On finding p-th nearest neighbours of scattered points in two dimensions for small p

George Goodsell

Department of Applied Mathematics and Theoretical Physics, University of Cambridge.

January 1997

Abstract

Given a large set of scattered points in the plane, we describe a new and efficient algorithm to find, for each point, the subset of p closest points, using the Dirichlet tessellation of the set of points, for small values of p. This problem has applications to interpolation and contouring.

Given a set, V, of n scattered data points in the plane,

$$V = \{\underline{x}_i : i = 1, 2, \ldots, n\}, \tag{1}$$

for large n, which are all different, and given a small positive integer p, we describe an algorithm to determine, for each data point, $\underline{x}_i \in V$, the subset, $V_i \subset V$, of the p data points closest to $\underline{x}_i$, using the Dirichlet tessellation of the set of n data points.

The Dirichlet tessellation partitions the plane so that each region (or tile) of the partition is the locus of points which are closer to a particular data point than to any other data point, see, for example, Green and Sibson [2]. Let $T_i \subset \Re^2$ be the tile which surrounds the data point $\underline{x}_i$. Then, given any point, $\underline{x} \in \Re^2$, we have that $\underline{x} \in T_i$ if and only if

$$\|\underline{x} - \underline{x}_i\|_2 = \min_{1 \leq j \leq n} \|\underline{x} - \underline{x}_j\|_2. \tag{2}$$

Each tile is the intersection of the open half-planes bounded by the perpendicular bisectors of the straight lines joining the data point which the tile surrounds with each of the other data points. Thus each tile is a convex polygon, and each tile which surrounds a data point on the boundary of the convex hull of the set of data points extends to infinity.

This problem, which has applications to interpolation and contouring, see Davis [1] and Powell [4], is described in Section 5.1 of Preparata and Shamos [5]. However, in Section 6.3.1 of Preparata and Shamos [5], the assertion is made that the Dirichlet tessellation has no means of dealing with this problem. In this paper, we shall prove the contrary; indeed, we shall show that the Dirichlet tessellation can be used to solve the problem with great efficiency. The key result we shall use is the following lemma, which is due to Powell [3].

Lemma Let $\underline{x}_i$, $i=1,2,\ldots,n$, be n different points in $\Re^2$ and let t be an integer from $[1,p]$. Although $\underline{x}_1$ can be any element of V, if $t\geq 2$ then the next $t-1$ points are ordered so that the conditions

$$\|\underline{x}_2-\underline{x}_1\|_2 \leq \|\underline{x}_3-\underline{x}_1\|_2 \leq \ldots \leq \|\underline{x}_t-\underline{x}_1\|_2 \leq \min_{t+1\leq j\leq n} \|\underline{x}_j-\underline{x}_1\|_2 \tag{3}$$

are satisfied. Further, for $i=1,2,\ldots,n$, we let S_i be the subset of $\{1,2,\ldots,n\}$ such that $j\in S_i$ if and only if the tiles, T_i and T_j, of the Dirichlet tessellation have a common edge of positive length, where we exclude i from S_i. Then every integer q in $[t+1,n]$ that satisfies the condition

$$\|\underline{x}_q-\underline{x}_1\|_2 \leq \|\underline{x}_j-\underline{x}_1\|_2, \qquad j=t+1,t+2,\ldots,n, \tag{4}$$

is in the set $S_1\cup S_2\cup\ldots\cup S_t$.

Proof Let $q\in[t+1,n]$ satisfy condition (4), so that

$$\|\underline{x}_q-\underline{x}_1\|_2 = \min_{t+1\leq j\leq n} \|\underline{x}_j-\underline{x}_1\|_2. \tag{5}$$

We consider the straight-line segment between $\underline{x}_1$ and $\underline{x}_q$. First we prove that, if $\underline{\widehat{x}}$ is any point on this line segment that satisfies $\underline{\widehat{x}}\in T_i$, where $i\neq q$, then the integer i is in the interval $[1,t]$.

