Graph-based time–space trade-offs for approximate near neighbors

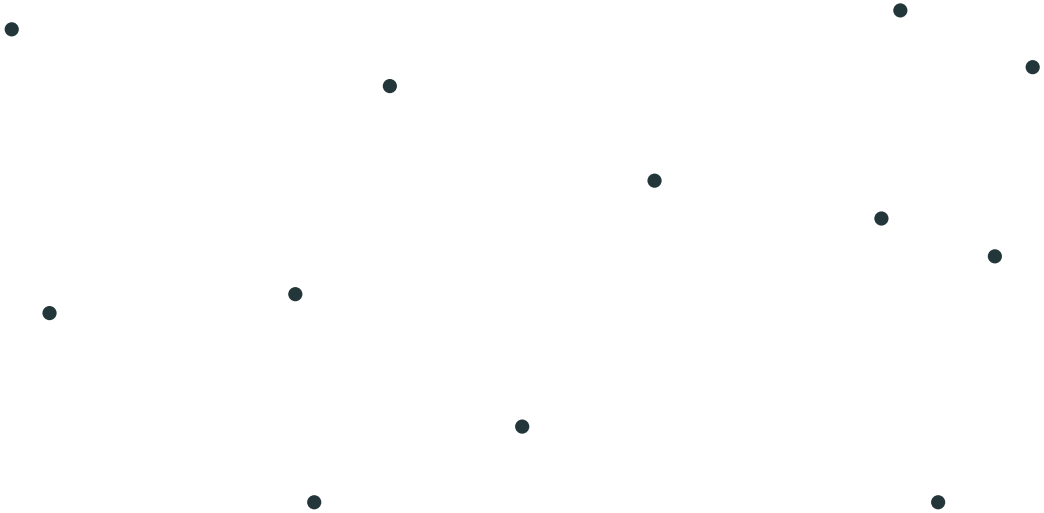
Thijs Laarhoven

mail@thijs.com
http://thijs.com/

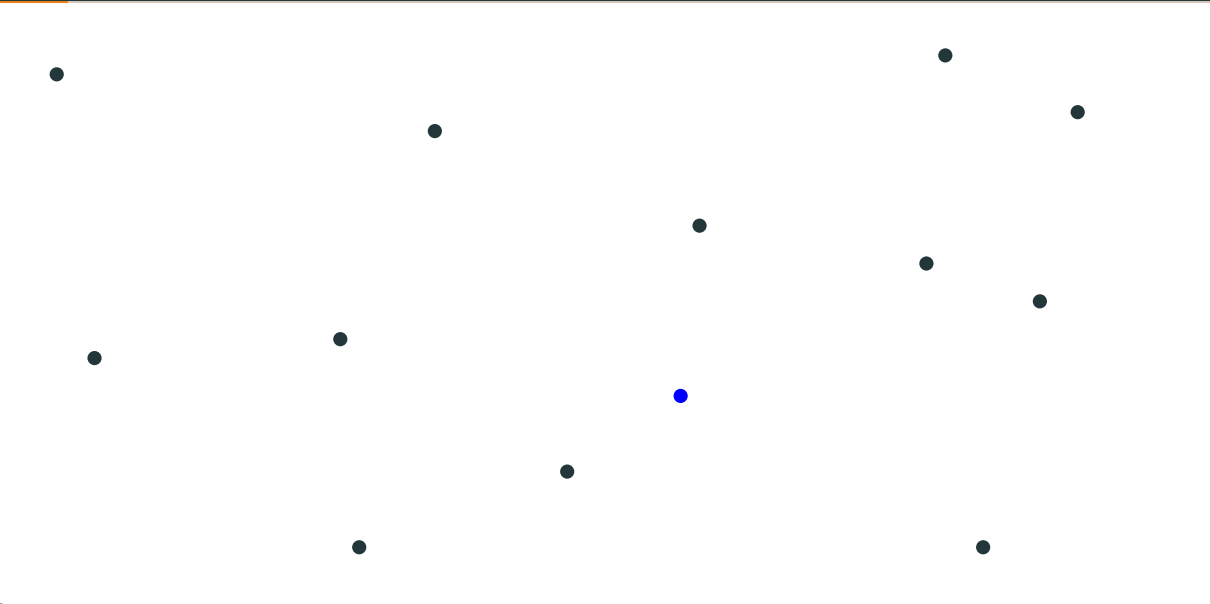
SoCG 2018, Budapest, Hungary
(June 13, 2018)

Nearest neighbor searching

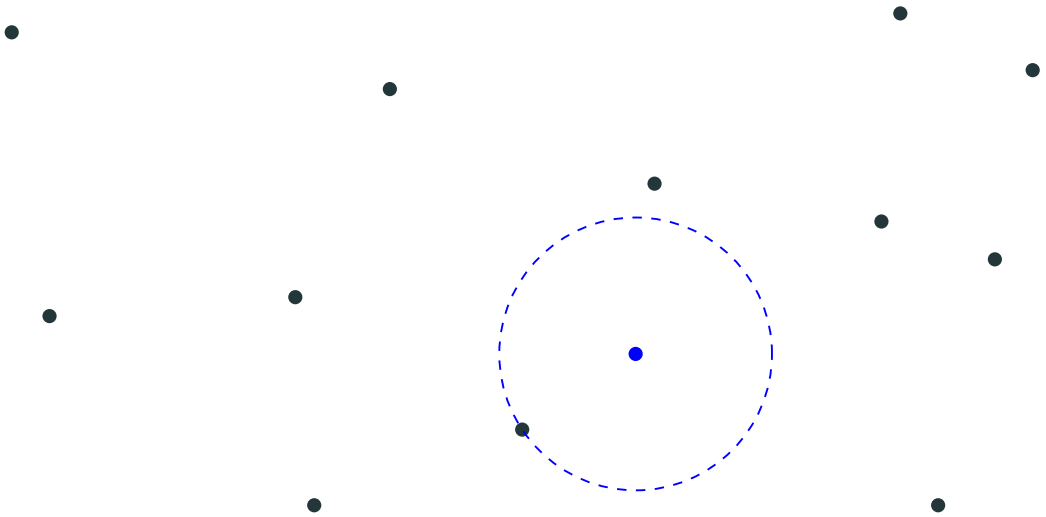
Nearest neighbor problem – Problem description



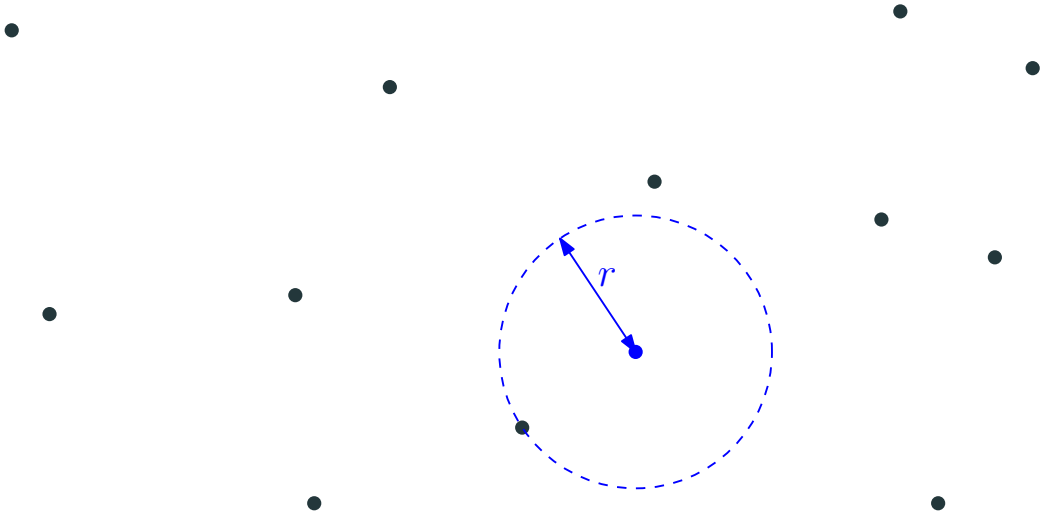
Nearest neighbor problem – Problem description



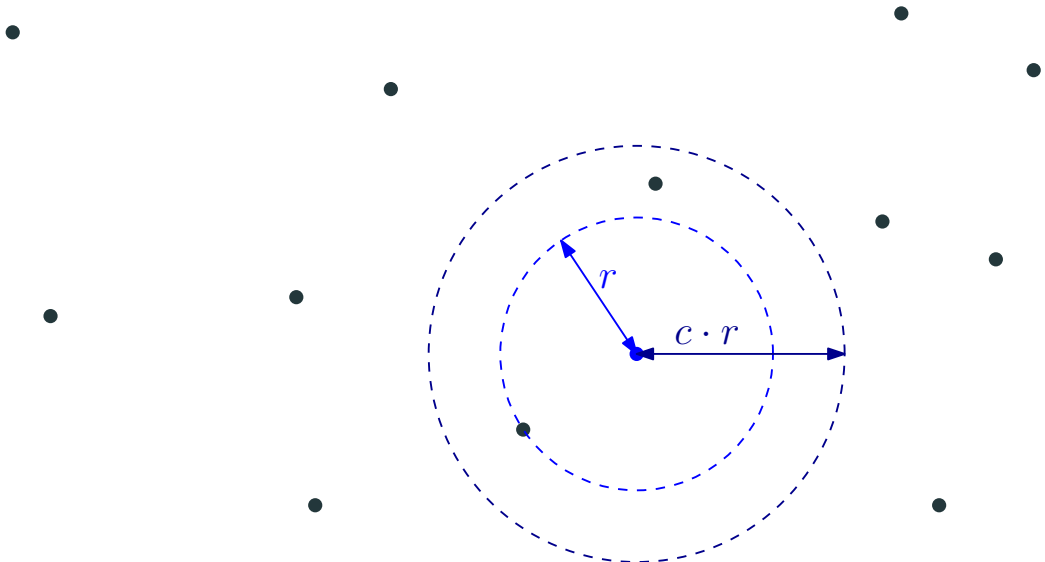
Nearest neighbor problem – Problem description



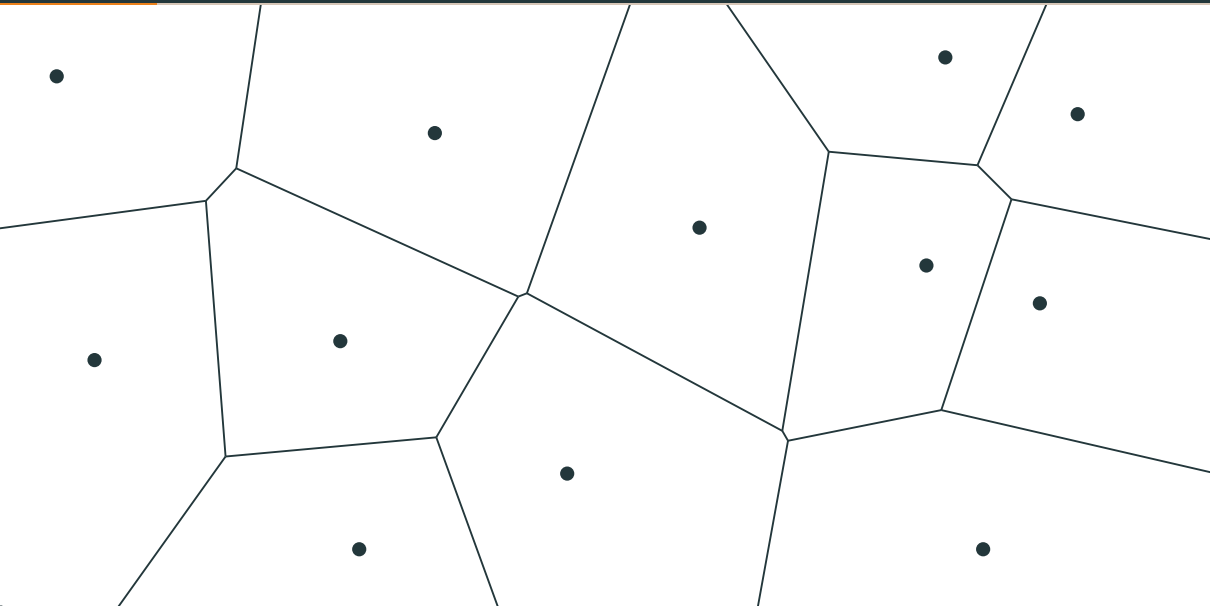
Nearest neighbor problem – Approximate solutions



