Smoothed Analysis for Partitioning Algorithms for d-dimensional Euclidean Optimization Problems

Nick Hoogendijk

July 1, 2018

1 Introduction

One of the classes of problems in combinatorial optimization is the class of Euclidean optimization problems. In a Euclidean optimization problem, we are give a set X of points in $\mathbb{R}^d$. Every point in X is connected to all other points in X, like a complete graph. The length of the edge between points $x, y \in X$ is the Euclidean distance $\| x - y \|$. Examples of Euclidean optimization problems are the Euclidean traveling salesman problem and the Euclidean Steiner tree problem. Current algorithms for most of these problems do not have polynomial-time complexity or do have polynomial-time complexity, but are sometimes too slow to solve the problem for large instances.

Solving the problem faster can be done with partitioning heuristics. This means that we divide a large instance in smaller instances, solve the problem for each of these instances and then join these solutions to one solution for the main instance. This has the consequence that this solution might not be an optimal solution, but can be close to it. Partitioning heuristics work well and fast in practice [9,12], but the worst-case analysis contradicts this. Worst-case analysis measures the complexity of an algorithm in the worst case. This type of analysis is often used in analyzing algorithms, because it gives an upper bound on the performance. A low worst-case performance results in a good performance for the algorithm, but a high worst-case performance does not necessarily mean that the algorithm is bad, as can be seen from some of the partitioning heuristics. When explaining the performances of partitioning heuristics with worst-case analysis, it shows that these algorithms can be slow and output solutions that are far from optimal. The reason for this is that the worst-case instances are usually not typical for the problem and occur rarely in practice. Therefor, worst-case analysis is too pessimistic for most algorithms of this type.

Another way to analyze the performance of an algorithm is average-case analysis. For this type of analysis, a distribution of the inputs is needed. Then, the expected performance of the algorithm is calculated, given that the inputs are drawn from the distribution. The distribution used in average-case analysis, is ideally the distribution of the instances that occur in practice, but it is often hard to determine or cleanly express this distribution. Also, in different application, the distribution of inputs can vary greatly [15].

To overcome the drawbacks of worst-case and average-case analysis, smoothed analysis was developed. Spielman and Teng [14] introduced smoothed analysis to explain the performance of the simplex method for linear programming. In smoothed analysis, an adversary specifies an instance and this instance is then slightly perbuted [2]. This can be motivated by the fact that practical data is often subject to a small amount of noise. This noise can be modeled by the perbutation. For example: in industrial optimization and economic prediction, the input parameters could be obtained by physical measurements, and measurements usually have some of low magnitude uncertainty [15].

Bläser, Manthey and Rao [2], developed a framework for smoothed analysis of partitioning heuristics for 2-dimensional Euclidean optimization problems. In this paper, we will generalize this framework, so it can be applied on higher

dimensions. We will mostly follow the outline of Bläser, Manthey and Rao's article [2], with certain definitions and theorems changed where needed.
We develop a model where the adversary specifies n density functions $f_1, ..., f_n : [0,1]^d \to [0, \phi]$, one for each point. Then we draw x_i independently to their corresponding density function to get the actual point set. ϕ lets us control the adversary's power, the larger ϕ, the more powerful the adversary becomes. A more detailed explanation of this model is given in 2.2.
After we have developed the framework, we will apply it to certain partitioning algorithms for Euclidean Minimum Matching (Sect. 4), Euclidean Steiner trees (Sect. 5), Euclidean traveling salesman problem (Sect. 6) and Euclidean degree-bounded minimum-spanning tree (Sect. 7).

2 Preliminaries

We denote probabilities as $\mathbb{P}$ and expected values as $\mathbb{E}$.

2.1 Euclidean Functionals

A *Euclidean functional* is a function $F : ([0,1]^d)^* \to \mathbb{R}$ that maps a finite set of points to a real number. The following are examples of Euclidean functionals:

- MM maps a point set to the length of its minimum-length perfect matching. (length means Euclidean distance, one point is left out if the cardinality of the point set is odd).
- TSP maps a point set to the length of its shortest Hamiltonian cycle, i.e. to the length of its optimal traveling salesman tour.
- MST maps a point set to the length of its minimum-length spanning tree.
- ST maps a point set to the length of its shortest Steiner tree.
- dbMST maps a point set to the length of its minimum-length spanning tree, restricted to tree of maximum degree at most b for some bound b.

The Euclidean functionals that we will use in this paper are all connected to certain optimization problems. As can be seen from the examples of Euclidean functionals above, the functional of a problem maps a point set to the value of its optimal solution. This shows to be useful when bounding the approximation ratio (subsec. 3.2).

To put some restrictions on functionals, we have the following definitions.

A Euclidean functional F is *smooth* [17] if there exists a constant c such that

$$|F(X \cup Y) - F(X)| \leq c|Y|^{(d-1)/d}$$

holds for all finite $X, Y \subset [0,1]^d$. The constant c only depends on F and on d, but not on X or Y. A smooth functional means that when a small amount of points is added to the point set X, the functional does not change much. A functional is required to be smooth for Theorem 2.1 to hold. The functionals MM, TSP, MST, ST and dbMST are smooth [17].
Let $C_1, ..., C_s$ be a partitioning of $[0,1]^d$ into hypercubes. These hypercubes do not necessarily have to have the same size. Each C_l is called a *cell*. Furthermore, we denote our point set by $X \subset [0,1]^d$, $X_l = X \cap C_l$, $n_l = |X_l|$ and $\text{diam}(C_l)$ as the diameter of C_l.
F is *near-additive* if for all partitions $C_1, ..., C_s$ and for all finite $X \subset [0,1]^d$ we have:

$$\left| F(X) - \sum_{i=1}^{s} F(X_i) \right| = O\left(\sum_{i=1}^{s} \text{diam}(C_i) \right).$$

Near-addivity is used to bound the approximation for Euclidean minimum matching (subsec. 4.2).
The functionals MM, TSP, MST, ST and dbMST are near-additive [2].
Rhee [13] proved a useful theorem which we will use to bound the probability that F assumes a too small function value.

