Algorithmic aspects of MAK reconstruction II

Treatment of real catalogues

Michel Hénon

Roya Mohayaee

Andreĭ Sobolevskiĭ

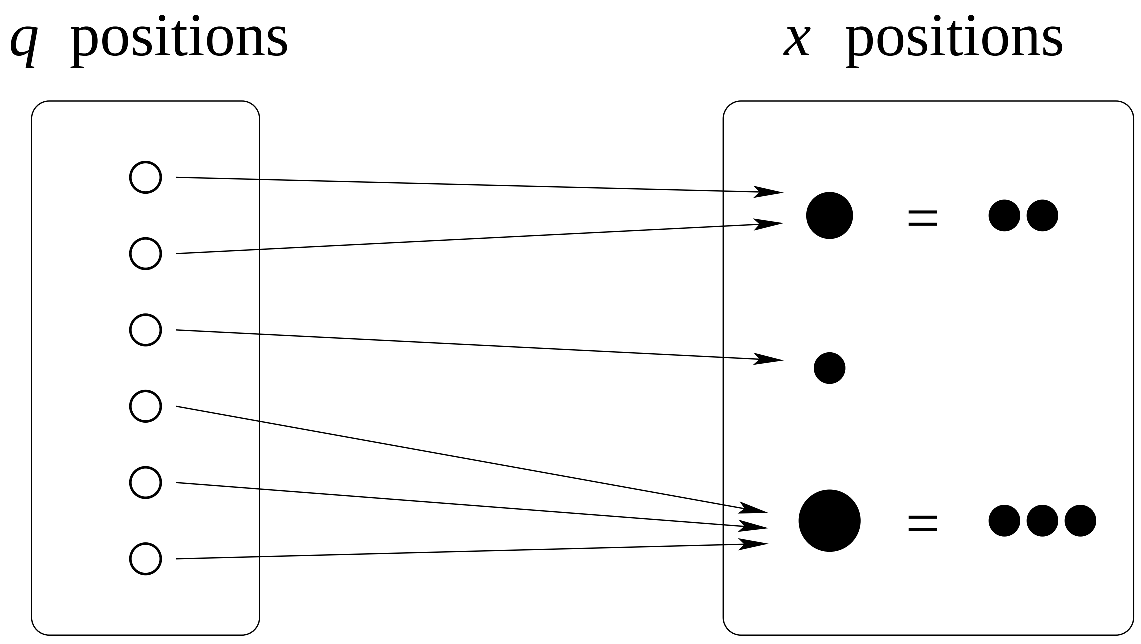
Observatoire de la Côte d'Azur

Nice

Features of real catalogues to be accounted for:

- Large scatter of masses (1 to 10^3)
- Unknown maximal displacement
- Redshift-space datasets

Treatment of large nonunit masses:



"Auctions with similar persons" (Bertsekas):

- k unit masses with the same x position submit a common bid
- search for k best q points is performed simultaneously
- $(k+1)$-best cost is kept instead of the second-best

Search for best-value (lowest-cost) points:

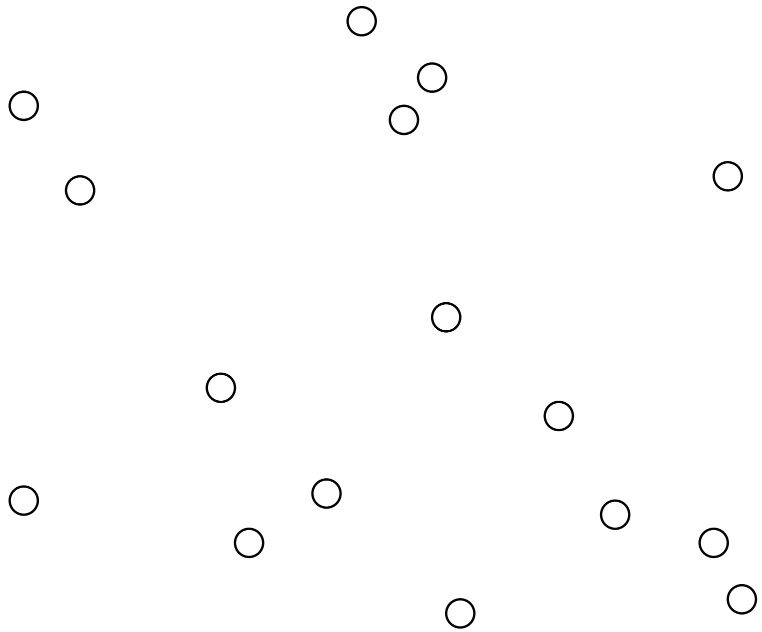
There is no *a priori* upper bound on displacements. Do we have to search all q positions then?

Idea: for a given position x and some position q_0, we want to discard all q positions that are guaranteed to have a larger cost.

Method: kd trees with prices.