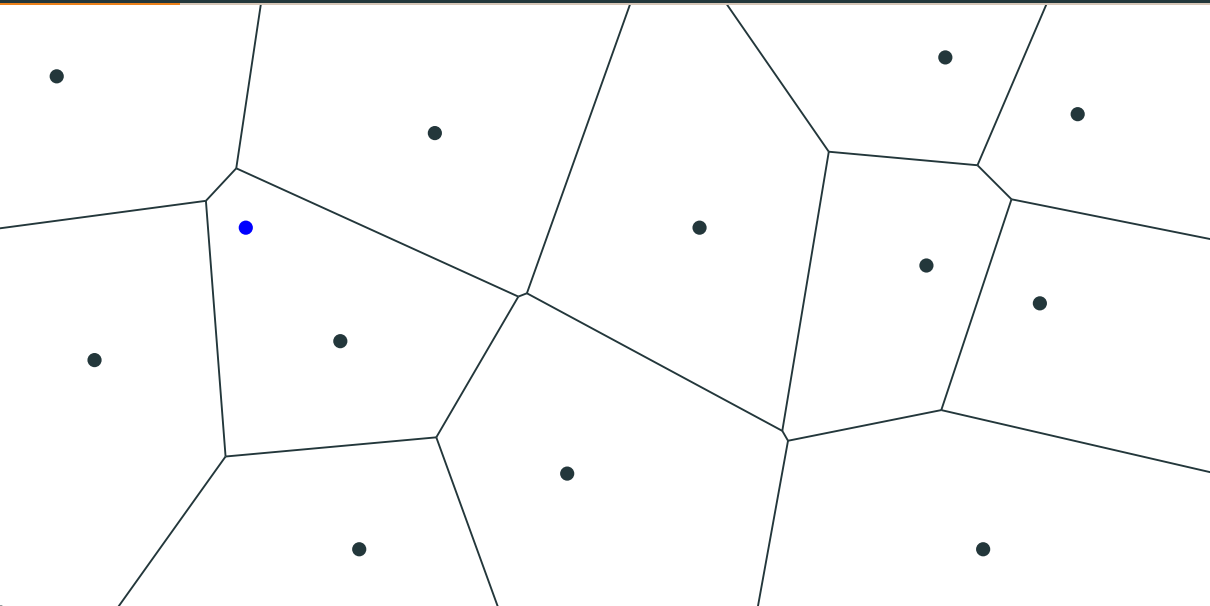
Nearest neighbor problem – Approximate solutions



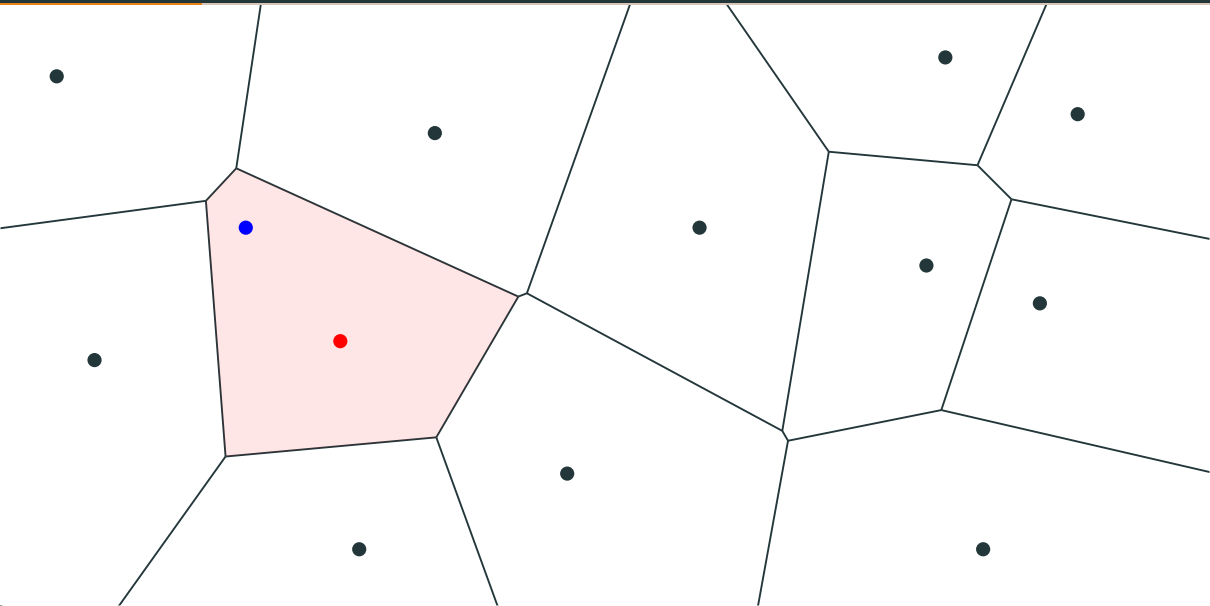
Nearest neighbor problem – Example: Voronoi cells



Nearest neighbor problem – Example: Voronoi cells

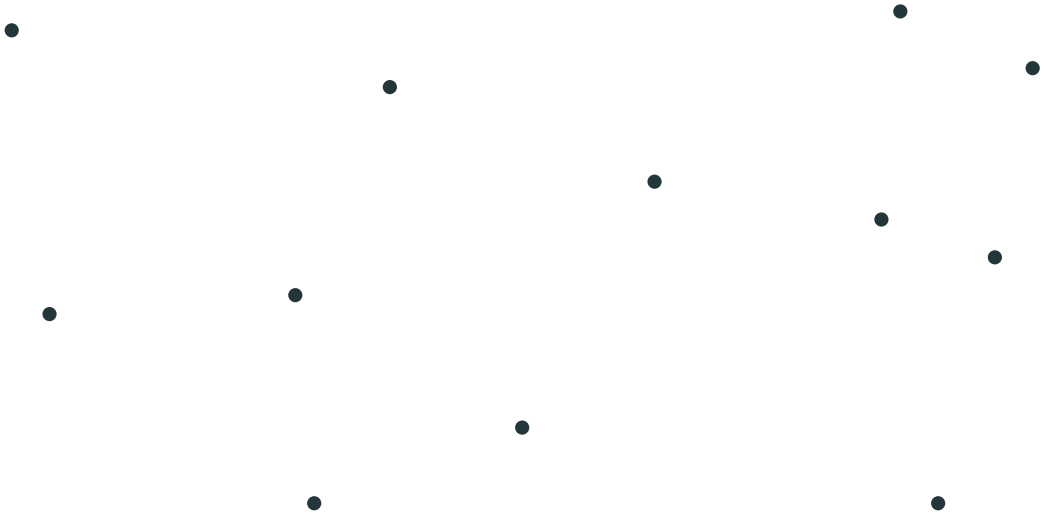


Nearest neighbor problem – Example: Voronoi cells

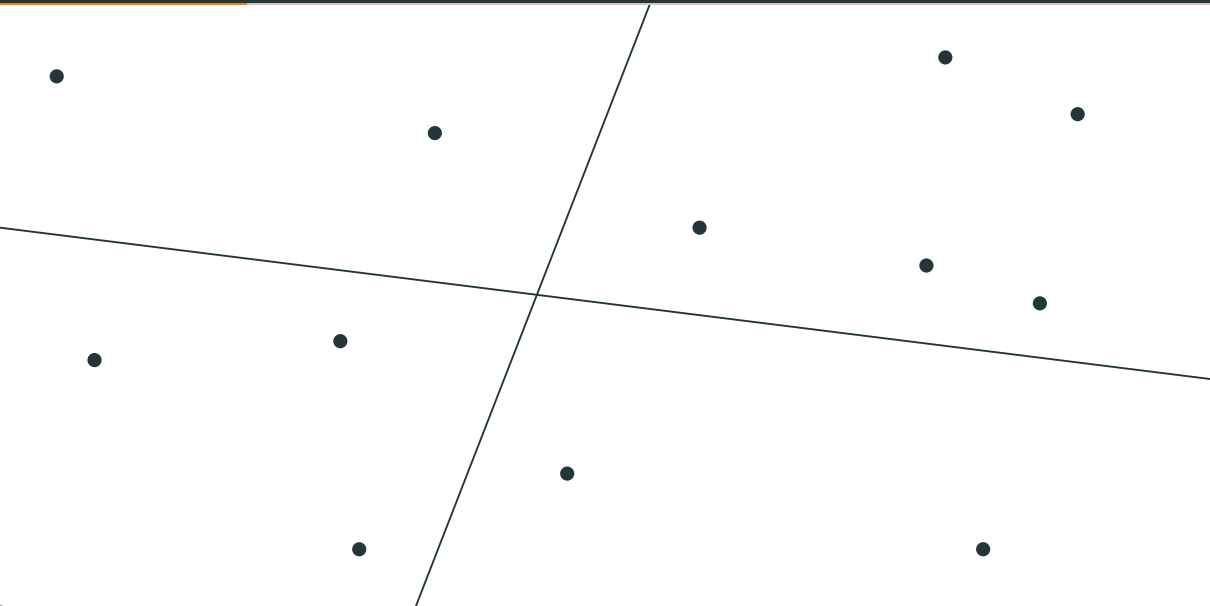


Partition-based methods

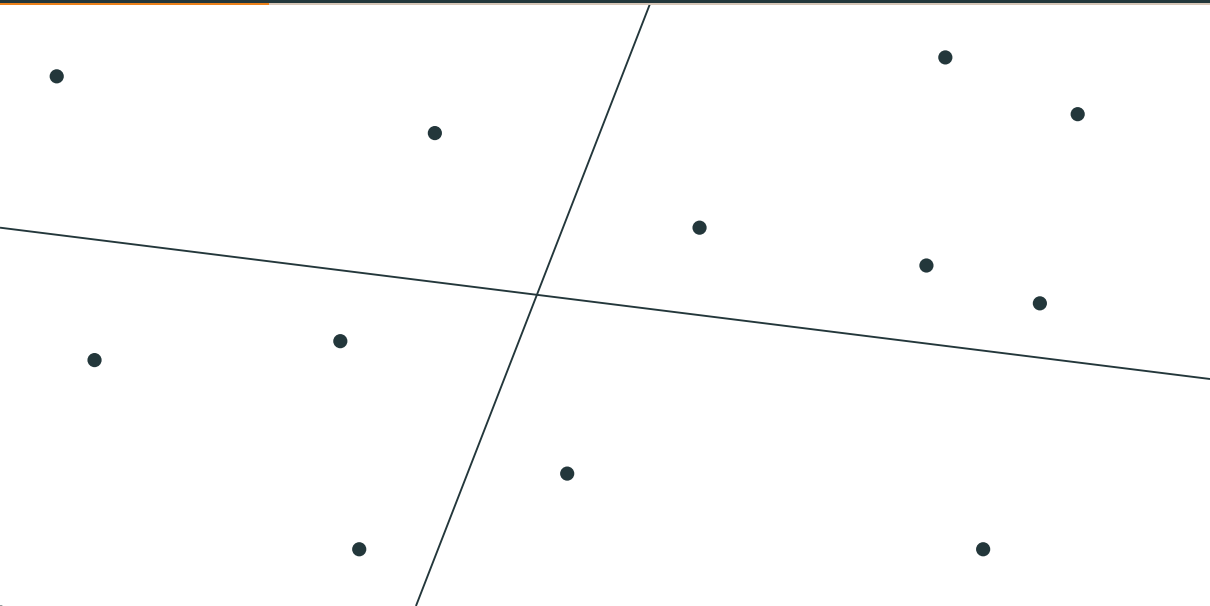
Partition-based methods – Data structure



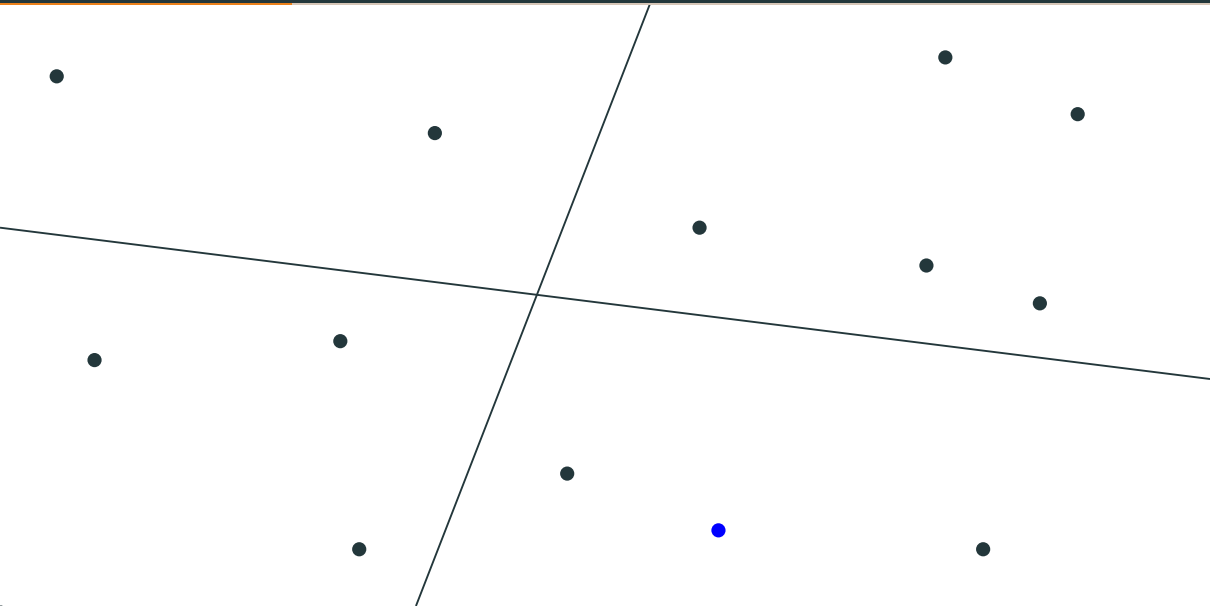
Partition-based methods – Data structure