Theorem 2.1 (Rhee [2], [13])**.** *Let X be a set of n points drawn independently according to distributions from $[0,1]^d$. Let F be a smooth Euclidean functional. Then there exist constants c and c' such that for all $t > 0$, we have*

$$\mathbb{P}[|F(X) - \mathbb{E}[F(X)]| > t] \leq c' \cdot \exp\left(\frac{c}{n} t^{2d/(d-1)} \right).$$

Remark 2.1.1. *As mentioned by Bläser, Manthey and Raghavendra Rao's [2], Rhee proved this theorem for the case that $x_1, .., x_n$ are identically distributed. However, Rhee mentioned it herself that the proof carries over when $x_1, ..., x_n$ are drawn independently, but their distributions are not necessarily identical.*

2.2 Smoothed Analysis

In the classical model of smoothed analysis [14], an adversary specifies a point set $\bar{X}$ and then this point set is perbuted by independent identically distributed random variables to obtain the input set X. The distributions all had a variance of σ^2, which controls the size of the perbutations.
Bläser, Manthey and Rao [2] explained a different view-point: the adversary can also specify the means of the probability distributions according to which the point set is drawn. Then they generalised this view-point as follows: For smoothed analysis, we let an adversary choose a probability distribution function for every point. Then points are drawn independently according to each density function. To not give the adversary too much power, we will set an upper bound ϕ for the density functions. The adversary can then choose any density function $[0,1]^d \to [0, \phi]$. If $\phi = 1$, we get the uniform distribution on the unit hypercube. If ϕ gets larger, the adversary gets more power and can choose a more precise location for the points. This is all summarized in the following model.

Model 2.1 ([2]). *Let $\phi \geq 1$. An adversary specifies n density function $f_1, ..., f_n : [0,1]^d \to [0\phi]$. We write $f = (f_1, ..., f_n)$ for short. We define $X = \{x_1, ..., x_n\}$ where x_i is drawn according to f_i, independently from the other points for $i = 1, ..., n$.*

We write $X \sim f$ if X is drawn as described above. If P is a performance measure algorithm (like running-time), then the smoothed performance is $\max_f(\mathbb{E}_{X\sim f}[P(X)])$. Furthermore, for a Euclidean functional F, we let $\mu_F(n, \phi, d)$ be a lower bound for the expected value of F if X is drawn according to the probabilistic model described above. So μ_F is a function such that $\mu_F(n, \phi, d) \leq \min_f(\mathbb{E}_{X\sim f}[F(X)])$.

3 Framework

Now we will present our framework generalized to higher dimensions based on Bläser, Manthey and Rao's framework [2].
Let A_{opt} be an optimal algorithm for some smooth and near-additive Euclidean functional F, and let A_{join} be an algorithm that combines solutions for each cell into a global solution. We assume that A_{join} runs in time linear in the number of cells. Then we obtain the following algorithm, which we call A.

Algorithm 3.1 (generic algorithm A, [2]).
Input: *$X \subset [0,1]^d$ of n points.*
1. *Divide $[0,1]^d$ into s cells.*
2. *Use A_{opt} to compute optimal solutions for each cell.*
3. *Use A_{join} to join the s solution into a solution for X.*

The cells in step 1 of Algorithm 3.1 are multi-dimensional boxes and are not necessarily have the same size or contain the same number points.

We also make the following assumptions on ϕ. We will note whenever we use them.

Assumption 3.1 ([2]).
1. *$\phi = O(s)$. This basically implies that the adversary cannot concentrate all points in a too small number of cells.*
2. *$\phi = \omega(s \log(n)/n)$. This provides a lower bound for the probability mass in a full cell. (Full is defined as in subsec. 3.1).*
3. *$\phi = o(\sqrt{n/\log(n)})$. With this assumption, the tail bound of Theorem 2.2 becomes sub-polynomial.*

The first two assumptions are used for smoothed analysis of the running-time (subsec. 3.1). The third one is used for smoothed analysis of the expected approximation ration (subsec. 3.2).
These assumptions are not too restrictive: for the partitioning algorithms we analyze here, we have $s = O(n/\log^{O(1)}(n))$ (for matching, we could use smaller s while maintaining polynomial, albeit worse, running-time; for the other problems, we even need $s = O(n/\log(n))$. Ignoring poly-logarithmic terms, the first

and third assumption translate roughly to $\phi = O(n)$ and $\phi = o(\sqrt{n})$, respectively. The second assumption roughly says $\phi = \omega(1)$, but for $\phi = O(1)$, we can expect roughly average-case behaviour, because the adversary has only little influence on the positions.

3.1 Smoothed Running-Time

For smoothed analysis of the running-time, we assume that the cells are of the same size. then, we know that each cell has a content (hyper-dimensional volume) of $1/s$. The cumulative density of f_i in the cell C_l is $\int_{C_l} f_i(x)$. Since f_i is bounded from above by ϕ, we have $\int_{C_l} f_i(x) \leq \frac{\phi}{s}$. Then we call a cell C_l full if $\sum_{i=1}^{n}(\int_{C_l} f(x)\mathrm{d}x) = \frac{n\phi}{s}$. Note that $\sum_{i=1}^{n}(\int_{C_l} f(x)\mathrm{d}x)$ is the expected number of points in C_l.
The goal of the adversary is to define the density functions $f_1, ..., f_n$ in such a way, such that the partitioning function will be as slow as possible. Bläser, Manthey and Rao [2] showed that the adversary should make as much full cells as possible in order to slow down the partitioning algorithm.
We will use the following lemma for Theorem 3.2.

Lemma 3.1. *(Inverse Jensen's inequality [2]) Let T be any convex, monotonically increasing function that is bounded by a polynomial, and let B be a binomially distributed random variable with parameters $n \in \mathbb{N}$ and $p \in [0, 1]$ with $p = \omega(\log(n)/n)$. Then*

$$\mathbb{E}[T(N)] = \Theta(T(\mathbb{E}[B])).$$

Now we can bound the smoothed expected running-time of A from above.

