

Junioraufgabe 1: Zum Winde verweht

J1.1 Lösungsidee

Hinter dieser Aufgabe verbirgt sich ein geometrisches Problem. Es ist hilfreich, die Situation aus Häusern und Windrädern in einer flachen Landschaft zunächst mathematisch zu erfassen. Hierzu betrachten wir die Häuser und Windräder als Punkte in einem zweidimensionalen Koordinatensystem. Mit H_1, \dots, H_n bezeichnen wir die Punkte der Häuser, mit W_1, \dots, W_m jene der Windräder. Dabei sind die Häuser von 1 bis n und die Windräder von 1 bis m durchnummierter.

Wir betrachten nun jedes Windrad separat und bestimmen seine maximale Höhe. Es bezeichne $\mathcal{H}(k)$ die maximale Höhe von Windrad k . Um $\mathcal{H}(k)$ zu bestimmen, reizen wir die 10H-Regel so weit es geht aus: Wir bauen Windrad k so hoch, dass der Mindestabstand $10 \cdot \mathcal{H}(k)$ zu allen Häusern gerade so eingehalten wird.

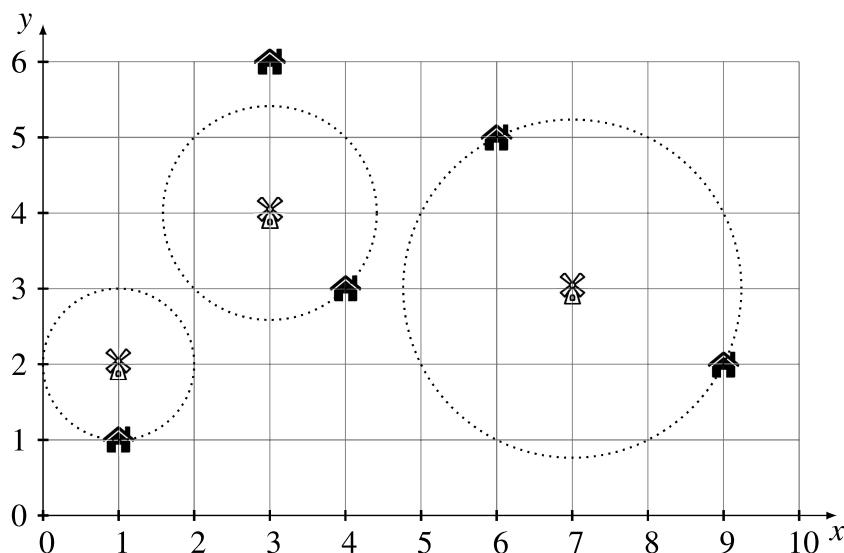


Abbildung J1.1: Beispiel mit $n = 5$ Häusern und $m = 3$ Windrädern

Geometrisch ist die Situation in Abbildung J1.1 dargestellt: Wir ziehen um jedes Windrad einen Kreis mit größtmöglichen Radius, sodass keine Häuser strikt innerhalb des Kreises liegen. Erhält ein Windrad k dann den Radius r , so muss laut 10H-Regel $r \geq 10 \cdot \mathcal{H}(k)$ sein, damit jedes Haus Abstand mindestens $r \geq 10 \cdot \mathcal{H}(k)$ zum Windrad hat. Wegen

$$r \geq 10 \cdot \mathcal{H}(k) \iff \mathcal{H}(k) \leq \frac{r}{10}$$

darf das Windrad maximal Höhe $\frac{r}{10}$ haben.

Damit können wir bereits die Aufgabe lösen: Für jedes Windrad berechnen wir den größtmöglichen Radius r , sodass kein Haus strikt innerhalb des Radius r um das Windrad liegt. Die maximale Höhe des Windrads ist dann $\frac{r}{10}$. Um r zu berechnen, bemerken wir, dass r der Abstand zum nächstgelegenen Haus ist. Wir können also alle Häuser durchgehen, jeweils den Abstand bestimmen und uns den (bisher) kleinsten Abstand merken.

Abstandsberechnung

Es bleibt eine Frage zu klären: Wie berechnen wir den Abstand zwischen einem Windrad k und einem Haus ℓ , also zwischen den Punkten W_k und H_ℓ ?

Bemerkung: Diese Frage ist im Rahmen der Lösungsidee zu beantworten. Eine Formel für den Abstand darf ohne Herleitung verwendet bzw. als bekannt vorausgesetzt werden, da oft aus dem Schulunterricht bekannt. Für Interessierte beschreiben wir folgend trotzdem, wie man auf die häufig verwendete und nützliche Formel kommt.

