x 成分に関してソートしたリストと y 成分に関してソートした 2 つのリストを作る. このとき同一成分がある場合は, もう一方の成分の大小で大きさを決める. そして, そのソートした点を小さい順にみていき各領域の順路と全体の巡回路を作る. そうすると, divide-and-sort のアルゴリズムの時間計算量は長方枠を求めるのに

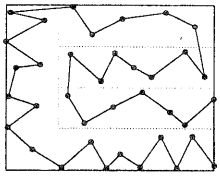

図 1: アルゴリズムで求めた解の例 ($k = 2$のとき)

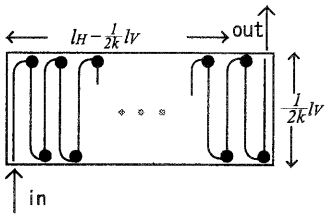

図 2: 最悪な場合の領域内の順路

$O(n)$ 時間, 点のソートには $O(n \log n)$ 時間かかるので $O(n \log n)$ 時間になる. 領域計算量はソートした点を保存するために必要な $O(n)$ 領域となる.

再帰的に分割する Karp のアルゴリズム [10] は $O(n \log n)$ 領域を必要とするが, divide-and-sort は $O(n)$ しか領域を必要としない. また, よく似た分割とつなぎ合わせを行う Halton と Terada のアルゴリズム [7] は分割した領域内の順路を求めるための時間計算量が非常に大きく, 最悪の場合, 点の数の指数時間かかる.

5 Divide-and-Sort の精度

5.1 最悪な場合の精度

Divide-and-sort で求められる解は最悪な場合でもどれだけの精度を保証するかを見積もる. まず, アルゴリズムで求められる解の長さを見積もる. 与えられた点を囲んだ長方形の横の長さを l_H, 縦の長さを l_V とする. また, 分割によってできた横長の長方形を R_i $(1 \leq i \leq 2k)$, 縦長の長方形を R_{2k+1} とする長方形 R_i $(1 \leq i \leq 2k+1)$ の中の点の個数を N_i とする.
このとき, 長方形 R_i $(1 \leq i \leq 2k)$ の中の巡路の長さは高々 $l_H - \frac{1}{2k} l_V + (N_i + 1)\frac{1}{2k} l_V$ となる (図 2). 同様に長方形 R_{2k+1} の中の巡路の長さは高々 $l_V + (N_{2k+1} + 1)\frac{1}{2k} l_V$ となる. 一般性を失わずに $l_H \geq l_V$ とすると巡回路の長さは,

$$\begin{aligned} & l_V + (N_{2k+1} + 1)\frac{1}{2k} l_V + \sum_{i=1}^{2k} \left(l_H - \frac{1}{2k} l_V + (N_i + 1)\frac{1}{2k} l_V \right) \\ \leq\ & \frac{1}{2k} l_V \cdot n + (2k+1) l_H \end{aligned}$$

従ってこのアルゴリズムによって求められる巡回路の長さは $\frac{1}{2k} l_V \cdot n + (2k+1) l_H$ 以下である. 次に分割数 k について考える. 巡回路の長さが最も短くなる分割数は $k = \left\lceil \frac{1}{2} \sqrt{\frac{l_V}{l_H}} \cdot \sqrt{n} \right\rceil$ である. このとき最悪な場合の巡回路の長さの上限を $W(C)$ とすると

$$W(C) \leq l_H \left(\frac{3}{2} \sqrt{\frac{l_V}{l_H}} + 4 \right)$$

となる. 問題例 C に対する最適解を $opt(C)$ とする. 最適解の長さは $2l_H$ 以上であり, また相加相乗平均より $\sqrt{l_H \cdot l_V} \leq l_H$ なので

$$W(C)/opt(C) \leq \frac{3}{4}\sqrt{n} + 2$$

が導かれる.
以上のことから, divide-and-sort で求めた解の評価値はは最適解の評価値の $\frac{3}{4}\sqrt{n}+2$ 倍以内である.

5.2 点の分布がポアソン分布に従う場合の精度

一般性を失わずに, 点は実数の組, 長方枠の面積は 1 とする. 点の数が十分大きいとき, アルゴリズムで求められる巡回路の長さ W は小さい定数 $c(>0)$ が存在して

$$W(C) \leq \left(\frac{3}{4} + c\right)\sqrt{l_V l_H \cdot n}$$

となる.