Theorem 3.2 ([2])**.** *Assume that the running-time of A_{opt} can be bounded from above by a convex function T that is bounded by a polynomial. Then with model 2.1 and under assumptions 3.1(1), and 3.1(2), the expected running-time of A on input X is bounded from above by*

$$O\left(\frac{s}{\phi} \cdot T\left(\frac{n\phi}{s}\right) + s\right)$$

Proof. Let $T(n)$ denote the worst-case running-time of A_{opt} on n points.
The expected running-time is maximized if we have the maximum amount of full cells. We can have at most $\lfloor s/\phi \rfloor$ full cells plus possibly one cell containing all of the remaining mass probability. Then the number of points in any full cell is a binomially distributed random variable B with parameters n and ϕ/s. By linearity of expectation, the expected running-time is bounded by

$$\left(\left\lfloor \frac{s}{\phi} \right\rfloor + 1\right) \cdot \mathbb{E}[T(B)] + O(s).$$

With Assumption 3.1(1), we can bound this by $O(\frac{s}{\phi} \cdot \mathbb{E}[T(B)] + s)$. We can then apply Lemma 3.1 with $p = \phi/s$ for $\phi = \omega(s \log(n)/n)$ to get $O(\frac{s}{\phi} \cdot T(n\phi/s) + s)$. We can say that $\phi = \omega(s \log(n)/n)$ by using Assumption 3.1(2). □

Remark 3.2.1. *This is the same theorem as proven by Bläser, Manthey and Rao for two dimensions. We have given a short overview of their proof. For further details, we refer to their article [2].*

3.2 Smoothed Approximation Ratio

We can bound the value of A from above by

$$A(X) \leq \sum_{l=1}^{s} F(X_l) + J',$$

where J' is an upper bound for the joining cost of the solution of the individual cells. If we assume that F is a near-additive Euclidean functional, we have $A(X) \leq F(X) + J$ for $J = J' + O(\sum_{l=1}^{s} \operatorname{diam}(C_l))$. If we divide by $F(X)$, we get

$$\frac{A(X)}{F(X)} \leq 1 + O\left(\frac{J}{F(X)}\right). \tag{1}$$

Combining this with $\mathbb{E}[F(X)] \geq \mu_F(n, \phi, d)$, we can get a bound for

$$\frac{\mathbb{E}(A(X))}{\mathbb{E}(F(X))} \leq 1 + O\left(\frac{J}{\mu_F(n, \phi, d)}\right).$$

While this gives some guarantee on the approximation performance, it is not exactly what we want. We want to find an upper bound for $\mathbb{E}[A(X)/F(X)]$. To do this, we need strong upper bounds for the probability that $F(X)$ is close to 0. Theorem 2.1 is too weak for this, but we have two other ways to handle this problem. The first and easiest way is if A has a worst-case guarantee $\alpha(n, d)$ on its approximation ratio for any instance of n points. Then we can apply Theorem 2.1 to bound $F(X)$ from below. If $F(X) \geq \mu_F(n, \phi, d)/2$, then we can use (1) to obtain a ration of $1 + O(\frac{J}{\mu_F(n,\phi,d)})$. Otherwise, we obtain a ratio of $\alpha(n, d)$. If $\alpha(n, d)$ is not too large compared to the tail bound obtained from Theorem 2.1, then this contributes only little to the expected approximation ratio. The following theorem summarizes this.

Theorem 3.3. *Assume that A has worst-case approximation ratio of $\alpha(n, d)$ for any instance consisting of n points. Then, with Model 2.1, the expected approximation ratio of A is*

$$\mathbb{E}\left[\frac{A(X)}{F(X)}\right] \leq 1 + O\left(\frac{J}{\mu_F(n, \phi, d)} + \alpha(n, d) \cdot \exp\left(-\frac{c\mu_F(n, \phi, d)^{2d/(d-1)}}{n}\right)\right).$$

Proof. We have

$$\frac{A(X)}{F(X)} \leq \min\left\{1 + O\left(\frac{J}{F(X)}\right), \alpha(n, d)\right\}. \tag{2}$$

By Theorem 2.1 and Remark 2.1.1, we have

$$\begin{aligned}\mathbb{P}\left[F(X) < \frac{\mu_F(n,\phi,d)}{2}\right] &\leq \mathbb{P}\left[F(X) < \frac{\mu_F(n,\phi,d)}{2}\right] + \\ &\qquad \mathbb{P}\left[\frac{\mu_F(n,\phi,d)}{2} \leq F(X) < \mathbb{E}\left[F(X)\right] - \frac{\mu_F(n,\phi,d)}{2}\right] \\ &= \mathbb{P}\left[F(X) \leq \mathbb{E}[F(X)] - \frac{\mu_F(n,\phi,d)}{2}\right] \\ &= \mathbb{P}\left[\mathbb{E}[F(X)] - F(X) > \frac{\mu_F(n,\phi,d)}{2}\right] \\ &\leq c' \exp\left(-\frac{c\mu_F(n,\phi,d)^{2d/(d-1)}}{n}\right)\end{aligned}$$

for some constants $c', c > 0$. Together with (2), we can get the following bound on the approximation ratio.

$$\mathbb{E}\left[\frac{A(X)}{F(X)}\right] \leq 1 + O\left(\frac{J}{\mu_F(n,\phi,d)} + \alpha(n,d) \cdot \exp\left(-\frac{c\mu_F(n,\phi,d)^{2d/(d-1)}}{n}\right)\right).$$

□

In the case that we do not have an upper bound for the worst-case approximation ratio of A, we need to find a way to bound $\mathbb{E}[1/F(X)]$. Note that we do not explicitly provide an upper bound for $\mathbb{E}[1/F(X)]$, but only a sufficiently strong tail bound $h_{n,d}$ for $1/F(X)$.

Theorem 3.4. *Assume that there exists a $\beta_d \leq J$ and a function $h_{n,d}$ such that $\mathbb{P}[F(X) \leq x] \leq h_{n,d}(x)$ for all $x \in [0, \beta_d]$. Then with model 2.1 the expected approximation ratio of A is*

$$\mathbb{E}\left[\frac{A(X)}{F(X)}\right] \leq 1+O\left(J \cdot \left(\frac{1}{\mu_F(n,\phi)} + \frac{\exp(-c\mu_F(n,\phi,d)^{2d/(d-1)})}{\beta_d} + \int_{1/\beta_d}^{\infty} h_{n,d}\left(\frac{1}{x}\right) \mathrm{d}x\right)\right).$$

Proof. If $F(X) \geq \mu_F(n,\phi,d)/2$, then the approximation ratio is

$$1 + O\left(\frac{J}{\mu_F(n,\phi,d)}\right).$$

By Theorem 2.1, the probability that this does not hold is bounded from above by $\exp(-\frac{c\mu_F(n,\phi,d)^{2d/(d-1)}}{n})$ for some constant $c > 0$. If we still have $F(X) \geq \beta_d$, then we can bound the ratio from above by

$$1 + O\left(\frac{J}{\beta_d}\right).$$

This contributes

$$\exp\left(-\frac{c\mu_F(n,\phi,d)^{2d/(d-1)}}{n}\right) \cdot \left(1 + O\left(\frac{J}{\beta_d}\right)\right) \leq \exp\left(-\frac{c\mu_F(n,\phi,d)^{2d/(d-1)}}{n}\right) \cdot O\left(\frac{J}{\beta_d}\right)$$

to the expected value, where the inequality follows from $\beta_d \leq J$. We are left with the case that $F(X) \leq \beta_d$. This case contributes

$$J \cdot \int_{1/\beta_d}^{\infty} \mathbb{P}\left[\frac{1}{F(X)} \geq x\right] \mathrm{d}x \leq J \cdot \int_{1/\beta_d}^{\infty} h_{n,d}\left(\frac{1}{x}\right) \mathrm{d}x$$

to the expected value, which completes the proof.