Sind $W_k = (x_k^{(W)}, y_k^{(W)})$ und $H_\ell = (x_\ell^{(H)}, y_\ell^{(H)})$, so können wir den Abstand $d(W_k, H_\ell)$ von W_k zu H_ℓ nach dem Satz des Pythagoras durch

$$d(W_k, H_\ell) = \sqrt{(x_k^{(W)} - x_\ell^{(H)})^2 + (y_k^{(W)} - y_\ell^{(H)})^2}$$

berechnen. Die Notation mag etwas sperrig erscheinen, doch der Abstandsformel liegt eine einfache Idee zugrunde: Wir erhalten das in Abbildung J1.2 dargestellte rechtwinklige Dreieck, indem wir den direkten Weg von W_k und H_ℓ (blau) aufteilen in eine x -Komponente und eine y -Komponente (jeweils orange) – die Katheten des Dreiecks.

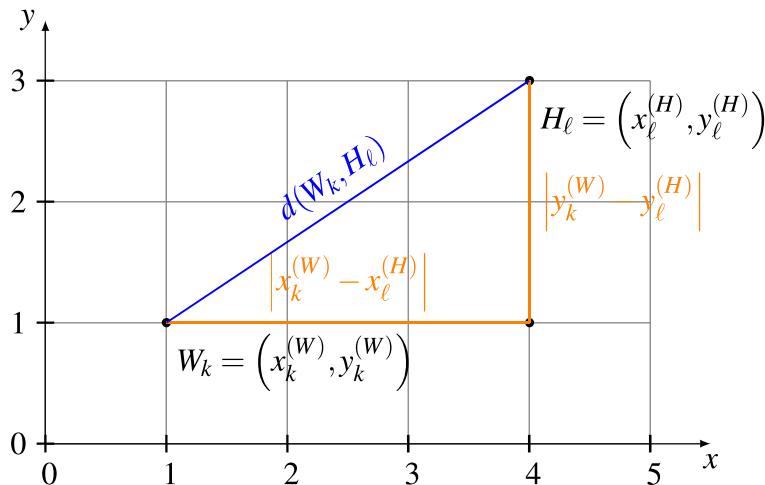


Abbildung J1.2: Abstandsberechnung mit dem Satz des Pythagoras

Die Längen der Katheten sind $|x_k^{(W)} - x_\ell^{(H)}|$ und $|y_k^{(W)} - y_\ell^{(H)}|$, wobei wir die Beträge bilden müssen, um in allen vier Situationen „ W_k links/rechts von H_ℓ “ und „ W_k über/unter H_ℓ “ den Abstand ohne negatives Vorzeichen zu erhalten. Da das Dreieck rechtwinklig ist, gilt nach Pythagoras

$$d(W_k, H_\ell)^2 = |x_k^{(W)} - x_\ell^{(H)}|^2 + |y_k^{(W)} - y_\ell^{(H)}|^2 = (x_k^{(W)} - x_\ell^{(H)})^2 + (y_k^{(W)} - y_\ell^{(H)})^2$$

und Wurzelziehen liefert die oben genannte Formel.

Algorithmus in Pseudocode

Es ist bei informatischen Problemen oft hilfreich, zwischen Lösungsidee und Programmcode eine Abstraktionsebene einzuschieben, den *Pseudocode*. Mittels Pseudocode können wir den

Algorithmus sehr präzise beschrieben, ohne uns gleich mit den Details einer Programmiersprache herumzuschlagen. Siehe Algorithmus 1 für eine Formulierung der oben beschriebenen Lösungsidee in Pseudocode.

Algorithmus 1 Bestimmung der maximalen Höhe aller Windräder

```

1: for  $k = 1$  to  $m$  do
2:    $r \leftarrow \infty$                                  $\triangleright$  größtmöglicher Radius um Windrad  $k$ 
3:   for  $\ell = 1$  to  $n$  do
4:      $r \leftarrow \min(r, d(W_k, H_\ell))$            $\triangleright d(W_k, H_\ell)$  via Abstandsformel
5:   end for
6:    $\mathcal{H}(k) \leftarrow \frac{r}{10}$ 
7: end for
8: return max. Höhe  $\mathcal{H}(k)$  für alle Windräder  $k = 1, \dots, m$ 
  
```

Ausgehend von diesem Pseudocode kann der Algorithmus sehr systematisch in einer (imperativen) Programmiersprache implementiert werden. Für die Instruktion $r \leftarrow \infty$ setzt man r in der Praxis oft auf einen ausreichend hohen Wert.