長方形 X (面積 1) 内の点の分布が以下の仮定を満たすとき, ポアソン分布 $\Pi_n(X)$ に従うという. (i) X を分割してできた各領域内の点の数はそれぞれの領域で独立している. (ii) 分割領域 Y 内の点の数の期待値は $n\mathcal{A}(Y)$ である. ただし, $\mathcal{A}(Y)$ は Y の面積. (iii) $\mathcal{A}(Y)$ が 0 になるより速く Y 内に 2 つ以上の点が存在する確率は 0 に近づく.
J.Beardwood らによって点の分布が $\Pi_n(X)$ に従うとき長方形 X 内の最適解の長さを $T_n^*(X)$ とすると, 任意の $\epsilon(>0)$ に対して

$$\left|\lim_{n\to\infty} \frac{T_n^*(X)}{\sqrt{n}} - \beta\right| \leq \epsilon$$

となる定数 β が確率 1 で存在することが証明されている [4]. 以上のことから点の数 n が十分に大きいとき, アルゴリズムで求めた解の評価値は確率 1 で最適解の評価値の定数倍以内になる.

よって, 点の数が十分大きく, かつ点の分布がポアソン分布に従うとき, divide-and-sort は確率的に定数近似アルゴリズムである.

6 実験結果

Divide-and-sort と従来のアルゴリズムとの比較を行った. Divide-and-sort の実装は C++ で行い, 点のソートには STL(Standerd Template Library) の共通アルゴリズムの sort を用いている. STL の sort 関数はクイックソートでソーティングを行うため, 長さ n の入力に対し平均 $O(n\log n)$ 時間で解法を行うが最悪の場合は $O(n^2)$ 時間かかる. 比較を行ったアルゴリズムは以下の通りである.

Karp のアルゴリム [10] は分割結合型のアルゴリズムで時間計算量は $O(n\log n)$ である. 図中では Karp3 と Karp8 で表される. 各数値は分割した領域内の点の数がその値以下になれば分割を終了する値である. また, 実験では分割領域内の点の数が 8 以下と小さいので, 分割領域内の最短の巡回路を求めるために $O(n^2 2^n)$ 時間で計算を行う DP アルゴリズム [3, 8] を使わずに, $(n-1)!/2$ 通りの数え上げを行った. Nearest Neighbor(N Neighbor)[15] は欲張り法を用いたアルゴリズムで, ランダムに与えられた巡回路から $O(n^2)$ 時間でよりよい巡回路を求める. 解の長さは最適解の $\log n$ に比例する. Double Min Spanning Tree(D Span) は最適解の 2 倍以内の長さの巡回路を求めることができる. Lin–Kernighan のアルゴリズム (L–K)[14] は局所探索を用いたアルゴリズムで 2-Opt と 3-Opt の両方を行っている. Lin–Kernighan のアルゴリズムは現在最も良いアルゴリズムといわれている. Arora のアルゴリズム [1] はランダム PTAS で portal が 3, cross は 1 で計算を行った. これら値は, 計算を現実的な時間で実行できる最も大きい値である.