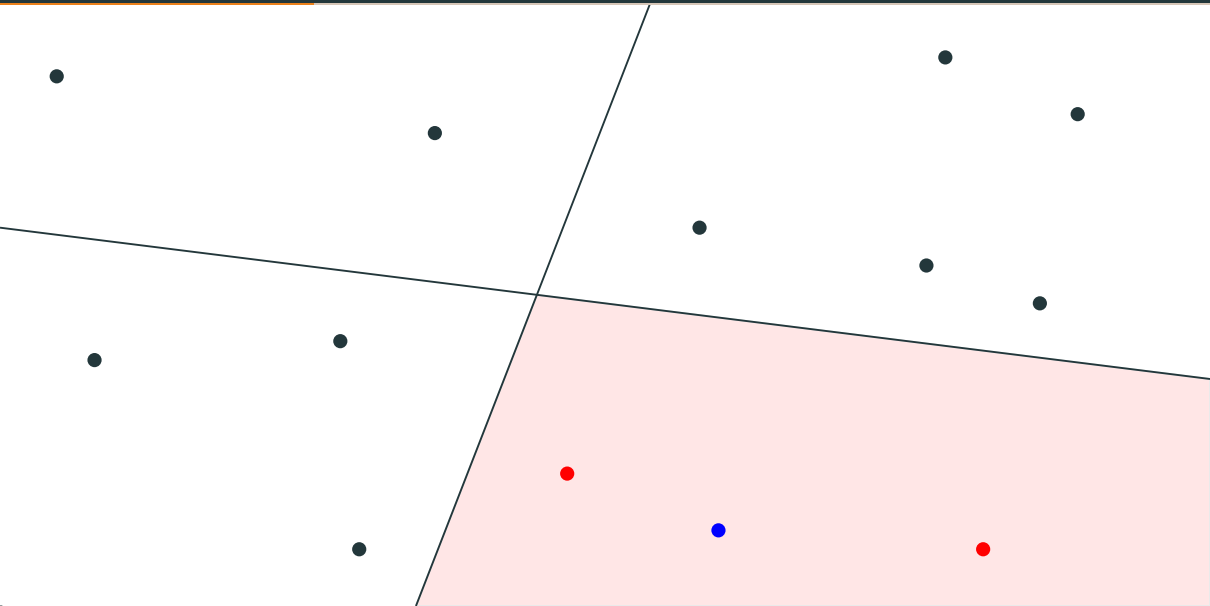
Partition-based methods – Hash table lookups



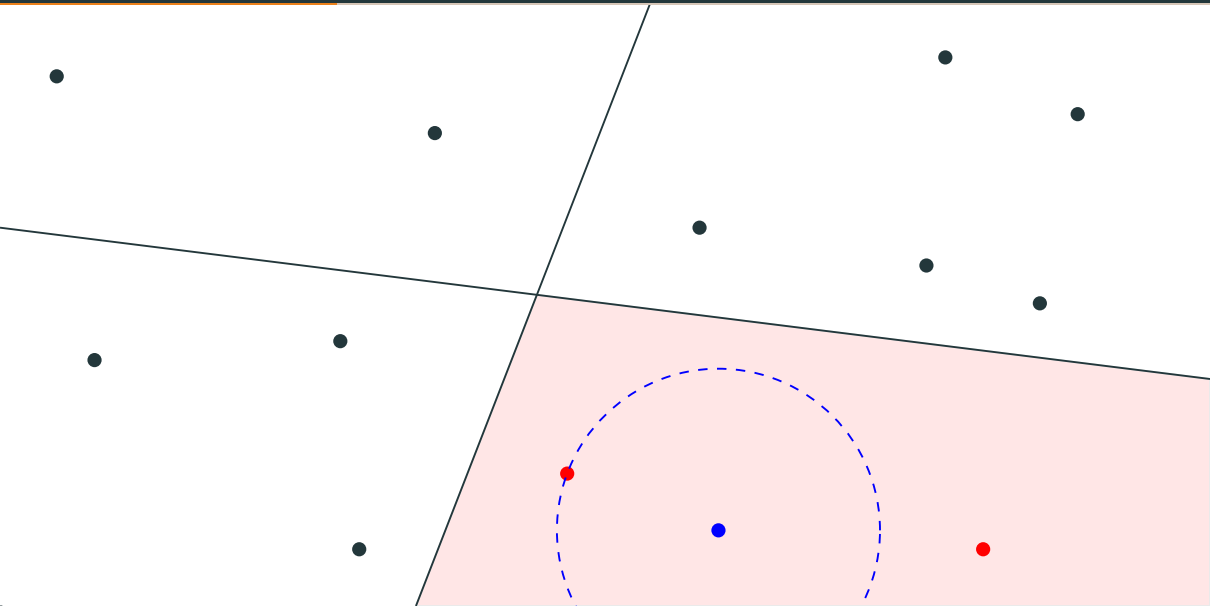
Partition-based methods – Hash table lookups



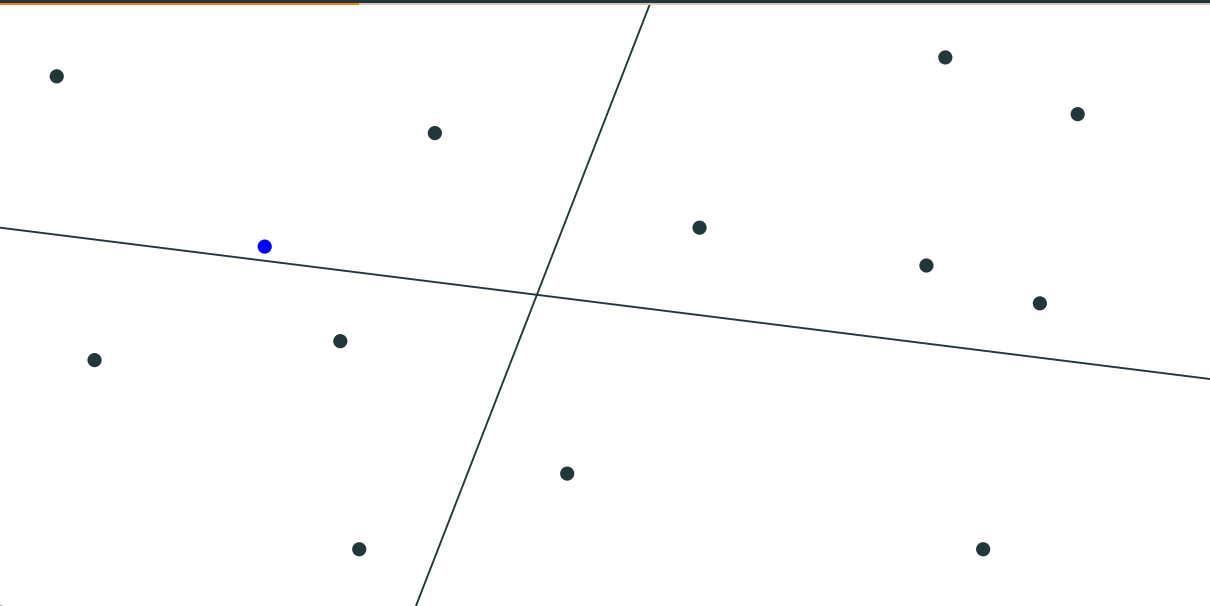
Partition-based methods – Hash table lookups



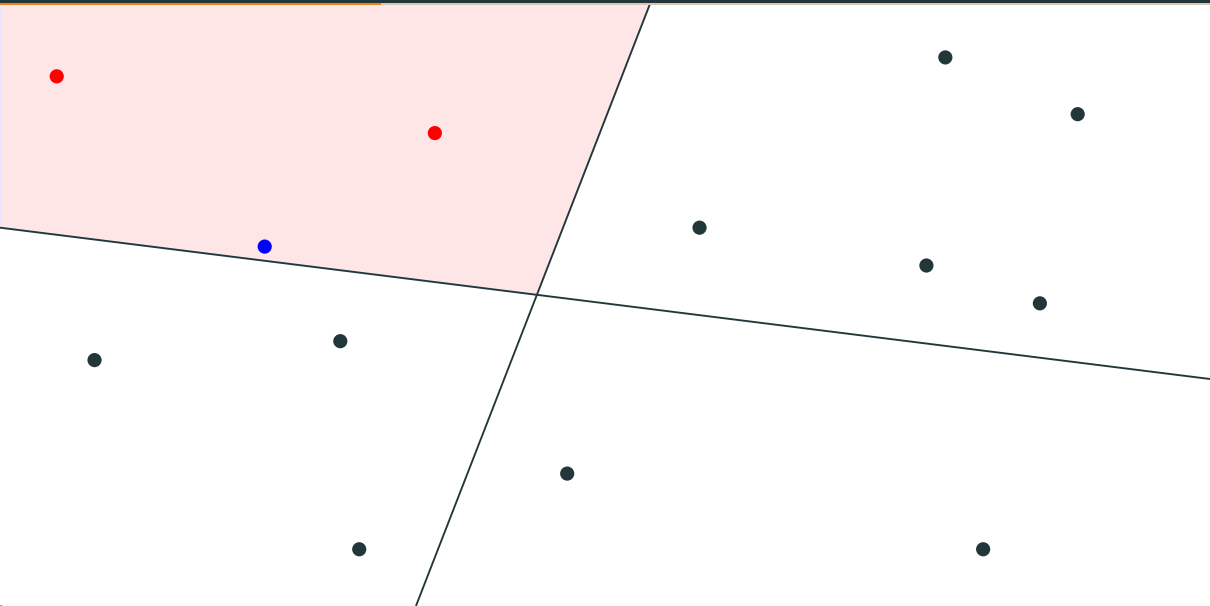
Partition-based methods – Hash table lookups



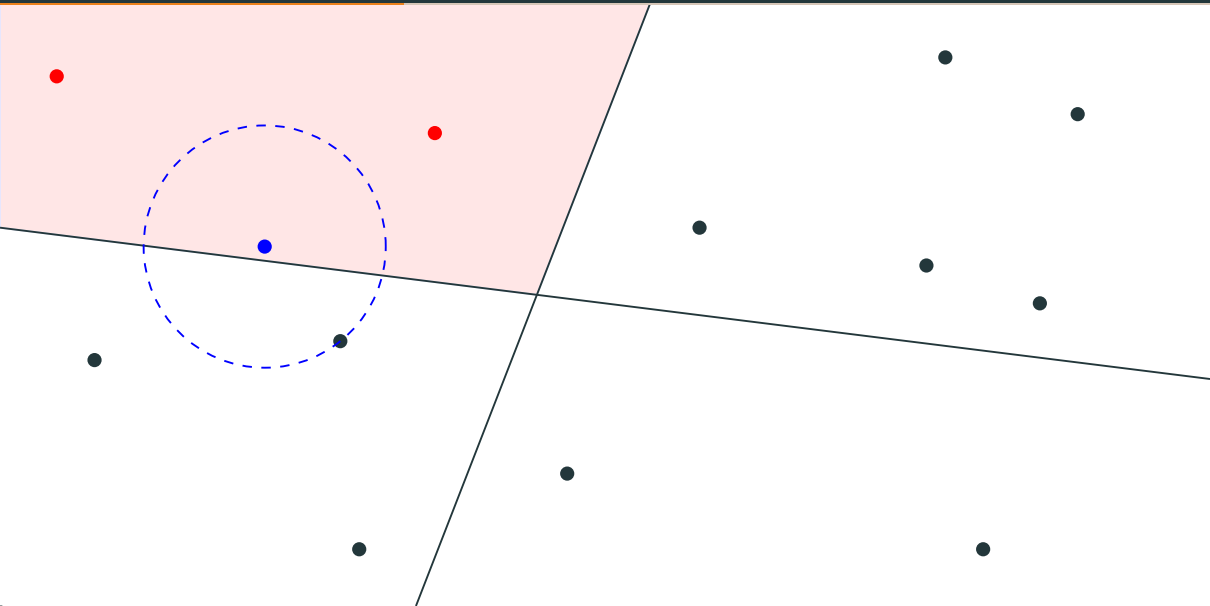
Partition-based methods – Near the boundaries