Laufzeitbetrachtung

Für fehlende Laufzeitbetrachtungen gibt es bei dieser Aufgabe keinen Punktabzug. Es gehört jedoch zum Algorithmendesign dazu, die Performance auf großen Eingaben mathematisch zu untersuchen. In der 1. Runde werden solche Analysen in der Regel nicht erwartet, aber in der 2. Runde.

Der Pseudocode bietet einen idealen Ausgangspunkt für unsere Laufzeitanalyse. In Abhängigkeit von der Anzahl Häuser n und der Anzahl Windräder m gilt es zu ermitteln, wie viel Zeit die Ausführung des Programms brauchen wird.

Für die Zeilen 2, 4, 6 in Algorithmus 1 werden jeweils konstant viele Rechenoperationen benötigt. Die Laufzeit liegt mithin in $\mathcal{O}(1)$ (O-Notation¹). Insgesamt werden die Instruktionen in Zeile 2 und Zeile 6 jeweils m -mal ausgeführt, während die Instruktion in Zeile 4 wegen der verschachtelten **for**-Schleifen $m \cdot n$ -mal ausgeführt wird. Die Gesamlaufzeit liegt somit in

$$m \cdot \mathcal{O}(1) + m \cdot \mathcal{O}(1) + m \cdot n \cdot \mathcal{O}(1) = \mathcal{O}(m \cdot n),$$

ist also proportional zum Produkt $m \cdot n$.

Ein konkretes Beispiel: Es ist zu erwarten, dass unser Programm bei 300 Häusern und 20 Windkraftanlagen etwa 12-mal so viel Rechenzeit benötigt wie bei 100 Häusern und 5 Windkraftanlagen, denn $300 \cdot 20 = 6000$ und $100 \cdot 5 = 500$.

J1.2 Effizientere Lösung

Wir haben eine einfache $\mathcal{O}(mn)$ -Lösung gesehen, die jedoch dann an ihre Grenzen stößt, wenn sowohl n als auch m beide groß sind, z. B. $n = m = 100000$ bei heutigen Computern. Somit

¹Siehe https://en.wikipedia.org/wiki/Big_O_notation. Die O-Notation ist für Laufzeitangaben zu empfehlen. Jedoch genügt eine präzise Beschreibung des Laufzeitverhaltens in Textform in der Regel auch den BWINF-Anforderungen.

stellt sich die Frage nach einem schnelleren Algorithmus, den wir hier skizzieren wollen. Die Laufzeit liegt in $\mathcal{O}((m+n) \cdot \log(m+n))$, d. h. sie ist nur geringfügig schlechter als linear.

Die hier beschriebene Lösung enthält einige interessante, aber auch sehr fortgeschrittene Konzepte und Ansätze. *Sie geht dabei weit über die Anforderungen einer Junioraufgabe und auch einer Erstrundenaufgabe hinaus.*

Voronoi-Diagramm

Wir berechnen zunächst das Voronoi-Diagramm² der Häuser. Dafür existieren verschiedene Algorithmen mit Laufzeit in $\mathcal{O}(n \log n)$, u. a. Fortunes Algorithmus. Das Voronoi-Diagramm unterteilt die Ebene in n polygonale Regionen bzw. „Einzugsgebiete“, die jeweils genau die Punkte enthalten, deren nächstes Haus das Haus in der entsprechenden Region (das sogenannte *Zentrum*) ist, siehe Abbildung J1.3.