□

4 Minimum Matching

In minimum matching, we want every point to be connected (matched) to exactly one other point. In case there are an odd number of points, one point is left out of the matching. A minimum matching is a matching such that the sum of the length of connections is the smallest over all possible matchings.
We will use an algorithm that is based on the two-dimensional version of Dyer's and Frieze's.

Algorithm 4.1 (A_{MM})**.**
Input: *$X \subset [0,1]^d$ of n points, n is even.*

1. *Partition $[0,1]^d$ into $s = k^d$ equal-sized sub-hypercubes $C_1, ...C_{k^d}$ each of side length $1/k$, where $k = (\frac{\sqrt{n}}{\log(n)})^{1/(d-1)}$.*
2. *Compute minimum-length perfect matching for C_l for $l = 1, .., k^d$*
3. *Compute a matching for the unmatched points from the previous step using the strip heuristic [16].*

Let A_{MM} be the cost of the matching made by the algorithm above on the input $X = \{x_1, .., x_n\}$ and let $\mathrm{MM}(X)$ be the cost of a perfect minimum matching of minimum cost. The choice for $k = (\frac{n}{\log(n)})^{1/(d-1)}$ in the above algorithm is made such that there is a balance between running-time and approximation ratio. Other values for k can be chosen.

4.1 Smoothed Running-Time

A minimum-cost perfect matching can be found in time $O(n^3)$ [1]. By Theorem 3.2, we get the following theorem:

Theorem 4.1. *With Model 2.1 and Assumption 3.1(1) and 3.1(2), the expected running-time of A_{MM} is at most*

$$O\left(\frac{n^3\phi^2}{k^4} + k^2\right).$$

If we then plug in $k = (\frac{\sqrt{n}}{\log(n)})^{1/(d-1)}$, we get

$$O\left(n^{1+2d/(d-1)}\phi^2 \log^{4/(d-1)}(n) + \left(\frac{n}{\log^2(n)}\right)^{1/(d-1)}\right).$$

4.2 Smoothed Approximation Ratio

We first derive a lower bound for $\mu_{\mathrm{MM}}(n, \phi, d)$. For this we use the nearest-neighbor graph for the point set $X \subseteq [0,1]^d$. Let $\mathrm{NN}(X)$ denote the total edge length of this nearest-neighbor graph. So

$$\mathrm{NN}(X) = \sum_{x \in X} \min_{y \in X : y \neq x} \| x - y \| .$$

We can use $\mathrm{NN}(X)$ to bound $\mathrm{MM}(X)$ from below: $\mathrm{MM}(X) \geq \mathrm{NN}(X)/2$. Furthermore, $\mathbb{E}[\mathrm{NN}(X)]$ is easier to than $\mathbb{E}[\mathrm{MM}(X)]$.

Lemma 4.2. *With model 2.1, we have*

$$\mathbb{E}[\mathrm{NN}(X)] = \Omega\left(\frac{n^{d/(d-1)}}{\phi^{1/d}}\right).$$

Proof. By linearity of expectation, we have $\mathbb{E}[\mathrm{NN}(X)] = n \cdot \mathbb{E}[\min_{i \geq 2} \| x_1 - x_i \|]$. Thus we have to prove that $\mathbb{E}[\min_{i \geq 2} \| x_1 - x_i \|] = \Theta((\phi n)^{-1/d})$. To bound this from below, we assume that x_1 is fixed by an adversary and that only $x_2, ..., x_n$ are drawn independently according to their density functions. Then we obtain

$$\begin{aligned}
\mathbb{E}\left[\min_{i \geq 2} \| x_1 - x_i \|\right] &= \int_0^\infty \mathbb{P}\left[\min_{i \geq 2} \| x_1 - x_i \| \geq r\right] \mathrm{d}r \\
&= \int_0^\infty \prod_{i=2}^{n} \left(1 - \mathbb{P}\left[\| x_1 - x_i \| \leq r\right]\right) \mathrm{d}r \\
&\geq \int_0^{\frac{1}{2}(\phi n)^{-1/d}} \prod_{i=2}^{n} \left(1 - \mathbb{P}\left[\| x_1 - x_i \| \leq r\right]\right) \mathrm{d}r
\end{aligned}$$

We can bound $\mathbb{P}[\| x_1 - x_i \| \leq r]$ from above by ϕ times the volume of a hypercube of side $2r$, which is $\phi(2r)^d$. So we get

$$\begin{aligned}
\mathbb{E}\left[\min_{i \geq 2} \| x_1 - x_i \|\right] &\geq \int_0^{\frac{1}{2}(\phi n)^{-1/d}} \left(1 - \phi(2r)^d\right)^{n-1} \mathrm{d}r \\
&\geq \int_0^{\frac{1}{2}(\phi n)^{-1/d}} \left(1 - \frac{1}{n}\right) \mathrm{d}r \geq \frac{1}{2e(\phi n)^{1/d}}.
\end{aligned}$$

The second equality holds, because $1 - \phi(2r)^d \geq 1 - \frac{1}{n}$ for $r \in [0, \frac{1}{2}(\phi n)^{-1/d}]$. The third equality uses $(1 - \frac{1}{n})^{n-1} \geq 1/e$. □

Since MM is near-additive and the diameter of each cell is $O(\sqrt{d}/k)$, we can use

$$J = O\left(\sum_{l=1}^{k^d} \mathrm{diam}(C_l)\right) = O(\sqrt{d} k^{d-1}). \tag{3}$$

We cannot bound the worst-case approximation ratio of Dyer and Frieze's partitioning algorithm. Thus, we cannot apply Theorem 3.3, but we have to use Theorem 3.4. To use this theorem, we need a tail bound for $1/\mathrm{MM}(X)$.