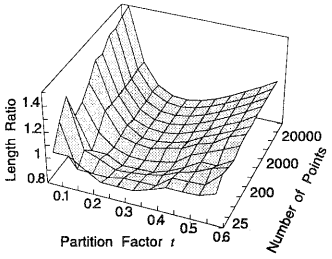

図 3: 最適な分割数

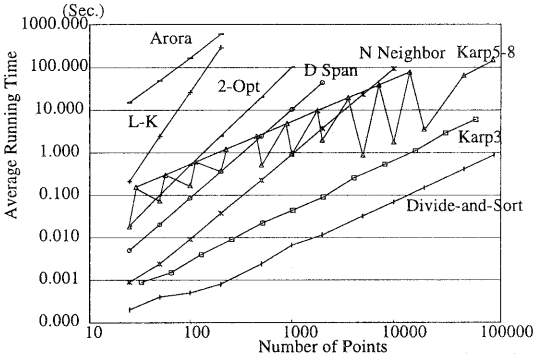

図 4: 各アルゴルズムの実行時間の比較

6.1 分割数 k について

分割数 $k = \left\lceil \frac{1}{2}\sqrt{\frac{l_V}{l_H}} \cdot \sqrt{n} \right\rceil$ は点の散らばり方が最悪の場合に最適な値である．そこで，点の分布に偏りのない問題例では分割数をいくらにすればよいか確かめた．実験は $k = \left\lceil t\sqrt{\frac{l_V}{l_H}} \cdot \sqrt{n} \right\rceil$ としたときの t の値と点の数をそれぞれ変化させ計算を行った．計算は点の数ごとに C++ のランダム関数によって生成した 100 通りの問題例に対して行った．そして，計算で求めた巡回路の長さと $t = 0.5 = \frac{1}{2}$ のときの巡回路の長さとの比の平均を求めた（図 3）．その結果，点の数に関わらず $k = 0.29 \sim \frac{1}{3.4}$ をとととれば divide-and-sort で最もよい評価値の解を求めることができることがわかった．また，点を囲む領域の縦と横の長さの比を 1： 1 から 1： 16 まで変化させたが，同様の結果が得られた．

6.2 実行時間の比較

Divide-and-sort の実行時間を他の TSP の近似アルゴリズムと比較した．実験は点の数ごとに C++ のランダム関数で 100 通りの問題例を生成し，それらにに対して計算を行いその平均を求めた（図 4，表 1）．プログラムの実行は Sun SPARC Ultra-1 ワークステーション上で行った．また，点の数が 113,336,1793,14337 は Karp のアルゴリズムで分割を終了したときの各領域内の点の数がすべて 8 になる問題例である．実験の結果 Divide-and-Sort が各アルゴリズムと比べ最も速い事がわかる．とくに，実行時間が $O(n \log n)$ である Karp のアルゴリズムと比べても速いことが確認できる．また，点の数が非常に大きくなっても現実的な時間で対処できることがわかる．

表 1: 各アルゴルズムの実行時間

Points	Div-&-Sort	Karp 3	Karp 8	N Neighbor	D Span	L–K	2-opt	Arora
100	0.0005	0.0040	0.170	0.0091	0.0858	26.2	0.508	165
113			*0.604*					
200	0.0008	0.0091	0.376	0.0378	0.363	293	2.50	586
225			*1.21*					
1000	0.0066	0.0444	1.00	0.909	10.27		102	
1793			*9.72*					
2000	0.0115	0.0903	1.96	3.68	43.9			
10000	0.0694	0.533	1.80	92.9				
14337			*77.8*					
100000	0.879	6.00	130					

6.3 精度の比較

Divide-and-sort の解の精度を他の TSP の近似アルゴリズムと比較した. 実験は各点の数に対して C++ のランダム関数で 100 通り問題例を与え計算を行った. そして, divide-and-sort の解の長さを 1 として他のアルゴリズムの解の長さとの比率を求め, その平均を求めた (図 5). Divide-and-sort の分割数は $\left\lceil \frac{1}{3.4}\sqrt{\frac{l_V}{l_H}}\sqrt{n} \right\rceil$ である. 各近似アルゴリズムと比較すると, Lin–Kernighan や Arora, 2-Opt ほど良い精度の解を求めことはできないが, Karp のアルゴリズムのように比較的計算量が小さいアルゴリズムより良い解を求めることができる.

6.4 TSPLIB との比較

TSPLIB[16] の問題例に対し各アルゴリズムで解法を行い, TSPLIB の解の長さを 1 として各アルゴリズムの解の評価値の比率を求めた (表 2). TSPLIB の解は最適解でなので表 2の値は最適解との長さの比になる. Lin–Kernighan と 2-Opt は 1 回のみ計算を行った. Arora は各問題例に対し 50 回計算を行いその中で最も評価値のよい解を選んだ. Divide-and-sort の分割数は $\left\lceil \frac{1}{3.4}\sqrt{\frac{l_V}{l_H}}\sqrt{n} \right\rceil$ である.