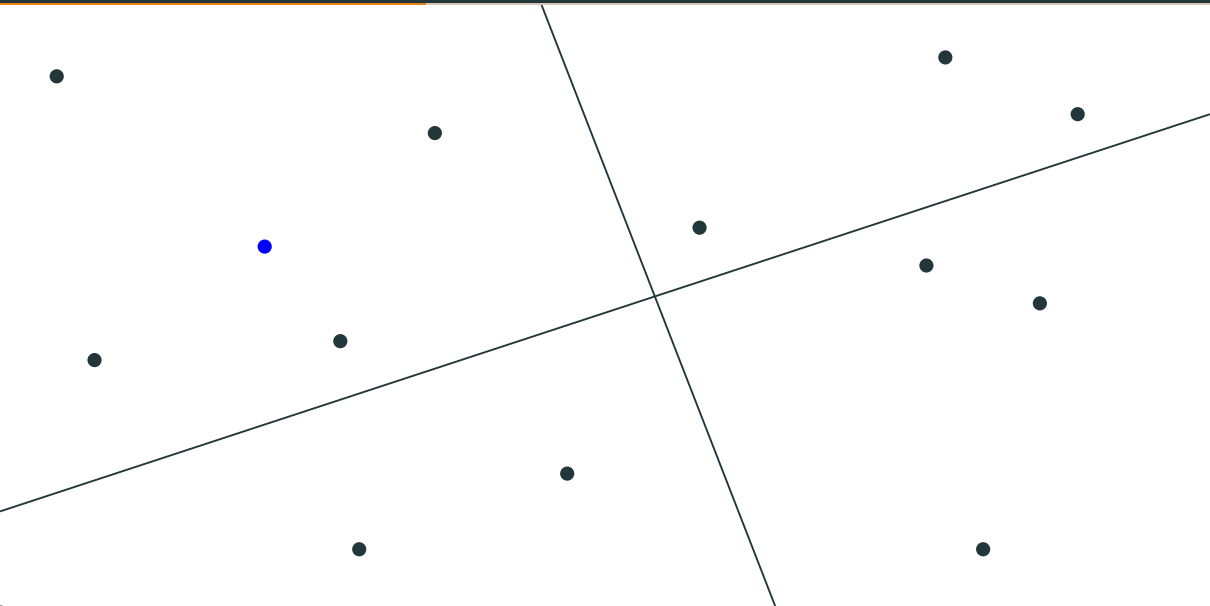
Partition-based methods – Near the boundaries



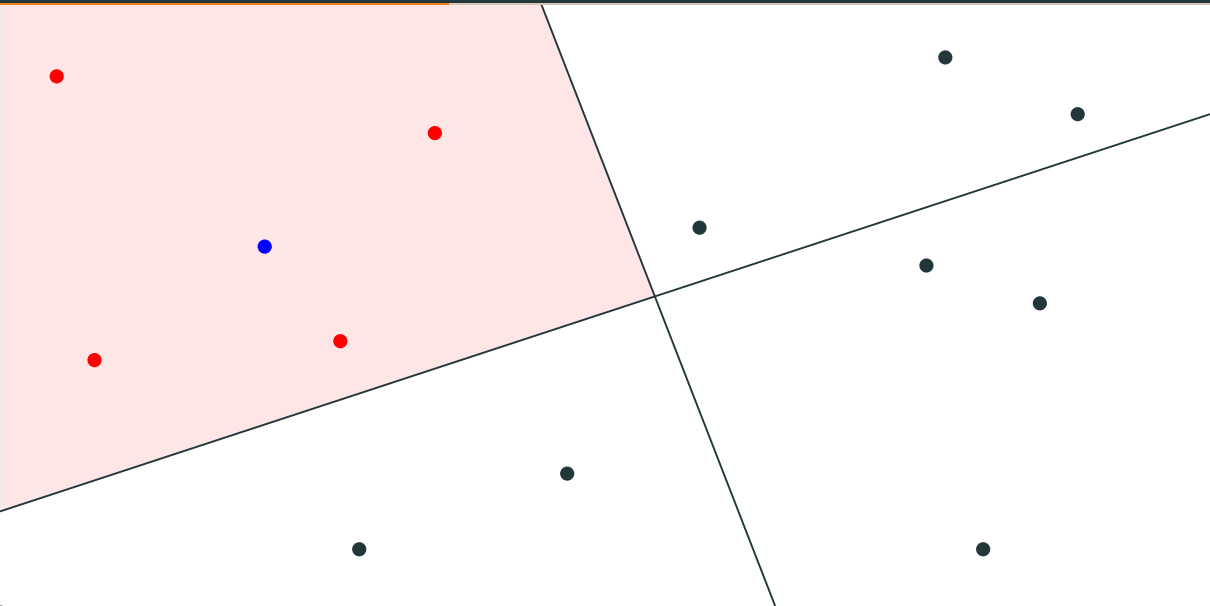
Partition-based methods – Near the boundaries



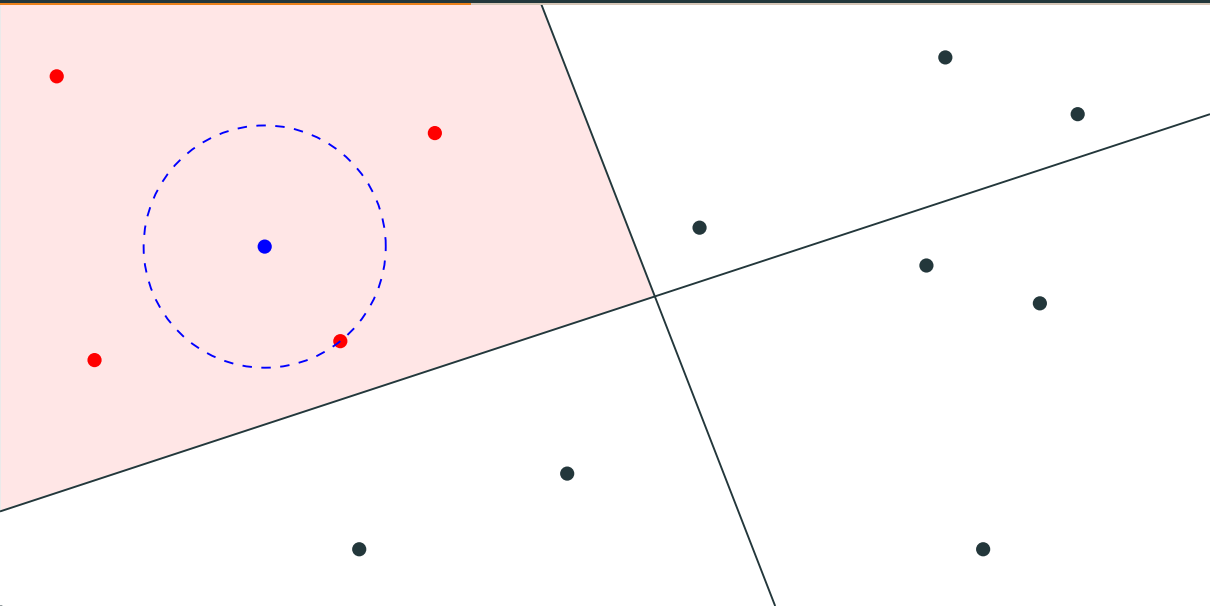
Partition-based methods – Randomizations



Partition-based methods – Randomizations



Partition-based methods – Randomizations



Partition-based methods – Challenges

Main problem: choosing the best types of space partitions.

- Requires an **efficient decoding** algorithm;
- Space partitions should have **nice shapes**.

Partition-based methods – Challenges

Main problem: choosing the best types of space partitions.

- Requires an **efficient decoding** algorithm;
- Space partitions should have **nice shapes**.

Utopia: disjoint spheres lying on an efficiently decodable code or lattice.

Partition-based methods – Challenges

Main problem: choosing the best types of space partitions.

- Requires an **efficient decoding** algorithm;
- Space partitions should have **nice shapes**.

Utopia: disjoint spheres lying on an efficiently decodable code or lattice.

Real world: approximate ideal solution as best as we can.

- Product of bisections; [Cha03]
- Voronoi cells induced by hypercube; [TT07, Laa16]
- Random (overlapping) spheres; [AI06, AINR14]
- Voronoi cells induced by cross-polytopes; [TT07, AIL+15, KW17]
- Voronoi cells induced by (pseudo)random points. [BDGL16, ALRW17, Chr17]

Partition-based methods – Challenges

Main problem: choosing the best types of space partitions.

- Requires an **efficient decoding** algorithm;
- Space partitions should have **nice shapes**.

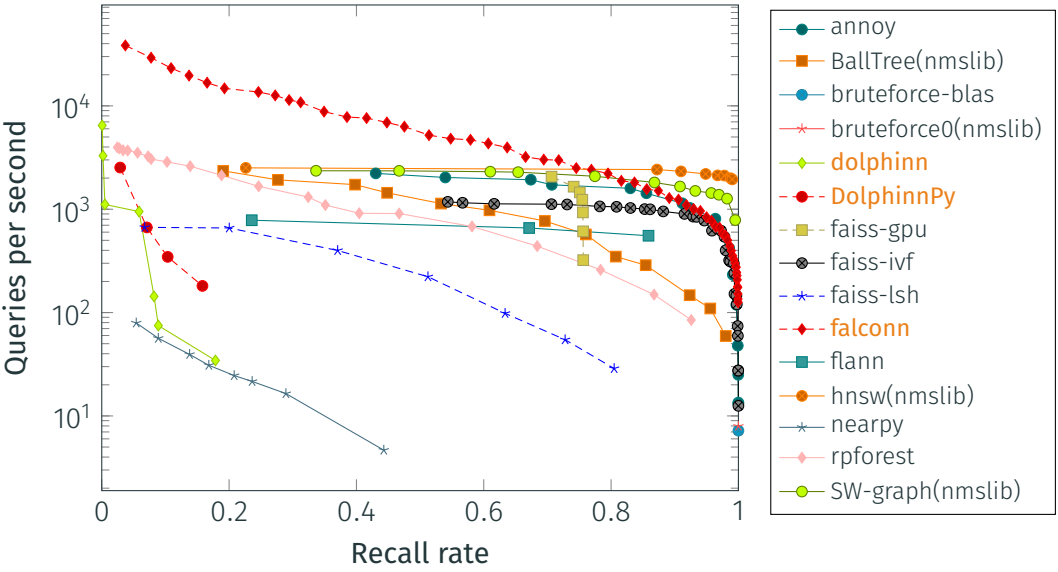
Utopia: disjoint spheres lying on an efficiently decodable code or lattice.

Real world: approximate ideal solution as best as we can.

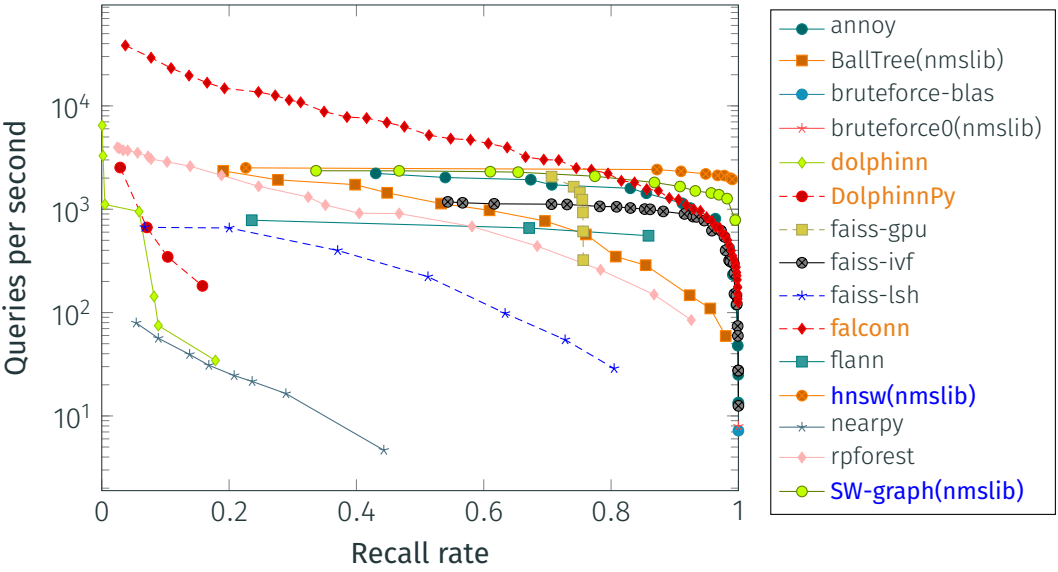
- Product of bisections; [Cha03]
- Voronoi cells induced by hypercube; [TT07, Laa16]
- Random (overlapping) spheres; [AI06, AINR14]
- Voronoi cells induced by cross-polytopes; [TT07, AIL+15, KW17]
- Voronoi cells induced by (pseudo)random points. [BDGL16, ALRW17, Chr17]

Best techniques are **theoretically optimal** as well as **practical**.