Lemma 4.3. *With model 2.1, we have*

$$\mathbb{P}[\mathrm{MM}(X) \leq c] \leq (2\phi c)^{n/2}$$

for all $c \leq \frac{1}{2}(2d)^{-1/(d-1)}$.

Proof. The proof is similar to Bläser, Manthey and Rao's proof of Lemma 4.4 in their article [2]. □

With this tail bound for $1/\mathrm{MM}(X)$, we can bound the smoothed approximation ratio.

Theorem 4.4. *With Model 2.1 and Assumption 3.1(3), we have*

$$\mathbb{E}\left[\frac{A_{\mathrm{MM}}(X)}{MM(X)}\right] \leq 1 + O\left(\sqrt{d}(2d)^{1/(d-1)}\frac{\sqrt{\phi}}{\log(n)}\right).$$

Proof. We apply Theorem 3.4. For this, we choose $\beta_d = \frac{1}{2\phi}(2d)^{-1/(d-1)}$. Lemma 4.2 allows us to choose $h_{n,d}(x) = (2\phi(2d)^{d-1}x)^{n/2}$ and yields

$$\begin{aligned}\int_{1/\beta}^{\infty} h_{n,d}\left(\frac{1}{x}\right)\mathrm{d}x &= (2\phi(2d)^{d-1})^{n/2} \cdot \int_{1/\beta}^{\infty}\left(\frac{1}{x}\right)^{n/2}\mathrm{d}x \\ &= (2\phi(2d)^{d-1})^{n/2} \cdot \frac{2\beta^{n/2-1}}{n-2} \\ &= \frac{4\phi}{n-2}(2d)^{1/(d-1)}.\end{aligned}$$

With (3) we get

$$J \cdot \frac{4\phi}{n-2}(2d)^{1/(d-1)} = O\left(\sqrt{d}k^{d-1}\frac{\phi}{n}(2d)^{1/(d-1)}\right)$$

and if we plug in $k = (\frac{\sqrt{n}}{\log(n)})^{1/(d-1)}$, we get

$$O\left(\sqrt{d}(2d)^{1/(d-1)}\frac{\phi}{\sqrt{n}\log(n)}\right) = o\left(\sqrt{d}(2d)^{1/(d-1)}\frac{\sqrt{\phi}}{\log(n)}\right).$$

The second equality holds, because of Assumption 3.1(3).
We can choose $\mu_{\mathrm{MM}}(n,\phi,d) = \Omega(n^{(d-1)/d}/\phi^{1/d})$ as $\mathrm{MM}(X) \geq \mathrm{NN}(X)/2 = \Omega(n^{(d-1)/d}/\phi^{1/d})$ by Lemma 4.2. Then we get

$$\begin{aligned}\exp\left(\frac{-c\mu_{\mathrm{MM}}(n,\phi d)^{\frac{2d}{d-1}}}{n}\right) &= \exp\left(\frac{-c\left(\frac{n^{(d-1)/d}}{\phi^{1/d}}\right)^{2d/(d-1)}}{n}\right) \\ &= \exp\left(-c\frac{n}{\phi^{2/(d-1)}}\right).\end{aligned}$$

Combining this with Assumption 3.1(3), this expression decreases faster to zero than any polynomial. So a polynomial times this expressions decreases also faster to zero than any polynomial, so it is negligible.

$$\frac{J}{\beta} \cdot \exp\left(\frac{-c\mu_{\mathrm{MM}}(n,\phi d)^{\frac{2d}{d-1}}}{n}\right) = 2\phi(2d)^{1/(d-1)}\sqrt{d}k^{d-1}\exp\left(\frac{-c\mu_{\mathrm{MM}}(n,\phi d)^{\frac{2d}{d-1}}}{n}\right)$$

is so small, it is negligible compared to the rest. Altogether, we get a bound of

$$1 + O\left(\frac{J}{\mu_{\mathrm{MM}}(n,\phi,d)}\right) + o\left(\sqrt{d}(2d)^{1/(d-1)}\frac{\sqrt{\phi}}{\log(n)}\right) = 1 + O\left(\sqrt{d}(2d)^{1/(d-1)}\frac{\sqrt{\phi}}{\log(n)}\right).$$

□

5 Euclidean Steiner Tree

The Steiner tree problem is about finding the minimum spanning tree, but we have the possibility to add points, such that the total cost becomes lower.
Kalpakis and Sherman [7] designed an algorithm for the Euclidean minimum Steiner tree problem. The algorithm goes as follows

Algorithm 5.1 (A_{KS})**.**
Input: *set $X \subset [0,1]^d$ of n points.*

1. *Let $s = n/\log(n)$. Partition $[0,1]^d$ into $\Theta(s)$ cells such that each cell contains at most $n/s = \log(n)$ points.*
2. *Solve the Steiner tree problem optimally within each cell.*
3. *Compute a minimum-length spanning tree to connect the forest thus obtained.*

It is shown that the running-time of this algorithm on a graph with t vertices equals $\frac{t^3}{2}+t^2(2^{t-1}-t-1)+t(3^{t-1}-2^t+3)/2$ [7], which is polynomial if $t = \log(n)$, as it is in the above algorithm. We cannot use our framework for the running-time estimation since the algorithm partitions $[0,1]^d$ into cells, based on the point set, such that in each cell there is an equal amount of points. Therefor the running time is set and only depends on the total number of points.
We can use our framework for the approximation ratio. Let $\mathrm{KS}(X)$ be the cost of the Steiner Tree computed by Algorithm 5.1. Let $\mathrm{ST}(X)$ be the cost of a minimum Steiner Tree for the point set X and $\mathrm{MST}(X)$ be the cost of a minimum-length spanning tree of X. Kalpakis and Sherman showed that

$$\begin{aligned} KS(X) &\leq ST(X) + O\left(1 + \log(n)^{\frac{d-2}{d-1}}\left(\frac{n}{\log(n)}\right)^{\frac{d-1}{d}}\right) \\ &= ST(X) + O\left(\log(n)^{\frac{d-2}{d-1}}\left(\frac{n}{\log(n)}\right)^{\frac{d-1}{d}}\right) \end{aligned}$$