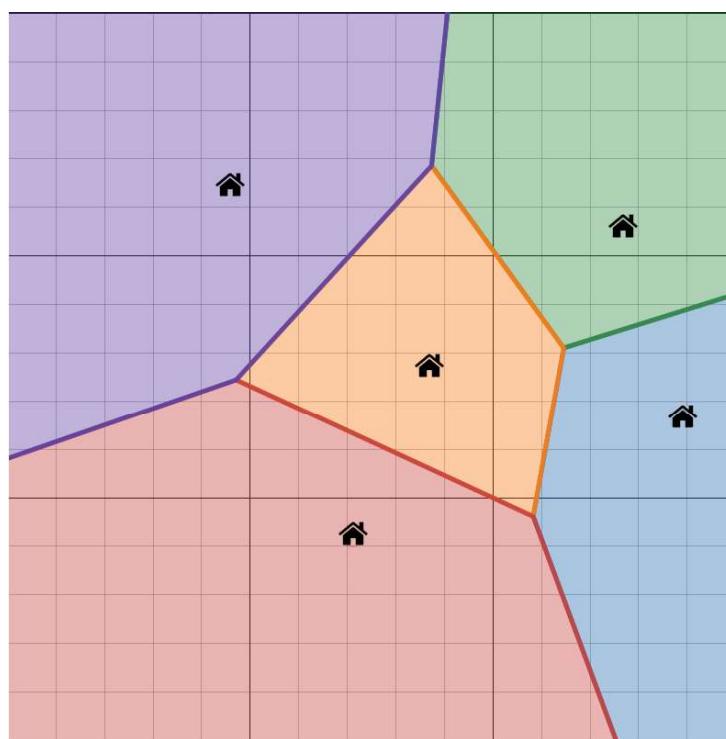


Abbildung J1.3: Beispiel für ein Voronoi-Diagramm der Häuser

Nearest-Neighbor-Abfragen

Die Frage nach dem jeweils nächstgelegenen Haus für alle Windräder ist als *Nearest-Neighbor-Problem* in der Literatur bekannt. Mithilfe des Voronoi-Diagramms können wir für jedes Windrad das nächstgelegene Haus ermitteln, indem wir die Voronoi-Region des Windrads ermitteln. Per Definition ist das Haus im Zentrum der Region das gesuchte Haus.

Wie finden wir zu einem gegebenen Punkt die Voronoi-Region? Die Idee ist ein Sweep-Line-Algorithmus, der die Ecken des Voronoi-Diagramms und die Windräder in einer bestimmten

²Jacques Heunis hat in diesem Blog-Post eine sehr schöne Erklärung zu Voronoi-Diagrammen und Fortunes Algorithmus veröffentlicht: <https://jacquesheunis.com/post/fortunes-algorithm/>.

Richtung durchläuft und dabei die folgenden Berechnungen ausführt. Wir stellen uns dabei eine vertikale Gerade (die *Sweep Line*) vor, die sich in horizontaler Richtung bewegt.³

- Ecken des Voronoi-Diagramms: Für jede Ecke gehen wir alle angrenzenden Kanten durch. Wir speichern uns die Kanten rechts von der Sweep Line in einer Datenstruktur \mathcal{D} sortiert nach y-Koordinaten. Mit y-Koordinaten sind hier die y-Koordinaten der Schnittpunkte der sich gerade bewegenden Sweepline mit den Kanten in \mathcal{D} gemeint. Kanten links von der Sweep Line werden entsprechend aus \mathcal{D} gelöscht.
- Windräder: Da \mathcal{D} sortiert ist, können wir mit einer binären Suche die zwei Kanten in \mathcal{D} finden, die am nächsten über bzw. unter dem Punkt des Windrads liegen. Diese zwei Kanten grenzen an genau eine gemeinsame Voronoi-Region: Die Region, in der sich das Windrad befindet.

Umsetzung und Laufzeit: Insgesamt wird die Sweep Line $\mathcal{O}(n + m)$ -mal bewegt, man nennt die „Zwischenstopps“ auch *Events*. Hier geht ein, dass das Voronoi-Diagramm maximal $2n - 5$ Ecken haben kann. Initial müssen die Events nach unserer Bewegungsrichtung sortiert werden, was Zeit $\mathcal{O}((n + m) \log(n + m))$ dauert. Die Binärsuche im Falle eines Windrads dauert Zeit $\mathcal{O}(\log n)$, da auch die Anzahl der Kanten des Voronoi-Diagramms linear bzw. genauer durch $3n - 6$ beschränkt ist. Schließlich kann auch das Einfügen und Löschen in bzw. aus \mathcal{D} in Zeit $\mathcal{O}(\log n)$ bewerkstelligt werden, wenn man für \mathcal{D} die richtige Datenstruktur wählt, z. B. einen balancierten binären Suchbaum⁴. Zusammengefasst wird man dann $\mathcal{O}(n + m)$ Events haben, die jeweils Zeit $\mathcal{O}(\log n)$ brauchen, sodass die Gesamlaufzeit vom Sortieren der Events in Zeit $\mathcal{O}((n + m) \log(n + m))$ dominiert wird.

Bei der Implementierung wird man wahrscheinlich auf weitere Schwierigkeiten und Sonderfälle stoßen, etwa die unbegrenzten Regionen. Wir haben mit der oben erwähnten „Bewegungsrichtung“ versucht, ein wenig zu veranschaulichen, wie tückisch solche Sonderfälle sind und wie sie behandelt werden können. Diese Sonderfälle treten häufiger in der algorithmischen Geometrie auf und es gibt für deren Lösung daher immerhin etablierte Ansätze.