Nearest neighbor methods – Practice (ANN Benchmarks [ABF17])

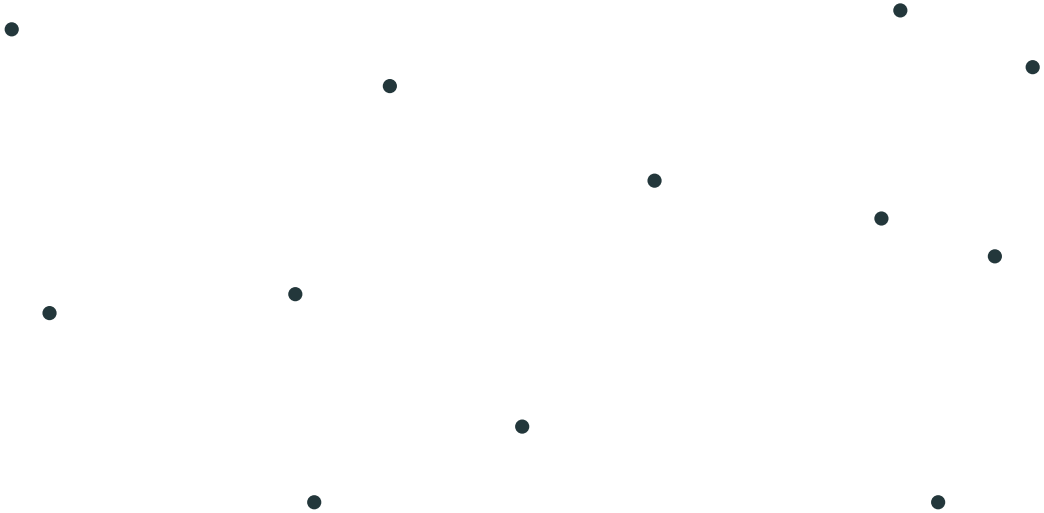


Nearest neighbor methods – Practice (ANN Benchmarks [ABF17])