Thus let $J = O\left(\log(n)^{\frac{d-2}{d-1}}\left(\frac{n}{\log(n)}\right)^{\frac{d-1}{d}}\right)$. Since minimum spanning trees are $2/\sqrt{3}$ approximations for Euclidean Steiner trees [4], we have $\mathrm{ST}(X) \geq \frac{\sqrt{3}}{2} \cdot \mathrm{MST}(X)$. Furthermore, we have $\mathrm{MST}(X) \geq \mathrm{NN}(X)$. Thus, we can choose $\mu_{\mathrm{ST}}(n, \phi, d) = \Omega(n^{(d-1)/d}/\phi^{1/d})$ by Lemma 4.2. Luckily, KS comes with a worst-case approximation ratio of $\alpha(n, d) = O(n)$. The reason is that for any two points $x, y \in X$, we have $\parallel x - y \parallel \leq \mathrm{ST}(X)$ by Lemma 5.1 and since the cost of a tree is at most n times the maximum distance between two points in the tree, we have $\mathrm{KS}(X) \leq n \cdot \max_{x,y \in X} \parallel x - y \parallel \leq n \cdot ST(X)$.

Lemma 5.1. *Let T be a tree on the point set $X \subseteq [0,1]^d$. Let $C(T)$ be the cost of T (the sum of the length of the edges). Then $\parallel x - y \parallel \leq C(T)$.*

Proof. Let $x_m, y_m \in X$ be two points such that $\parallel x_m - y_m \parallel$ is maximum for all $x_m, y_m \in X$. Let $C(x, y)$ be the cost of the path between x and y in tree T. Then $\parallel x_m - y_m \parallel \leq C(x_m, y_m)$ by the triangle inequality and $C(x, y) \leq C(T)$ for any $x, y \in X$. Now we have

$$\parallel x - y \parallel \leq \parallel x_m - y_m \parallel \leq C(x, y) \leq C(T).$$

□

Since Kalpakis and Sherman's partitioning algorithm outputs at most a linear number of edges, we have $\mathrm{KS}(X) \leq O(n \cdot \mathrm{ST}(X))$. Now that we have an worst-case approximation ratio of $O(n)$, we can derive the expected approximation ratio

Theorem 5.2. *With Model 2.1 and Assumption 3.1(3), the expected approximation ratio of* KS *is*

$$\mathbb{E}\left[\frac{\mathrm{KS}(X)}{\mathrm{ST}(X)}\right] \leq 1 + O\left(\left(\frac{\phi}{\log(n)^{1/(d-1)}}\right)^{1/d}\right).$$

Proof. We use Theorem 3.3. If we substitute $J = O\left(\log(n)^{\frac{d-2}{d-1}}\left(\frac{n}{\log(n)}\right)^{\frac{d-1}{d}}\right)$, $\mu_{\mathrm{ST}}(n, \phi, d) = \Omega(n^{(d-1)/d}/\phi^{1/d})$ and $\alpha(n, d) = O(n)$, we get

$$\begin{aligned}\mathbb{E}\left[\frac{\mathrm{KS}(X)}{\mathrm{ST}(X)}\right] &\leq 1 + O\left(\frac{\log(n)^{\frac{d-2}{d-1}}\left(\frac{n}{\log(n)}\right)^{\frac{d-1}{d}}}{\frac{n^{(d-1)/d}}{\phi^{1/d}}} + n \cdot \exp\left(-\frac{c\left(\frac{n^{(d-1)/d}}{\phi^{1/d}}\right)^{\frac{2d}{d-1}}}{n}\right)\right) \\ &= 1 + O\left(\left(\frac{\phi}{\log(n)^{1/(d-1)}}\right)^{1/d} + n \cdot \exp\left(-c\frac{n}{\phi^{\frac{2}{d-1}}}\right)\right).\end{aligned}$$

With Assumption 3.1(3), $n \cdot \exp\left(-c\frac{n}{\phi^{2/(d-1)}}\right)$ goes faster to zero than any other polynomial, so we get

$$\mathbb{E}\left[\frac{\mathrm{KS}(X)}{\mathrm{ST}(X)}\right] \leq 1 + O\left(\left(\frac{\phi}{\log(n)^{1/(d-1)}}\right)^{1/d}\right).$$

□

6 Euclidean Traveling Salesman Problem

The traveling salesman problem is about finding a closed path that visits every point exactly once (the hamiltonian cycle), such that the total length is minimal.
Karp's partitioning scheme [8] is a partitioning algorithm for Euclidean TSP that computes near optimal solutions on average. Karp explained it for 2 dimensional space, but we can alter it so it can be used in d-dimensions.

Algorithm 6.1 (A_{KP})**.**
Input: *set $X \subset [0,1]^d$ of n points.*

1. *for $i = 1, .., d$*
   Keep partitioning the cells into $k = (\frac{n}{\log(n)})^{1/d}$ subcells, such that each subcell contains $n/(k^i)$ points. So eventually, there are k^d cells, each containing $\log(n)$ points.
2. *Compute optimal TSP tours for each cell.*
3. *Join the tours to obtain a TSP tour for X.*

The running-time of above algorithm is $2^t \cdot t^2 \cdot s + R(\text{join})$, where t is the number of points in a cell and $R(\text{join})$ is the running-time of the joining of the solutions. With our choice for $k = (\frac{n}{\log(n)})^{1/d}$ and the fact that $R(\text{join})$ is linear with the number of cells, the running time is polynomial. We can again not use our framework for the running-time for the same reason as we cannot apply our framework on the running time of Algorithm 5.1, but we can apply our framework on the approximation ratio.
Let $\mathrm{KP}(X)$ be the cost of the TSP tour created by the above algorithm and $\mathrm{TSP}(X)$ be the cost of an optimal TSP tour.