Bemerkung: Es ist auch möglich, die Nearest-Neighbor-Abfragen schon während der Konstruktion des Voronoi-Diagramms mit Fortunes Algorithmus zu beantworten. Hierzu sei auf die Publikation von Dinis und Mamede⁵ verwiesen.

J1.3 Beispiele

In Tabelle 2 sind die Ergebnisse für alle vier BWINF-Beispiele dargestellt. Am wichtigsten ist jeweils die Angabe der maximalen Höhen $\mathcal{H}(1), \dots, \mathcal{H}(m)$ aller Windräder. Die Werte sind auf zwei Nachkommastellen gerundet.

In der letzten Spalte kann jeweils die benötigte Rechenzeit des Python-Programms auf einem halbwegs modernen Linux-Laptop abgelesen werden. Der Leser sei dazu angehalten, den etwa proportionalen Zusammenhang zwischen $n \cdot m$ und der Rechenzeit zu überprüfen.

³Um auch vertikale Kanten des Voronoi-Diagramms korrekt bearbeiten zu können, wählen wir die Richtung ggf. (leicht) anders.

⁴Ein Kandidat wäre der AVL-Baum: <https://de.wikipedia.org/wiki/AVL-Baum> In vielen Programmiersprachen ist eine passende Datenstruktur schon integriert, beispielsweise in Python, C++ oder Java unter der Bezeichnung `set` bzw. `TreeSet`.

⁵https://www.researchgate.net/publication/2895260_A_Sweep_Line_Algorithm_for_Nearest_Neighbour_Queries

#	Häuser n	Windräder m	Max. Höhen $\mathcal{H}(1), \dots, \mathcal{H}(m)$	Rechenzeit
1	12	3	48.52 m, 158.98 m, 72.41 m	< 1 ms
2	94	15	115.16 m, 201.25 m, 138.85 m, 209.12 m, 132.01 m, 186.16 m, 161.68 m, 133.30 m, 133.54 m, 128.77 m, 91.78 m, 118.28 m, 161.95 m, 142.39 m, 177.04 m	2 ms
3	2382	16	451.57 m, 393.79 m, 336.70 m, 280.74 m, 444.62 m, 385.71 m, 327.11 m, 269.02 m, 440.84 m, 381.25 m, 321.73 m, 262.32 m, 440.31 m, 380.54 m, 320.78 m, 261.02 m	34 ms
4	9993	30	0.00 m, 383.81 m, 262.45 m, 233.99 m, 296.19 m, 71.76 m, 181.41 m, 235.40 m, 343.11 m, 177.90 m, 449.16 m, 408.03 m, 317.95 m, 221.29 m, 520.12 m, 394.71 m, 433.25 m, 703.83 m, 168.42 m, 201.27 m, 139.16 m, 348.99 m, 297.91 m, 110.23 m, 813.60 m, 236.19 m, 391.94 m, 125.78 m, 241.01 m, 625.40 m	246 ms

Tabelle 1: Ausgaben unseres Programms für die BWINF-Beispieleingaben

#	Häuser n	Windräder m	Max. Höhen $\mathcal{H}(1), \dots, \mathcal{H}(m)$
4 (alt)	114993	30	0.0 m, 179.29 m, 107.89 m, 151.18 m, 115.28 m, 71.76 m, 42.22 m, 58.58 m, 67.58 m, 112.13 m, 63.16 m, 251.61 m, 155.14 m, 118.78 m, 336.01 m, 109.77 m, 260.79 m, 350.22 m, 135.99 m, 68.08 m, 139.16 m, 50.91 m, 72.25 m, 110.23 m, 532.78 m, 94.06 m, 209.56 m, 28.15 m, 76.85 m, 425.83 m

Tabelle 2: Ausgaben unseres Programms für die alte Version der 4. BWINF-Beispieleingabe

Bei dieser Aufgabe bietet sich eine Visualisierung im Stile von Abbildung J1.1 an. Daran lässt sich die Korrektheit der berechneten Radien geometrisch prüfen. Siehe Abbildung J1.4 bis J1.7 für die Visualisierungen zu `landkreis1.txt` bis `landkreis4.txt`. Man beachte, dass in Abbildung J1.6 und J1.7 die Häuser durch orange Punkte dargestellt sind – mehrere Tausend kleine Icons zu rendern ist eben auch nicht immer geschenkt...