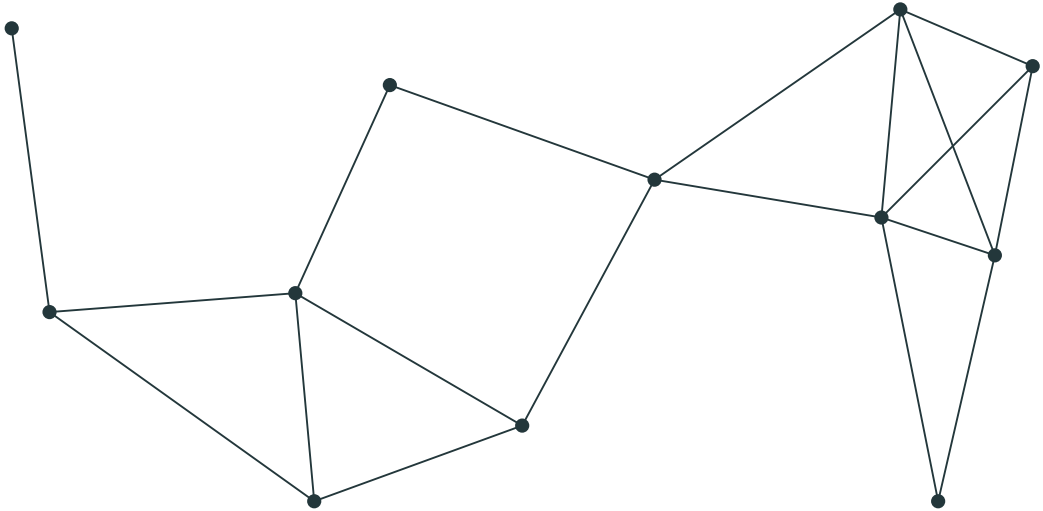


Graph-based methods

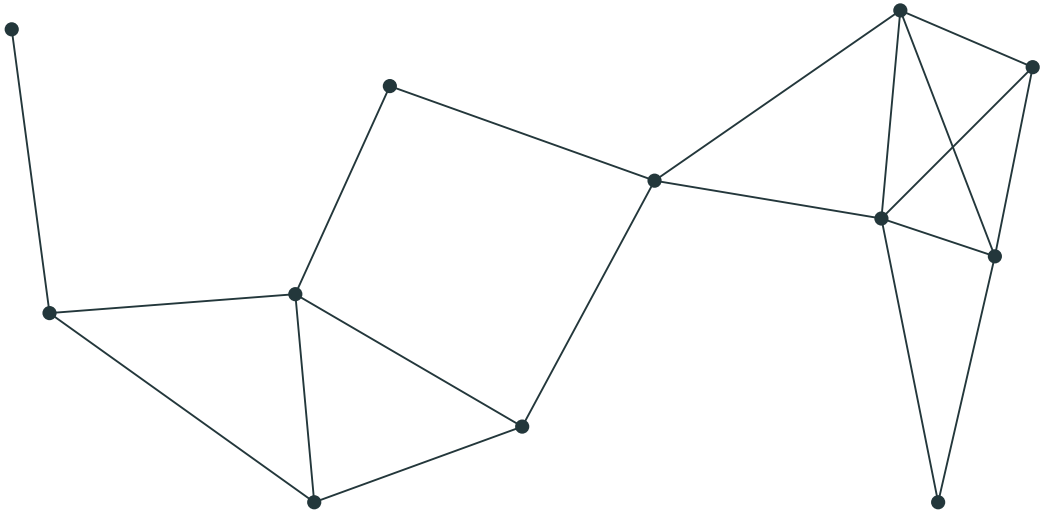
Graph-based methods – Data structure



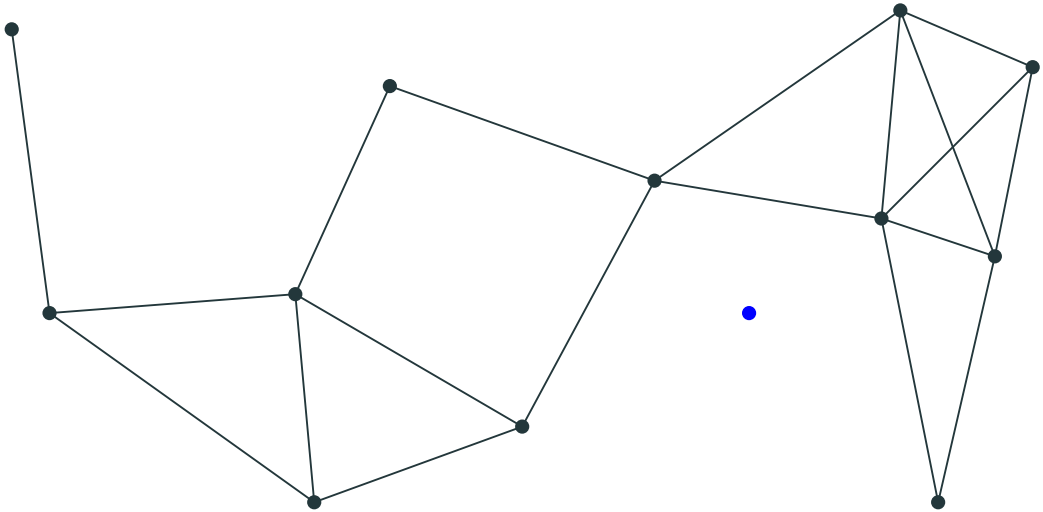
Graph-based methods – Data structure



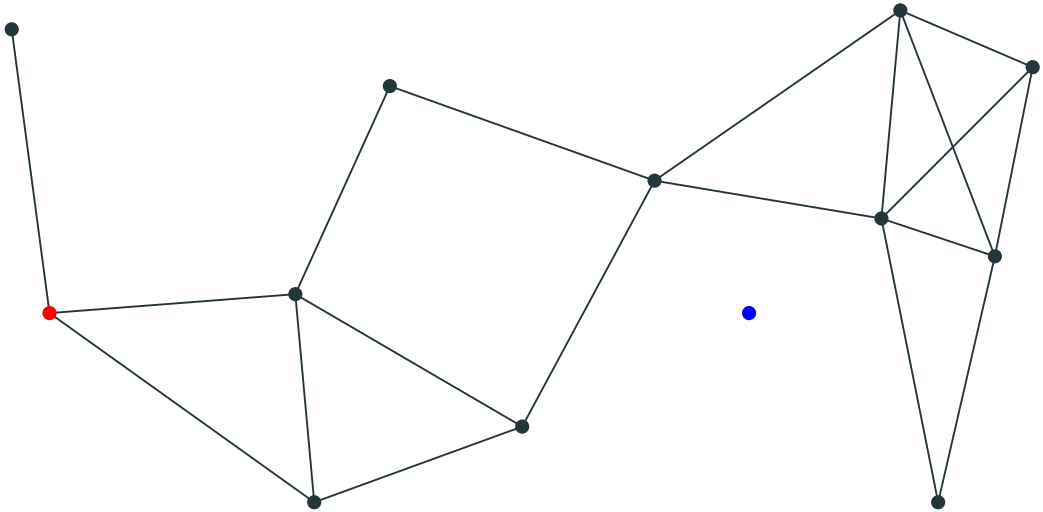
Graph-based methods – Greedy algorithm



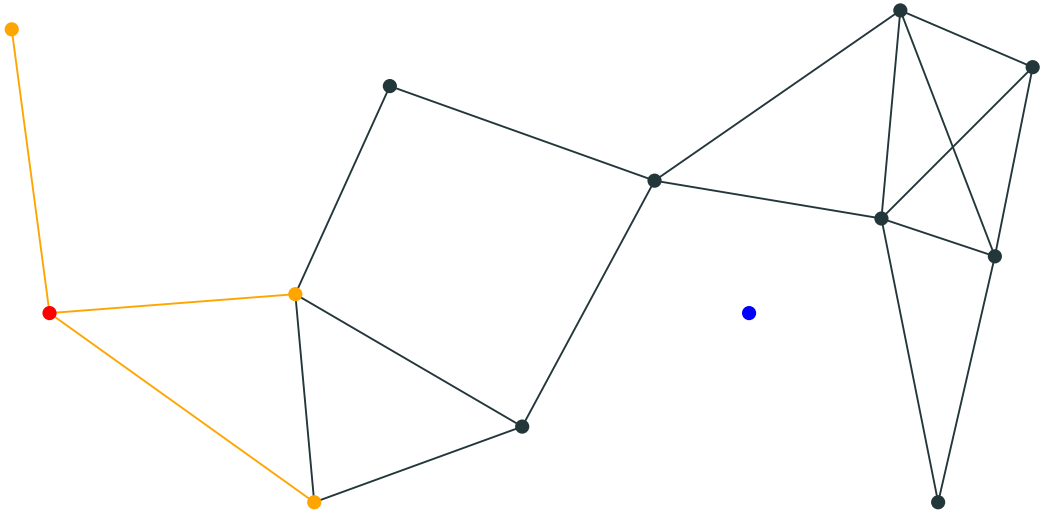
Graph-based methods – Greedy algorithm



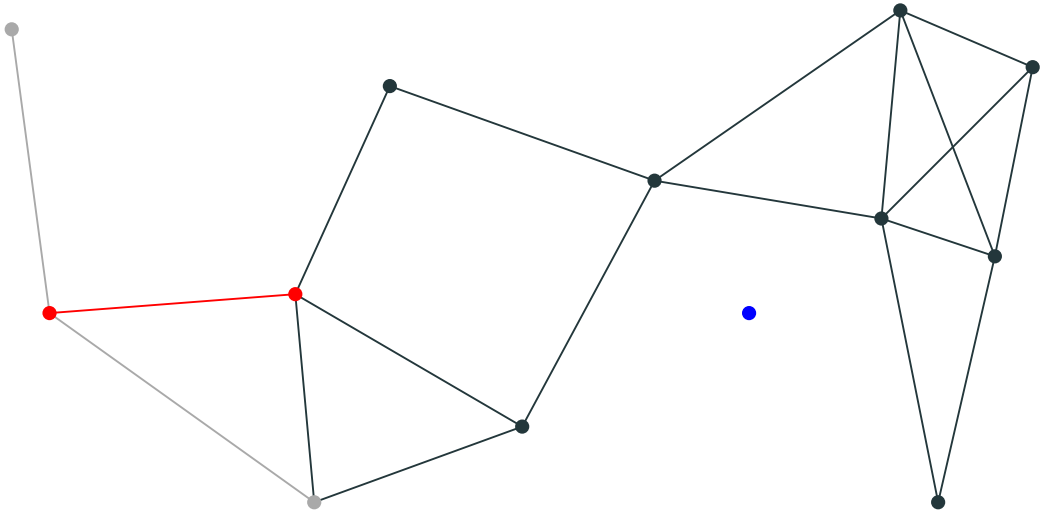
Graph-based methods – Greedy algorithm



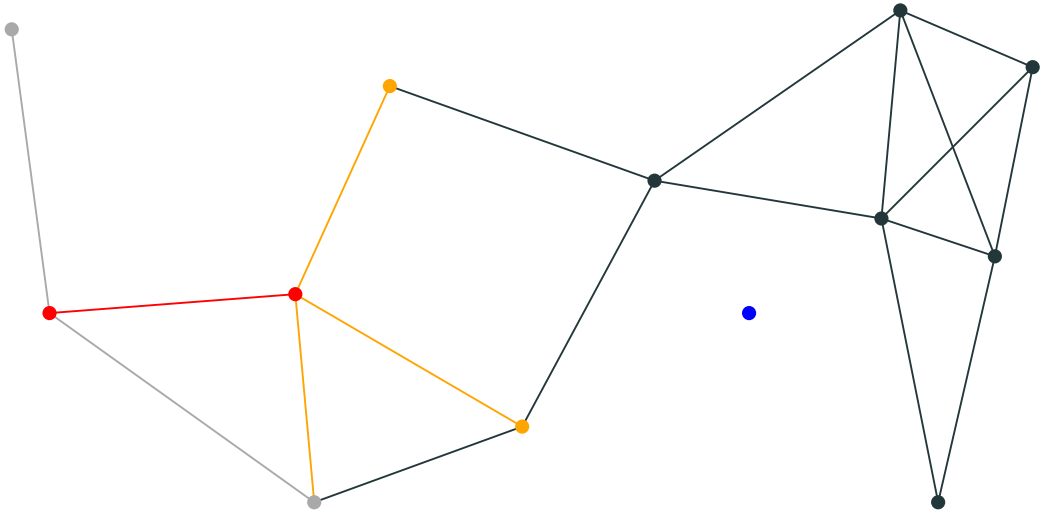
Graph-based methods – Greedy algorithm



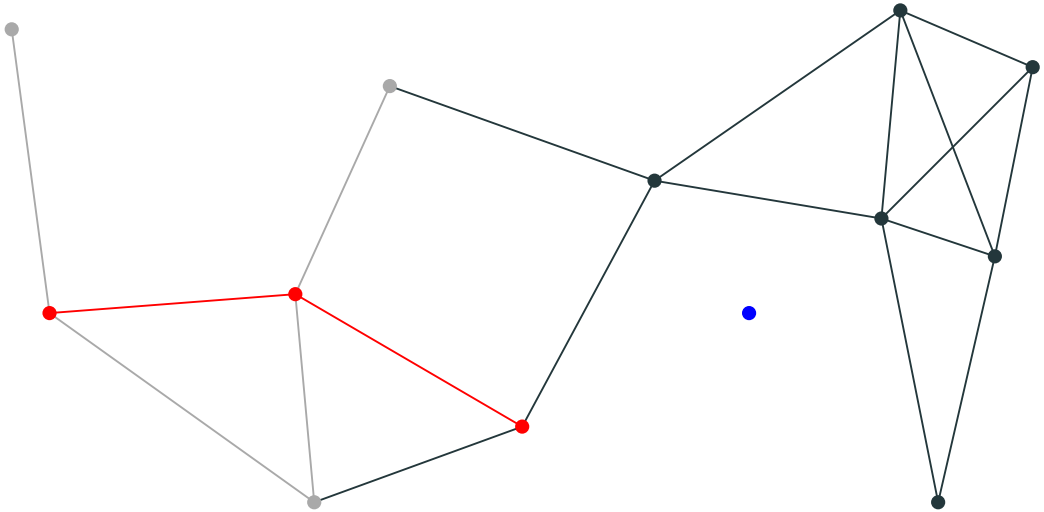
Graph-based methods – Greedy algorithm



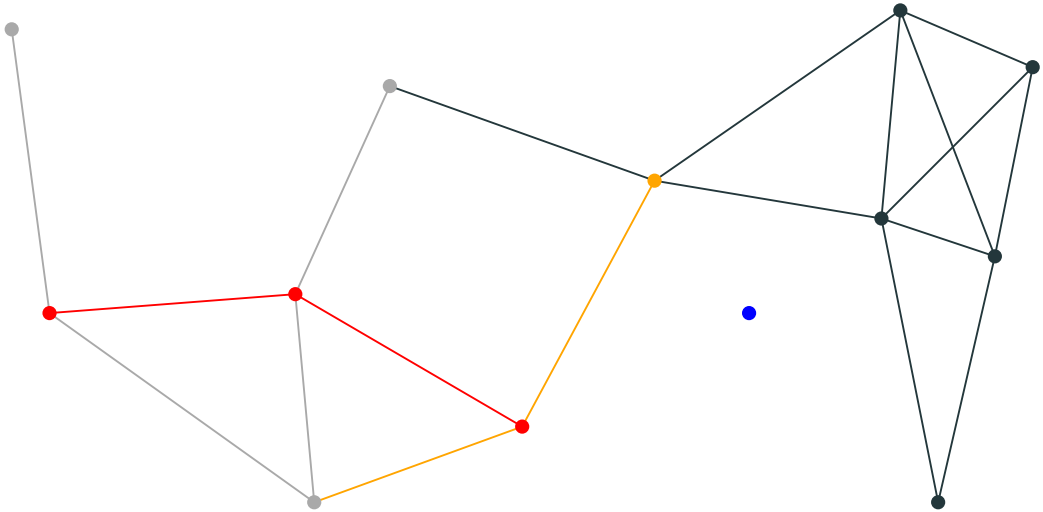
Graph-based methods – Greedy algorithm



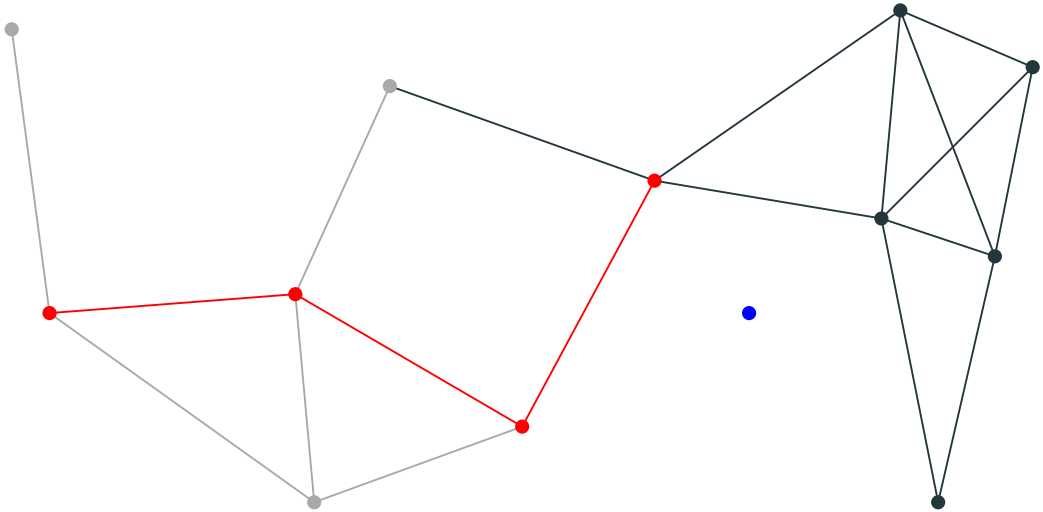
Graph-based methods – Greedy algorithm



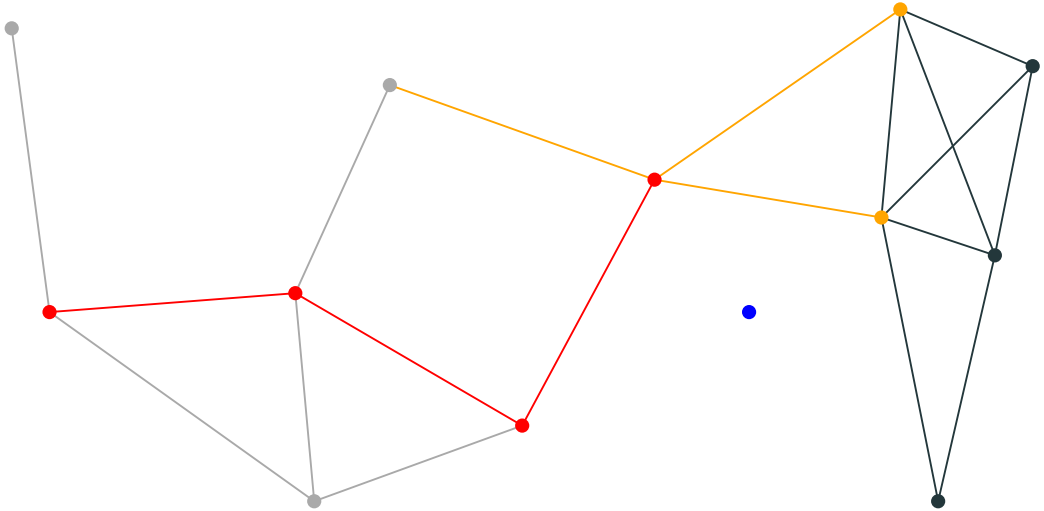
Graph-based methods – Greedy algorithm



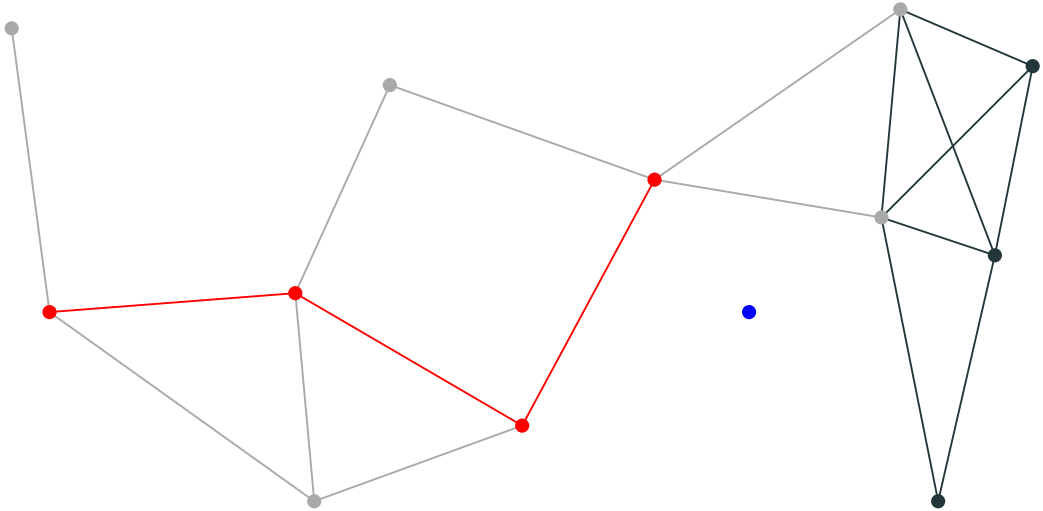
Graph-based methods – Greedy algorithm



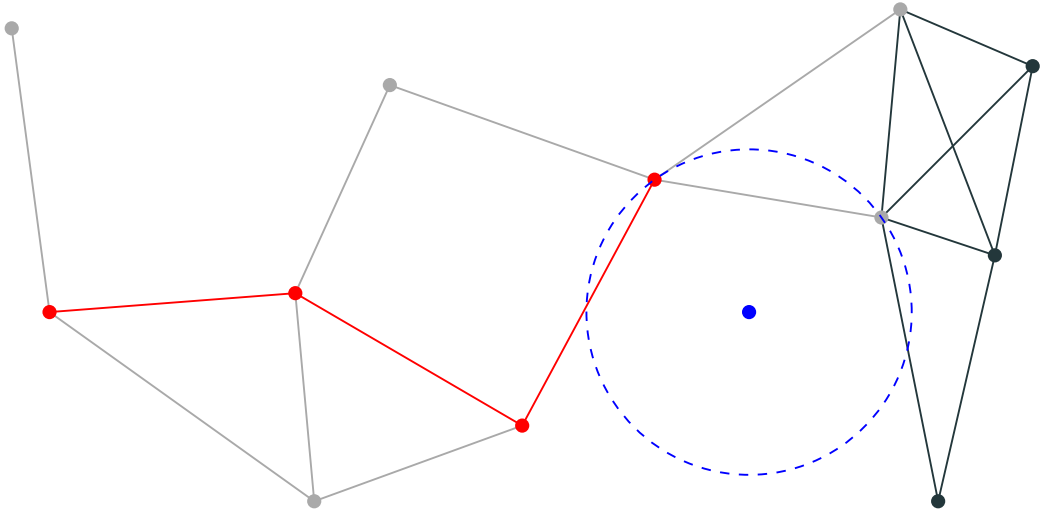
Graph-based methods – Greedy algorithm



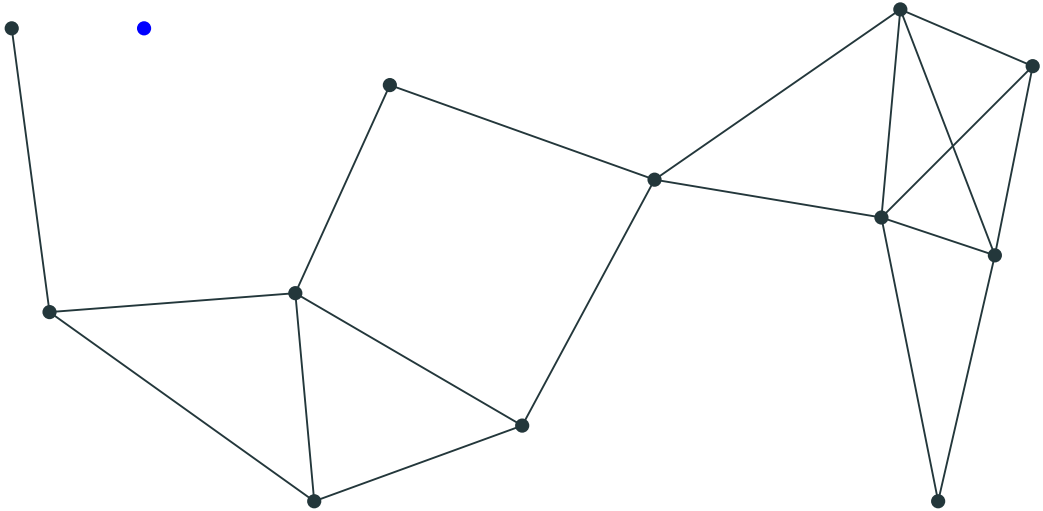
Graph-based methods – Greedy algorithm



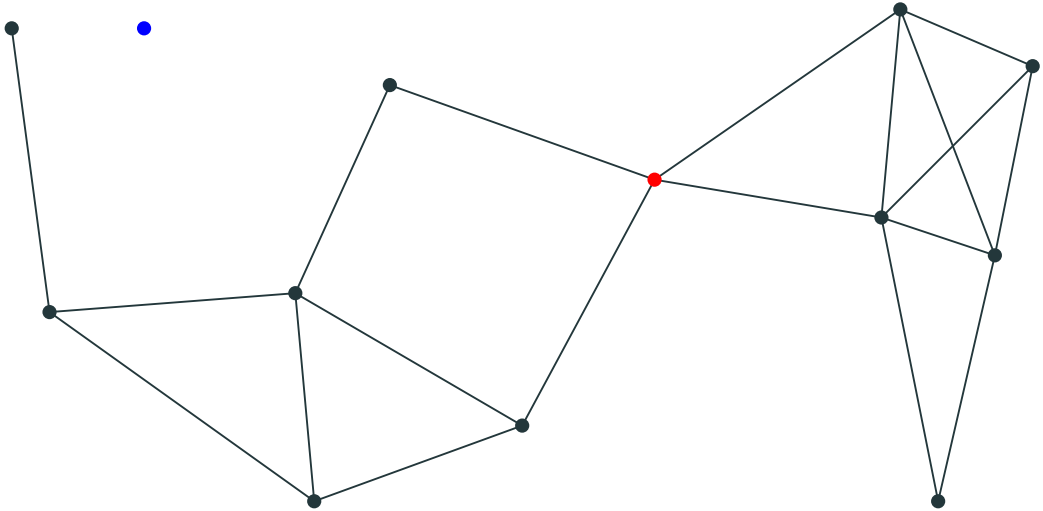
Graph-based methods – Greedy algorithm



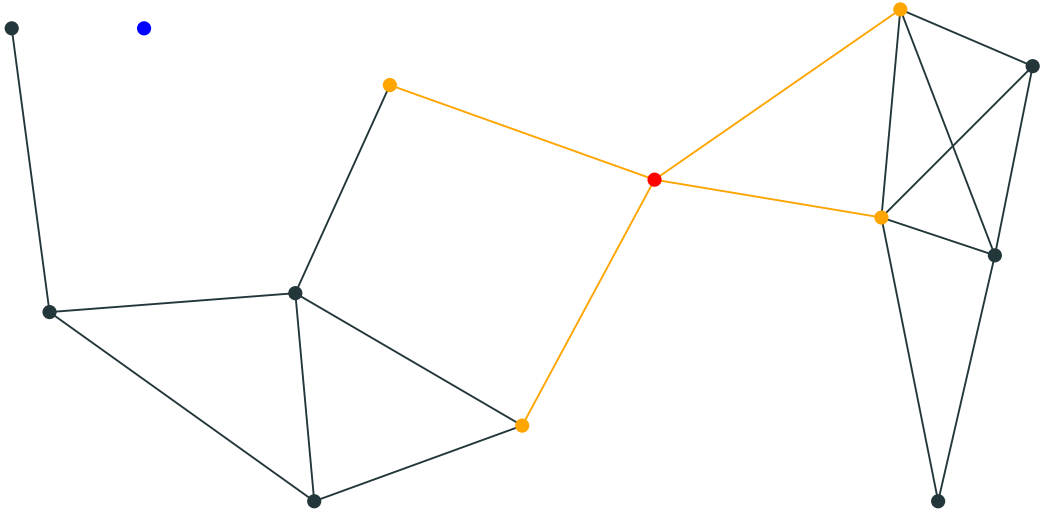
Graph-based methods – Local solutions



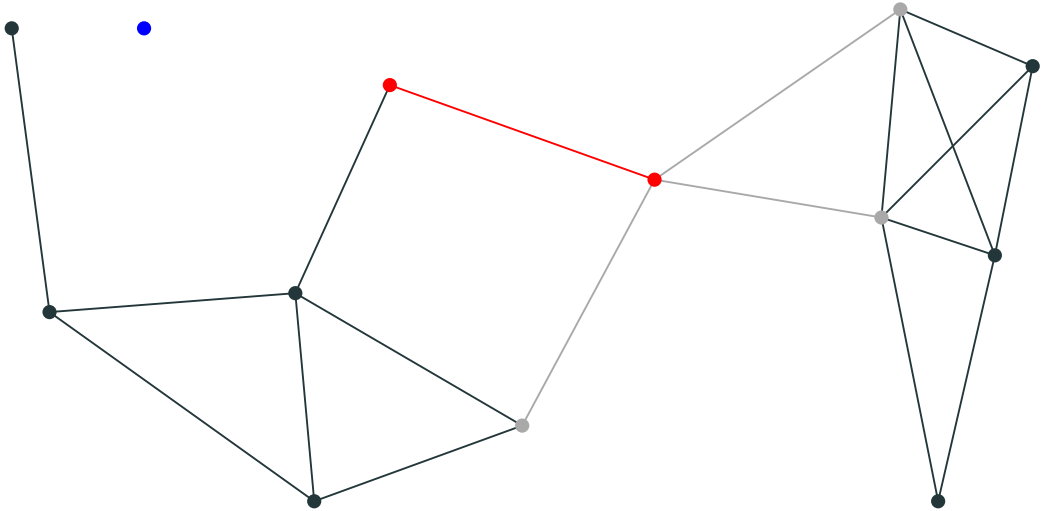
Graph-based methods – Local solutions



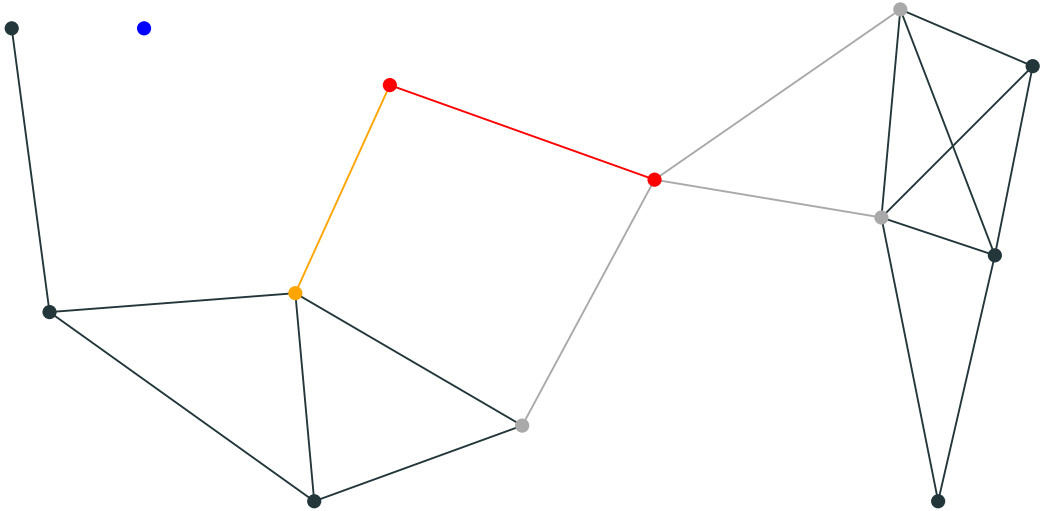
Graph-based methods – Local solutions



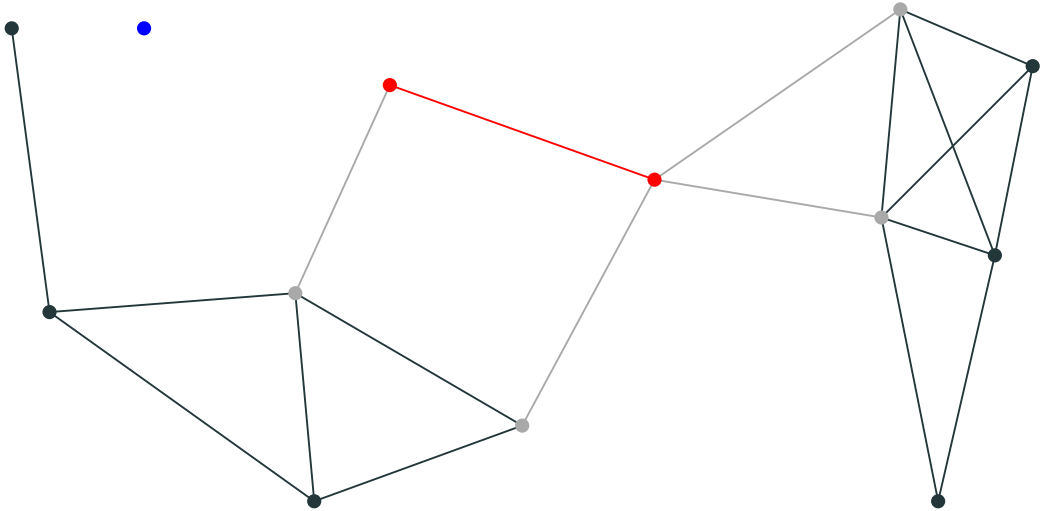
Graph-based methods – Local solutions



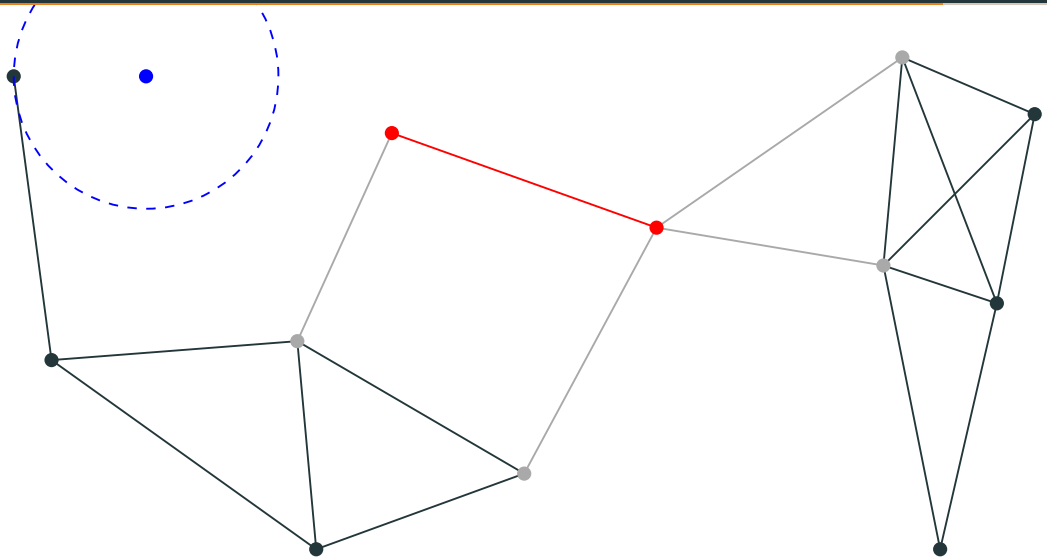
Graph-based methods – Local solutions



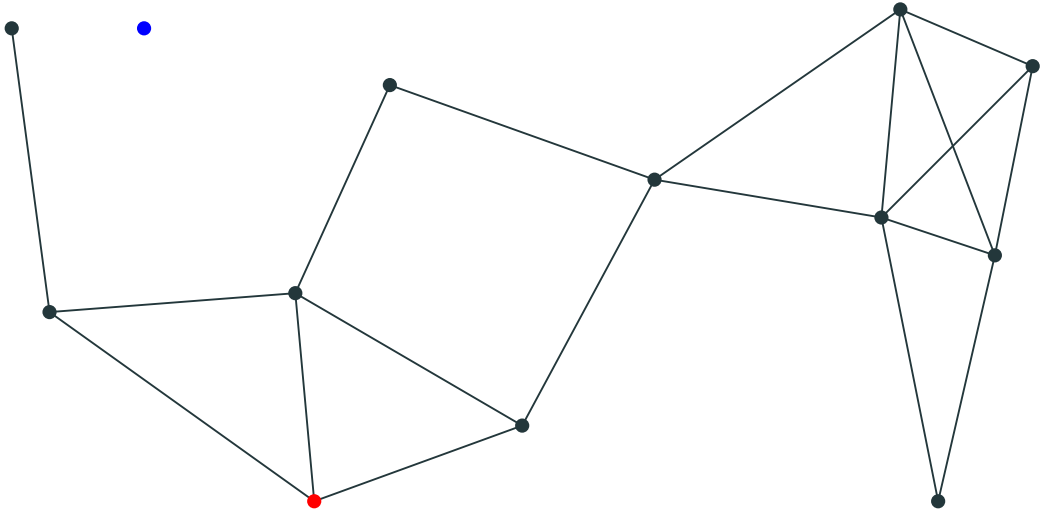
Graph-based methods – Local solutions



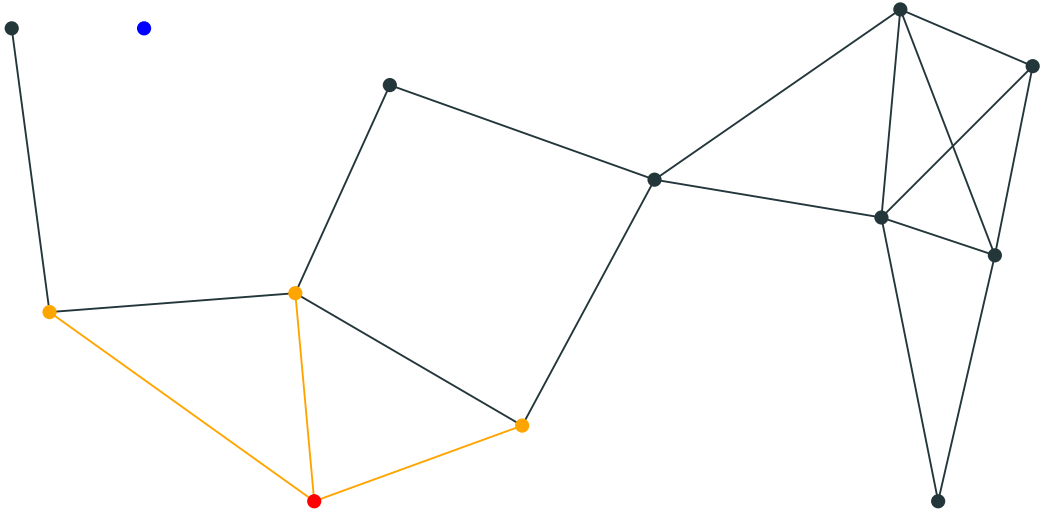
Graph-based methods – Local solutions



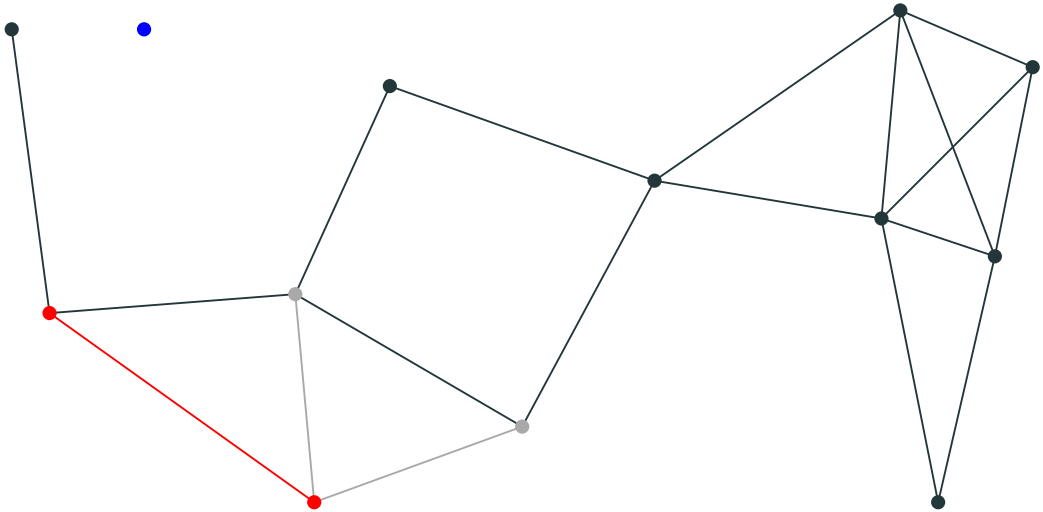