Indeed, suppose that $\underline{\widehat{x}}\in T_i$, where $i\neq q$ is in the interval $[t+1,n]$. Then $\underline{x}_i\neq\underline{x}_1$ and $\underline{x}_i\neq\underline{x}_q$. Further, the position of $\underline{\widehat{x}}$, the definition of the Dirichlet tessellation and the triangle inequality provide the bound

$$\begin{aligned}
\|\underline{x}_q-\underline{x}_1\|_2 &= \|\underline{x}_q-\underline{\widehat{x}}\|_2+\|\underline{\widehat{x}}-\underline{x}_1\|_2 \\
&\geq \|\underline{x}_i-\underline{\widehat{x}}\|_2+\|\underline{\widehat{x}}-\underline{x}_1\|_2 \\
&\geq \|\underline{x}_i-\underline{x}_1\|_2.
\end{aligned} \tag{6}$$

Our choice of q does not allow either of these inequalities to be strict. The last one is satisfied as an equation if and only if $\underline{x}_i$ is at or beyond $\underline{\widehat{x}}$ on the straight line from $\underline{x}_1$ through $\underline{\widehat{x}}$, and by construction $\underline{x}_q$ has this property too. Further, the first inequality holds as an equation if and only if $\underline{x}_q$ and $\underline{x}_i$ are equidistant from $\underline{\widehat{x}}$. It follows from these remarks that $\underline{x}_i=\underline{x}_q$, which is the required contradiction.

In order to complete the proof, we move from $\underline{x}_q$ to $\underline{x}_1$ along the line segment that joins these points until we reach the boundary of T_q. Let $\underline{\widetilde{x}}$ be this point. It follows from this choice of $\underline{\widetilde{x}}$ that there exists an integer $j\neq q$ in $[1,n]$ such that the tiles T_j and T_q have a common edge of positive length and such that $\underline{\widetilde{x}}$ is on this edge. Thus, by the definition of S_j, we have $q \in S_j$. Further, because $\underline{\tilde{x}}$ is a point of T_j that is on the line segment from $\underline{x}_1$ to $\underline{x}_q$, the first part of the proof establishes that j is in the interval $[1, t]$. Therefore the lemma is true. □

Thus the following algorithm is suitable for determining the subsets, $V_i \subset V$, $i = 1, 2, \ldots, n$. The indices of the data points in each subset, V_i, are assembled in the array `INDEX`, which is maintained throughout as an ordered list of no more than $p + 1$ indices of data points sorted in ascending order of Euclidean distance from the data point $\underline{x}_i$. Also, the elements of the array `FLAG` are set to either one or zero to indicate whether or not the corresponding indices are currently elements of the array `INDEX`, and the integer `NVEC` is set to the number of indices which are currently elements of the array `INDEX`.

Step 1: Construct the Dirichlet tessellation of the set of n data points, V.

Step 2: Set `FLAG`$(i) := 0$, $i = 1, 2, \ldots, n$.

Step 3: For $i = 1, 2, \ldots, n$,

- **Step 4:** Set `INDEX`$(1) := i$, set `FLAG`$(i) := 1$, and set `NVEC` $:= 1$.
- **Step 5:** For $j = 1, 2, \ldots, p$,
  - **Step 6:** Set $k :=$ `INDEX`(j).
  - **Step 7:** For each element, ℓ, of the subset S_k for which `FLAG`$(\ell) = 0$,
    - **Step 8:** Set `FLAG`$(\ell) := 1$.
    - **Step 9:** Insert the index, ℓ, at the appropriate place in the array `INDEX`, maintaining the array `INDEX` as an ordered list of indices of data points sorted in ascending order of Euclidean distance from the data point $\underline{x}_i$.
    - **Step 10:** If `NVEC` $= p + 1$, then reset `FLAG(INDEX`$(p + 2)) := 0$. Otherwise, set `NVEC` $:=$ `NVEC` $+ 1$. Thus the array `INDEX` is maintained as an ordered list of no more than $p + 1$ indices of data points.
    - **Step 11:** Next ℓ.
  - **Step 12:** Next j.
- **Step 13:** Let $V_i = \left\{ \underline{x}_{\ell_j} : j = 1, 2, \ldots, p \right\}$, where $\ell_j =$ `INDEX`$(j + 1)$, $j = 1, 2, \ldots, p$.
- **Step 14:** Reset `FLAG(INDEX`$(j)) := 0$, $j = 1, 2, \ldots, p + 1$.
- **Step 15:** Next i.

The operation count to construct the Dirichlet tessellation is optimally $\mathcal{O}(n \log n)$, see, for example, Theorem 5.15 of Preparata and Shamos [5]; the computation of Step 2 of the algorithm can be included in this operation count.