TSP has a worst-approximation guarantee, since for any two points $x, y \in X$, any tour must visit both x and y, its length is at least $2 \parallel x - y \parallel$ by the triangle inequality. Since a tour consists of n edges, any tour has a length of at most $\frac{n}{2} \cdot \mathrm{TSP}(X)$. So, we can use Theorem 3.3 with $\alpha(n) = n/2$. All we need to do is find J and $\mu_{\mathrm{TSP}}(n, \phi, d)$. As with Minimum Matching, the nearest-neighbor functional NN is a lower bound for TSP, so we get $\mu_{\mathrm{TSP}}(n, \phi, d) = \Omega(\frac{n^{(d-1)/d}}{\phi^{1/d}})$ from Lemma 4.2.
To find an upper bound for J, we first find an upper bound for J'. We notice we can join the solutions by a snake-like matter (see Figure 1 (Sect. 9)). The joining of two joined cells is at most the sum of the two diameters, except the cost of joining the first and the last cells (the upper left and bottom right respectively in fig. 1). This cost is at most the diameter of $[0,1]^d$, which is $\sqrt{d}$. Let p be an permutation, such that cell $C_{p(i)}$ is joined with $C_{p(i+1)}$ and $C_{p(n)}$ is joined with $C_{p(1)}$. Now we can bound J'.

$$
\begin{aligned}
J' &\leq \sqrt{d} + \sum_{l=1}^{k^d-1} \left(\operatorname{diam}(C_{p(i)}) + \operatorname{diam}(C_{p(i+1)})\right) \\
&\leq \sqrt{d} + \sum_{l=1}^{k^d} \left(\operatorname{diam}(C_{p(i)}) + \operatorname{diam}(C_{p(i+1)})\right) \\
&= \sqrt{d} + \sum_{l=1}^{k^d} 2 \cdot \operatorname{diam}(C_{p(i)}).
\end{aligned}
$$

Let C_{l_j} be the length of cell C_l in the direction of dimension j. Then we can bound the diameter of cell C_l from above by $\sum_{i=1}^{d} C_{l_j}$. We can then apply the same method as in Figure 2 (Sect. 9). Since step 1 in Algorithm 6.1 divides every subcell in k cells, we can bound J' from above by

$$
J' \leq \sqrt{d} + 2 \cdot \sum_{l=1}^{k^d} \operatorname{diam}(C_{p(i)}) = \sqrt{d} + 2 \cdot \sum_{i=1}^{d} k \cdot 1 = \sqrt{d} + 2dk.
$$

Since $J = J' + O(\sum_{i=1}^{n} \operatorname{diam}(C_l))$, we get $J = O(\sqrt{d} + 3dk)$.
We can now bound the approximation ratio for KP.

Theorem 6.1. *With Model 2.1 and Assumption 3.1(3), the expected approximation ratio of KS is*

$$
\mathbb{E}\left[\frac{\mathrm{KP}(X)}{\mathrm{TSP}(X)}\right] \leq O\left(1 + \frac{\phi^{1/d} d^{3/2} n^{2-d}}{\log(n)^{1/d}}\right).
$$

Proof. We apply theorem 3.3 and substitute $\alpha(n) = O(n)$, $\mu_{TSP}(n, \phi, d) = \Omega(\frac{n^{(d-1)/d}}{\phi^{1/d}})$ and $J = O(\sqrt{d} + 3dk)$. We get

$$
\mathbb{E}\left[\frac{\mathrm{KP}(X)}{\mathrm{TSP(X)}}\right] \leq O\left(1 + \frac{\sqrt{d} + 3dk}{\frac{n^{(d-1)/d}}{\phi^{1/d}}} + n \exp\left(-\frac{c\left(\frac{n^{(d-1)/d}}{\phi^{1/d}}\right)^{2d/(d-1)}}{n}\right)\right).
$$

□

As in the proof of Theorem 5.2, we know that the exp(...) part goes to zero faster than any polynomial with Assumption 3.1(3). So $n \cdot \exp(...)$ goes to zero faster than any polynomial, so it is negligible compared to the rest. We get

$$
\mathbb{E}\left[\frac{\mathrm{KP}(X)}{\mathrm{TSP(X)}}\right] \leq O\left(1 + \frac{\sqrt{d} + 3dk}{\frac{n^{(d-1)/d}}{\phi^{1/d}}}\right) = O\left(1 + \frac{\phi^{1/d} d^{3/2} n^{2-d}}{\log(n)^{1/d}}\right).
$$

7 Euclidean Degree-Bounded Minimum Spanning Tree

A b-degree-bounded minimum spanning tree is a spanning tree in which every point has at most a degree of b. For $2 \leq b \leq 4$, this problem is NP-hard, and it is solvable in polynomial time for $b \geq 5$ [11]. Let dbMST denote the Euclidean function that maps a point set to the length of its shortest b-degree-bounded minimum spanning tree. Bläser, Manthey and Rao proved that dbMST is a near-additive functional [2]. This implies that we can adapt Algorithm 6.1 to an partitioning algorithm for degree-bounded minimum spanning tree. We call this adapted algorithm P-bMST.

Algorithm 7.1 ($A_{\text{P-bMST}}$)**.**
Input: *set $X \subset [0,1]^d$ of n points.*

1. *for $i = 1, .., d$*
   Keep partitioning the cells into $k = (\frac{n \log(\log(n))}{\log(n)})^{1/d}$ subcells.
2. *Compute optimal degree-bounded minimum spanning tree solutions for each cell.*
3. *Join the tours to obtain a global solution for X.*

Our choice for k lets P-bMST run in polynomial-time as a degree-bounded minimum-length spanning tree on m points can be found in $2^{O(m \log(m))}$ using brute-force search. So again, we will not apply our framework on the running-time, but we will apply it on the approximation ratio.
We can use the same logic we applied for ST to get $\alpha(n,d) = O(n)$ for P-bMST. Since $\text{dbMST}(X) = \Omega(\text{NN}(X))$, we have $\mu_{\text{dbMST}}(n, \phi, d) = \Omega(\frac{n^{(d-1)/d}}{\phi^{1/d}})$ by Lemma 4.2. To determine J we can apply the same method as we did for determining J for TSP, but now we do not have to make a tour, so there is no need to add the diameter of $[0,1]^d$. We get $J = O(3dk) = O(d(\frac{n \log(\log(n))}{\log(n)})^{1/d})$. Now we can bound the approximation ratio of P-bMST.