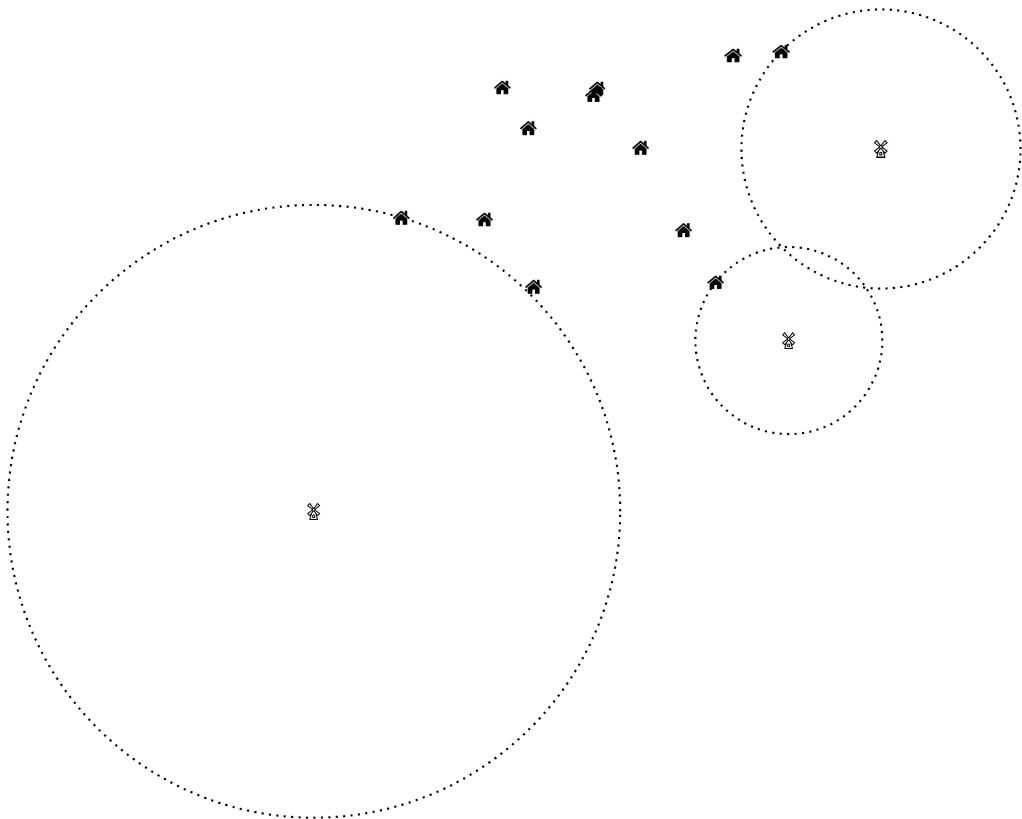


Abbildung J1.4: Visualisierung der Ergebnisse für Landkreis 1

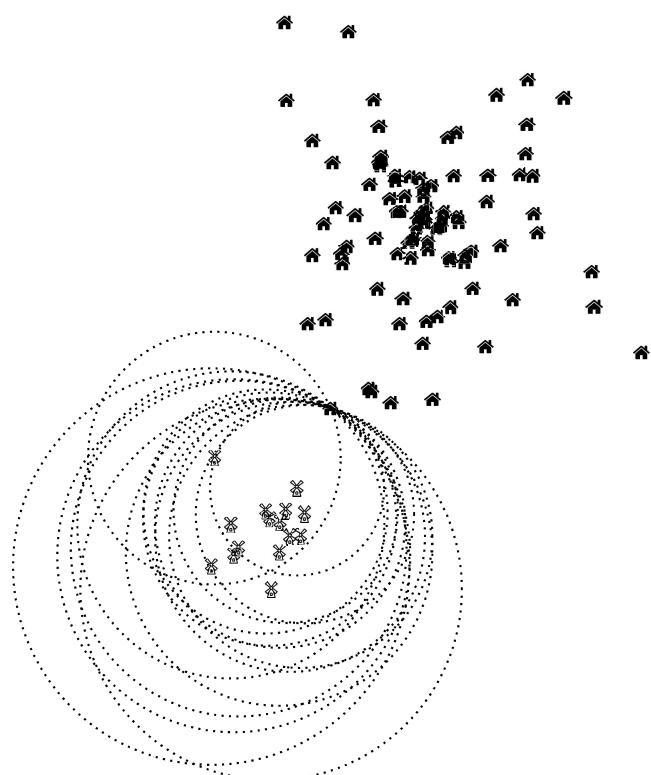


Abbildung J1.5: Visualisierung der Ergebnisse für Landkreis 2

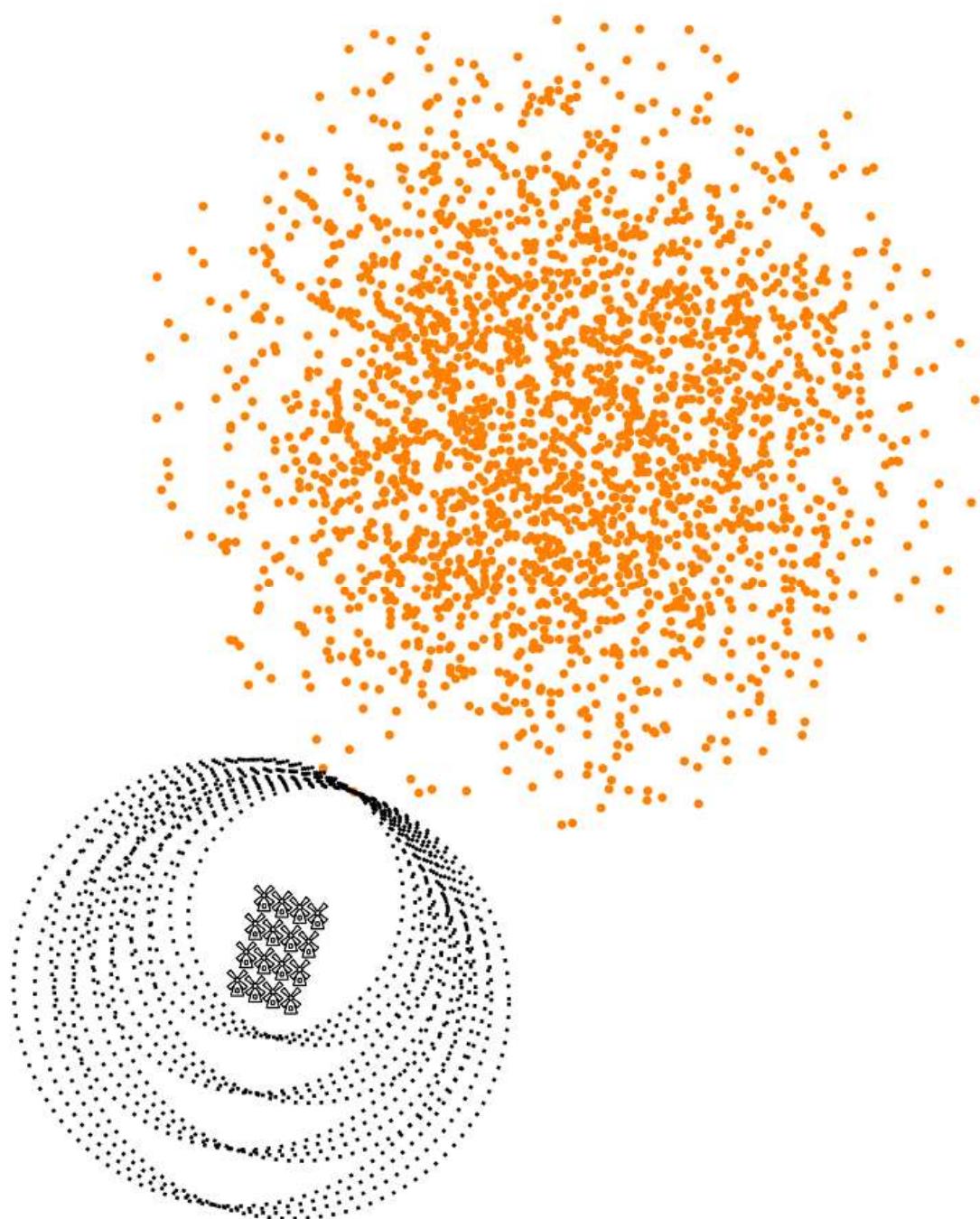


Abbildung J1.6: Visualisierung der Ergebnisse für Landkreis 3

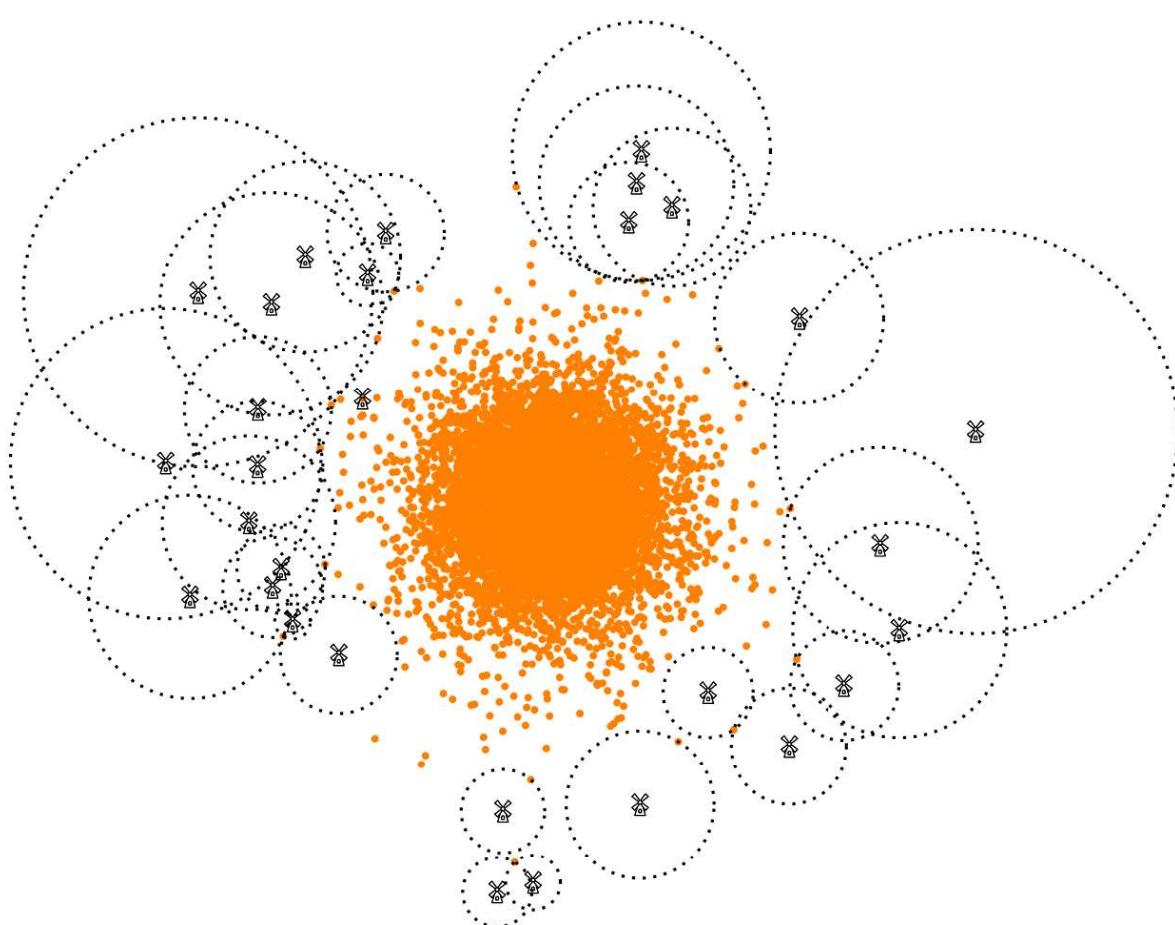


Abbildung J1.7: Visualisierung der Ergebnisse für Landkreis 4

J1.4 Bewertungskriterien

Die Bewertungskriterien vom Bewertungsbogen werden hier erläutert (Punktabzug in []).