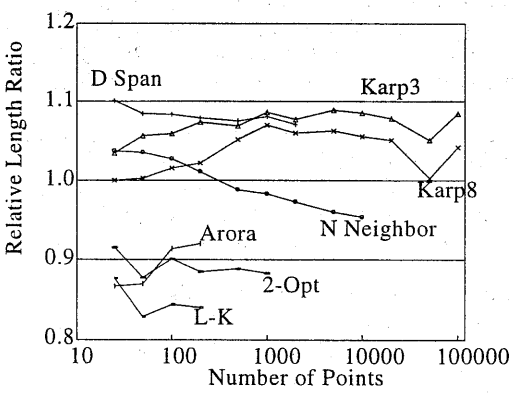

図 5: 精度の比較 (divide-and-sort の解の長さを 1 としたときの解の長さ)

表 2: TSPLIB との精度の比較（最適解の長さを 1 としたときの各アルゴリズムの解の長さ）

Name	Points	Div-&-Sort	Karp 8	N Neighbor	D Span	L-K	2-opt	Arora
eli101	101	1.17	1.11	1.29	1.29	1.05	1.13	1.12
kroA100	100	1.38	1.24	1.26	1.28	1.04	1.05	1.11
lin105	105	1.36	1.49	1.27	1.03	1.14	1.12	1.10
pr1002	1002	1.54	1.42	1.22	1.33	1.06	1.09	1.24
st70	70	1.23	1.30	1.19	1.29	1.04	1.08	1.10

7 まとめ

本アルゴリズムは，実験により非常に高速であることが確認できた．また，最悪の場合の近似率が $O(\sqrt{n})$ であるにも関わらず，他の速度重視のアルゴリズムと較べよりよい解を求めることができる．また，本アルゴリズムは非常に単純な構成で実装が容易にできる．

以上のことから Divide-and-sort は時間のコストを節約したいときに非常に有効な手段である．

参考文献

[1] S. Arora, Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems, *J. ACM* 45 (5), 753–782, 1998.

[2] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, M. Protasi, *Complexity and Approximation*, Springer, 1999.

[3] R. E. Bellman, Dynamic programming treatment of the traveling salesman problem, *J. ACM* 9, 61–63, 1962.

[4] J. Beardwood, J. H. Halton, J. M. Hammersley, The shortest path through many points, In *Proc. Cambridge Philosophical Society* 55, 299–327, 1959.

[5] N. Christofides, Worst-case analysis of a new heuristic for the traveling salesman problem, Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA., 1976.

[6] M. R. Garey, R. L. Graham, D. S. Johnson, Some NP-complete geometric problems, In *Proc. the Eighth ACM Symposium on Theory of Computing*, 10–22, 1976.

[7] J. H. Halton, R. Terada, A fast algorithm for the Euclidean traveling salesman problem, optimal with probability one, *SIAM J. Comput.* 11, 28–46, 1982.

[8] M. Held, R. M. Karp, A dynamic programming approach to sequencing problems, SIAM J. 10, 196–210, 1962.

[9] R. M. Karp, Reducibility among combinatorial problems, *Complexity of Computer Computations*, Plenum Press, 85–103, 1972.

[10] R. M. Karp, Probabilistic analysis of partitioning algorithms for the traveling-salesman problem in the plane, *Math. Oper. Res.* 2, 209–224, 1977.

[11] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, D.B. Shmoys, *The Traveling Salesman Problem*, Wiley, 1992.

[12] S. Lin, B. W. Kernighan, An effective heuristic algorithm for the traveling-salesman problem, *Oper. Res.* 21, 498–516, 1973.

[13] C. H. Papadimitriou, Euclidean TSP is NP-complete, *Theoret. Comput. Sci.* 4, 237–244, 1977.

[14] C. H. Papadimitriou, The complexity of the Lin-Kernighan heuristic for the traveling salesman problem, *SIAM J. Copmut.* 21, 450–465, 1992.

[15] D. J. Rozenkrantz, R. E. Stearns, P. M. Lewis II, An analysis of several heuristics for the traveling salesman problem, *SIAM J. Comput.* 6 (3), 563–581, 1977.

[16] TSPLIB, Traveling salesman problem library, `http://www.iwr.uni-heidelberg.de/iwr/comopt/software/TSPLIB95/`