Each element of each subset S_i, $i = 1, 2, \ldots, n$, represents an edge of the Dirichlet tessellation, and each edge of the Dirichlet tessellation corresponds to a unique edge in its straight-line dual, the Delaunay triangulation of the convex hull of the set of

n data points, V; see, for example, Sibson [6]. The Delaunay triangulation, being a triangulation, has, by Euler's formula, at most $3n - 6$ edges, and each edge of the Dirichlet tessellation is represented in exactly two of the subsets S_i. Thus, while a particular subset S_i may contain as many as $n - 1$ elements, on average the number of elements in each subset S_i, $i = 1, 2, \ldots, n$, does not exceed six. Therefore, we expect the operation count to determine each subset, $V_i \subset V$, $i = 1, 2, \ldots, n$, to be on average no worse than $\mathcal{O}(p^2)$; timings suggest the operation count to be between $\mathcal{O}(p)$ and $\mathcal{O}(p^2)$.

In Section 6.3.1 of Preparata and Shamos [5], the problem is solved using the p-th order Dirichlet tessellation of the set of n data points, V. This partitions the plane so that each region (or tile) of the partition is the locus of points which are closer to every member of a particular subset of p data points than to any other data point. The operation count to construct this higher order Dirichlet tessellation is $\mathcal{O}(p^2\, n\, \log n)$, and the subsequent operation count to determine each subset, $V_i \subset V$, $i = 1, 2, \ldots, n$, is $\mathcal{O}(p + \log n)$. Our algorithm, therefore, provides a considerable reduction in the amount of computation required.

Timings were obtained for the algorithm, run on a Sparc 10 workstation, where the data points were chosen randomly from the uniform distribution on the unit disc. Separate timings (in seconds) for the tasks of Steps 1–2 and Steps 3–15 of the algorithm are given in Table 1 for a range of values of n and p.

Acknowledgements

This research was undertaken whilst the author was a Research Associate in the Department of Applied Mathematics and Theoretical Physics at the University of Cambridge, under the direction of Professor M. J. D. Powell.

I wish to thank Professor Powell for his invaluable help and guidance, and also the Engineering and Physical Sciences Research Council for their financial support.

References

[1] Davis, P. J. "Interpolation and Approximation", Dover, New York, 1975.

[2] Green, P. J. and Sibson, R. "Computing Dirichlet tessellations in the plane", Comput. J., **21**, 168–173, 1978.

[3] Powell, M. J. D. Private Communication, 1996.

[4] Powell, M. J. D. "A new iterative algorithm for thin plate spline interpolation in two dimensions", Ann. Numer. Math., **4**, 519–527, 1997.

[5] Preparata, F. P. and Shamos, M. I. "Computational Geometry *An Introduction*", Springer-Verlag, New York, 1985.

[6] Sibson, R. "Locally equiangular triangulations", Comput. J., **21**, 243–245, 1978.

n	p	Steps 1–2		Steps 3–15		
		time	time/(n $\log n$)	time	time/np	time/np^2
10000	10	2.69	0.00002921	2.69	0.00002690	0.0000026900
10000	20	2.67	0.00002899	5.61	0.00002805	0.0000014025
10000	30	2.75	0.00002986	8.95	0.00002983	0.0000009944
10000	40	2.68	0.00002910	12.69	0.00003172	0.0000007931
10000	50	2.71	0.00002942	16.53	0.00003306	0.0000006612
20000	10	5.70	0.00002878	6.58	0.00003290	0.0000032900
20000	20	5.74	0.00002898	13.53	0.00003382	0.0000016912
20000	30	6.03	0.00003044	20.96	0.00003493	0.0000011644
20000	40	5.73	0.00002893	28.68	0.00003585	0.0000008962
20000	50	5.68	0.00002868	37.01	0.00003701	0.0000007402
50000	10	16.08	0.00002972	19.32	0.00003864	0.0000038640
50000	20	16.07	0.00002970	38.68	0.00003868	0.0000019340
50000	30	15.98	0.00002954	59.76	0.00003984	0.0000013280
50000	40	15.98	0.00002954	81.84	0.00004092	0.0000010230
50000	50	16.55	0.00003059	104.51	0.00004180	0.0000008361
100000	10	35.26	0.00003063	40.90	0.00004090	0.0000040900
100000	20	35.18	0.00003056	82.46	0.00004123	0.0000020615
100000	30	34.86	0.00003028	125.97	0.00004199	0.0000013997
100000	40	35.02	0.00003042	172.44	0.00004311	0.0000010777
100000	50	36.79	0.00003196	220.43	0.00004409	0.0000008817

Table 1: Timings for the algorithm