Theorem 7.1. *With Model 2.1 and Assumption 3.1(3), we have*

$$\mathbb{E}\left[\frac{\text{P-bMST}(X)}{\text{dbMST}(X)}\right] \leq O\left(1 + \frac{d}{n^{d-1}}\left(\frac{\phi \log(\log(n))}{\log(n)}\right)^{1/d}\right).$$

Proof. We apply Theorem 3.3 and substitute $\alpha(n,d) = n$, $\mu_{\text{dbMST}}(n, \phi, d) = \Omega(\frac{n^{(d-1)/d}}{\phi^{1/d}})$ and $J = O(d(\frac{n \log(\log(n))}{\log(n)})^{1/d})$. We get

$$\mathbb{E}\left[\frac{\text{P-bMST}(X)}{\text{dbMST}(X)}\right] \leq O\left(1 + \frac{d(\frac{n \log(\log(n))}{\log(n)})^{1/d}}{\frac{n^{(d-1)/d}}{\phi^{1/d}}} + n \cdot \exp\left(-\frac{c\left(\frac{n^{(d-1)/d}}{\phi^{1/d}}\right)^{2d/(d-1)}}{n}\right)\right).$$

For the same logic as we applied in the proofs of Theorems 5.2 and 6.1, the $n \cdot \exp(...)$ is so small, it is negligible compared to the rest. So we get

$$\mathbb{E}\left[\frac{\text{P-bMST}(X)}{\text{dbMST}(X)}\right] \leq O\left(1 + \frac{d(\frac{n\log(\log(n))}{\log(n)})^{1/d}}{\frac{n^{(d-1)/d}}{\phi^{1/d}}}\right) = O\left(1 + \frac{d}{n^{d-1}}\left(\frac{\phi\log(\log(n))}{\log(n)}\right)^{1/d}\right).$$

□

8 Concluding remarks

We have shown that the framework developed by Bläser, Manthey and Rao [2] can be extended to higher dimensions. The results can be extended to distributions over $\mathbb{R}^d$ by scaling down the instance so that the inputs lie in $[0,1]^d$. We used the Euclidean distance for bounding J for TSP among others. This bound might change when using another metric.
This framework can be used for Euclidean partitioning algorithms combined with a smooth functional. Problems with no corresponding smooth functional, like minimum-weight triangulation, cannot be analyzed with this framework.

9 Figures

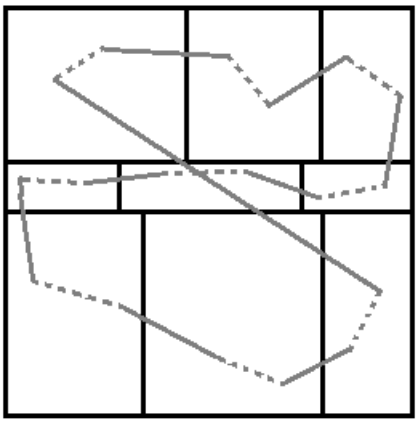

Figure 1: We can join the solutions of each cell in a snake-like way. Here the dotted lines are the detours between the two points according to the traveling salesman solution of each cell.

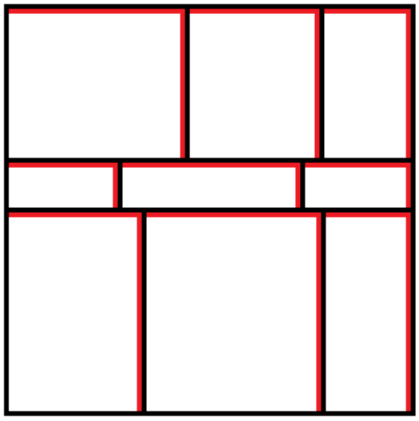

Figure 2: We can bound the diameter of a cell by the sum of its horizontal and vertical length (the red lines). As we can see, in each row, the sum of the horizontal length of each cell is just the horizontal length of $[0,1]^2$, which is 1. Since each row has k cells, we can sum the vertical length of the ith cell in each row, which is the vertical length of $[0,1]^2$, which is also 1.

References

[1] Ahuja, R.K., Magnanti, T.L. Orlin, J.B. *Network Flows: Theory, Algorithms, and Applications* Prentice-Hall, Englewood Cliffs (1993)

[2] Bläser, M., Manthey, B., Rao, B.V.R. *Smoothed Analysis of Partitioning Algorithms for Euclidean Functionals*, Algorithmica 2013

[3] Dreyfus, S.E., Wagner, R.A. *The Steiner problem in graphs.* Networks 1(3), 195-207 (1971)

[4] Du, D.-Z., Hwang, F.K. *A proof of the Gilber-Pollak conjecture on the Steiner ratio* Algorithmica 7(2&3), 121-135 (1992)

[5] Dyer, M.E., Frieze, A.M. *Partitioning algorithm for minimum weighted Euclidean matching* Information Processing Letters 1984

[6] Gary, M.R., Graham, R.L, Johnson, D.S. *The complexity of computing Steiner minimal trees.* SIAM J. Appl. Math. 32(4), 835-859 (1977)

[7] Kalpakis, K., Sherman, A.T. *Probabilistic Analysis of an Enhanced Partitioning Algorithm for the Steiner Tree Problem in* R^d. Networks 24(3), 147-159 (1994)

[8] Karp, R.M. *Probabilistic analysis of partitioning algorithms for the traveling-salesman problem in the plane.* Math. Oper. Res. 2(30, 209-224 (1977)

[9] Johnson, D.D., McGeoch, L.A. *Experimental analysis of heuristics for the STSP.* In: Gutin, G., Punnen, A.P. (eds.) *The Traveling Salesman Problem and Its Variations.* pp. 369-443. Kuwer Academic, Dordrecht (2002). Chapter 9.

[10] Papadimitriou, C.H. *The Euclidean traveling salesman problem is NP-complete.* Theor. Comput. Sci. 4(3), 237-244 (1977)

[11] Papadimitriou, C.H., Vazirani, U.V. *On two geometrix prblems related to the traveling salesman problem.* J. Algorithms 5(2), 231-246 (1984)

[12] Ravada, S., Sherman, A.T. *Experimental evaluation of a partitioning algorithm for the Steiner tree problem in* R^2 *and* R^3. Networks 24(8), 409-415 (1994)

[13] Rhee, W.T. *A Matching problem and subadditive Euclidean functionals*, The annals of Applied Probability 1993

[14] Spielman, D.A., Teng, S.-H. *Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time* J. ACM 51(3), 385-463 (2004)

[15] Spielman, D.A., Teng, S.-H. *Smoothed Analysis: An Attempt to Explain the Behavior of Algorithms in Practice* Communications of the ACM, pages 76–84, 2009.

[16] Supowit, K.J., Reingold, E.M. *Divide and conquer heuristics for minimum weighted Euclidean matching* SIAM J. Comput. 12(1), 118-143 (1983)

[17] Yukich, J.E. *Probability Theory of Classical Euclidean Optimization Problems* Springer 1998