Graph-based methods – Randomizations



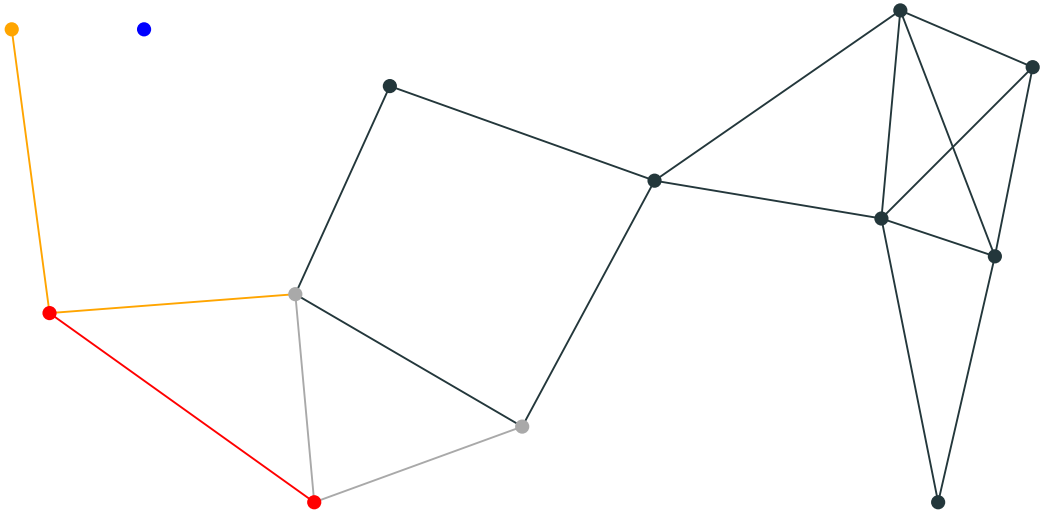
Graph-based methods – Randomizations



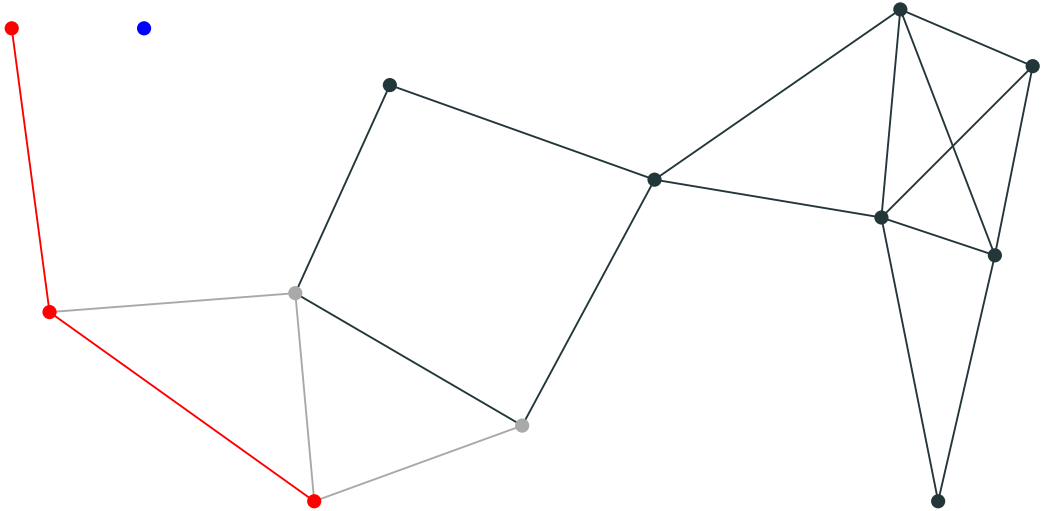
Graph-based methods – Randomizations



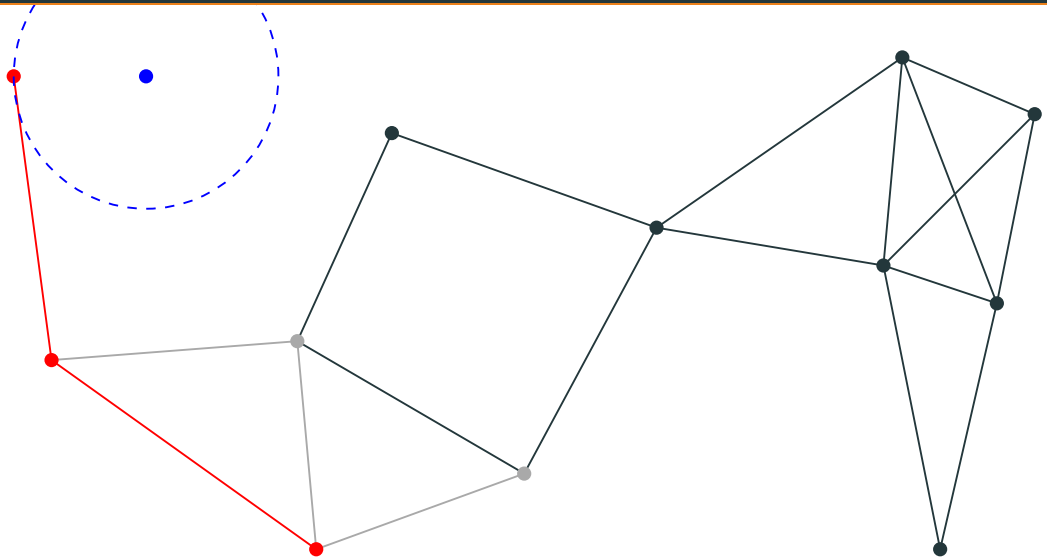
Graph-based methods – Randomizations



Graph-based methods – Randomizations



Graph-based methods – Randomizations



Graph-based methods – Challenges

Main problem: designing the graph.

- Intuitively: connect **near neighbors** for gradual progress;
- Avoiding local minima: add a few **long edges**;
- **Hierarchical graphs**: long edges in upper layers, short edges in bottom layers.

Graph-based methods – Challenges

Main problem: designing the graph.

- Intuitively: connect **near neighbors** for gradual progress;
- Avoiding local minima: add a few **long edges**;
- **Hierarchical graphs**: long edges in upper layers, short edges in bottom layers.

Practically, graph-based methods are very efficient as well.

Graph-based methods – Challenges

Main problem: designing the graph.

- Intuitively: connect **near neighbors** for gradual progress;
- Avoiding local minima: add a few **long edges**;
- **Hierarchical graphs**: long edges in upper layers, short edges in bottom layers.

Practically, graph-based methods are very efficient as well.

Theoretically, little is known about the performance of these methods.

Graph-based methods – Challenges

Main problem: designing the graph.

- Intuitively: connect **near neighbors** for gradual progress;
- Avoiding local minima: add a few **long edges**;
- **Hierarchical graphs**: long edges in upper layers, short edges in bottom layers.

Practically, graph-based methods are very efficient as well.

Theoretically, little is known about the performance of these methods.

Graph-based methods – Contributions

Theorem (Main result, informal)