- **[−1] Lösungsverfahren fehlerhaft**

Das Lösungsverfahren muss die maximalen Höhen für jedes Windrad korrekt berechnen. Den Radius mit 10 zu multiplizieren statt zu dividieren, Abstände falsch zu berechnen usw. führen sämtlich zu Punktabzug.

- **[−1] Abstands- und Distanzberechnung nicht erklärt**

Es ist zu erklären, wie die maximalen Höhen berechnet werden. Eine Lösung für die Abstandsberechnung ist zumindest zu erwähnen, gleichwohl eine Formel ohne weitere Erklärungen verwendet werden darf.

- **[−1] Ergebnisse schlecht nachvollziehbar**

Für jedes Beispiel sind die maximalen Höhen aller Windräder abzudrucken, sodass eine Zuordnung anhand der Nummerierung schnell möglich ist. Eine Angabe der Koordinaten ist nicht erforderlich. Werden die maximalen Höhen zu Koordinaten statt Nummern angegeben, ist das akzeptabel.

- **[−1] Beispiele fehlerhaft bzw. zu wenige oder ungeeignete Beispiele**

Die Dokumentation soll Ergebnisse zu mindestens 3 der vorgegebenen Beispiele (landkreis1.txt bis landkreis4.txt) enthalten. Dabei wird akzeptiert, wenn die Höhen auf ganze Zahlen gerundet werden.