For randomized greedy walks on the near neighbor graph and for "random" data sets, we can solve the approximate nearest neighbor problem on n points with query time $O(n^{\rho_q})$ and space $O(n^{1+\rho_s})$ with $\rho_q, \rho_s \geq 0$ satisfying

$$(2c^2 - 1)\rho_q + 2c^2(c^2 - 1)\sqrt{\rho_s(1 - \rho_s)} \geq c^4.$$

Graph-based methods – Contributions

In the most common regime of $c \approx 1$ (**high recall rate**) and $\rho_s \approx 0$ (near-linear space), this **scales equivalently** as the best partition-based trade-offs: [ALRW17]

$$\rho_q = 1 - 4(c-1)\sqrt{\rho_s} \cdot (1 + o(1)). \tag{1}$$

Graph-based methods – Contributions

In the most common regime of $c \approx 1$ (**high recall rate**) and $\rho_s \approx 0$ (near-linear space), this **scales equivalently** as the best partition-based trade-offs: [ALRW17]

$$\rho_q = 1 - 4(c-1)\sqrt{\rho_s} \cdot (1 + o(1)). \tag{1}$$

Positive result: greedy algorithm already "optimal" for $c \approx 1$ and $\rho_s \approx 0$.

Graph-based methods – Contributions

In the most common regime of $c \approx 1$ (**high recall rate**) and $\rho_s \approx 0$ (near-linear space), this **scales equivalently** as the best partition-based trade-offs: [ALRW17]

$$\rho_q = 1 - 4(c-1)\sqrt{\rho_s} \cdot (1 + o(1)). \tag{1}$$

Positive result: greedy algorithm already "optimal" for $c \approx 1$ and $\rho_s \approx 0$.

Negative result: (analysis of) this algorithm is not competitive for $c \gg 1$ or $\rho_s \gg 0$.

Graph-based methods – Open problems

Various **open problems** remain:

- Current **analysis** may not be sharp – can it be tightened?
- Does adding **long edges** lead to better theoretical guarantees?
- What can theoretically be said about **hierarchical approaches**?
- Can we obtain **lower bounds** showing limitations of graph-based methods?

Thank you!