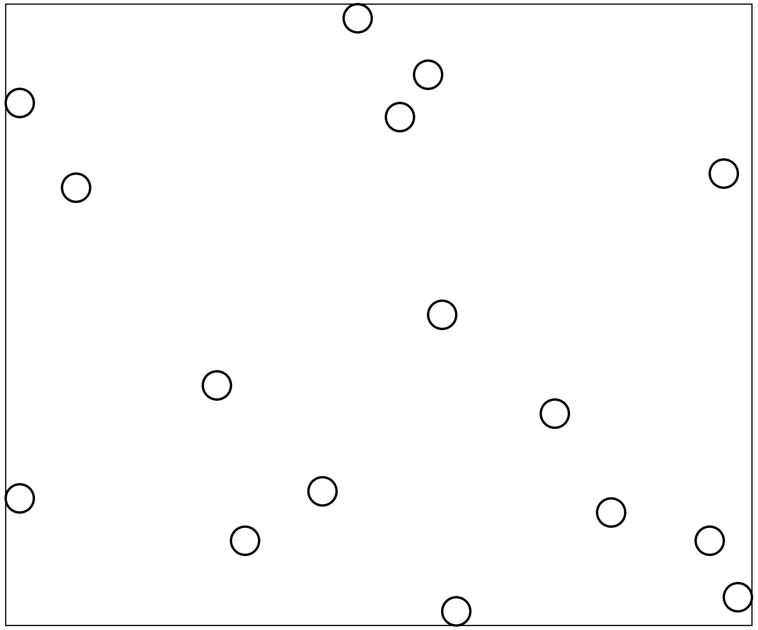
Search for best-value (lowest-cost) points:

Construction of a *kd* tree



Search for best-value (lowest-cost) points:

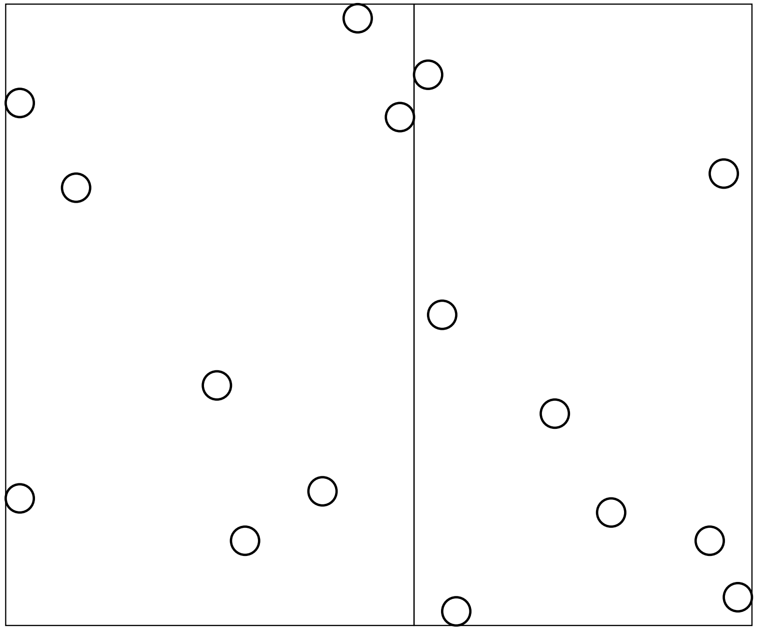
Construction of a *kd* tree



Root node contains all positions in a bounding box

Search for best-value (lowest-cost) points:

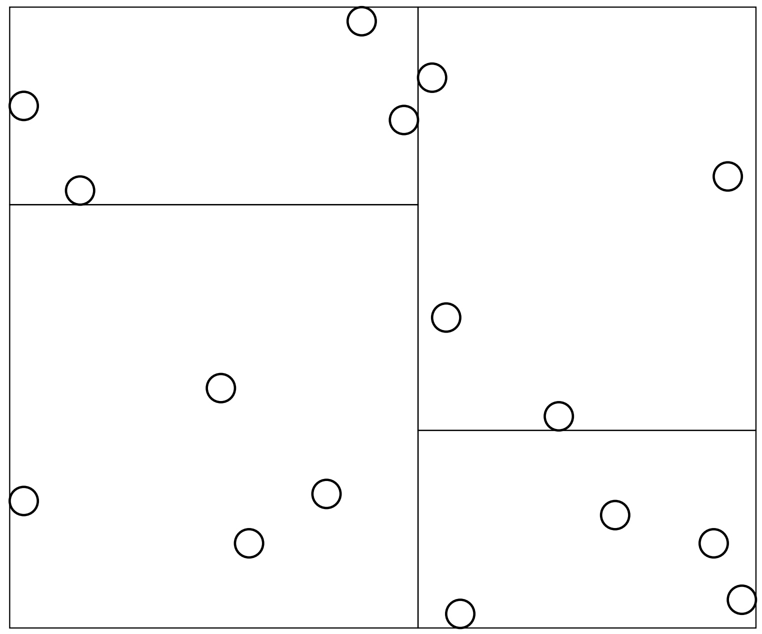
Construction of a *kd* tree



Two level 1 subnodes contain 8 positions each

Search for best-value (lowest-cost) points:

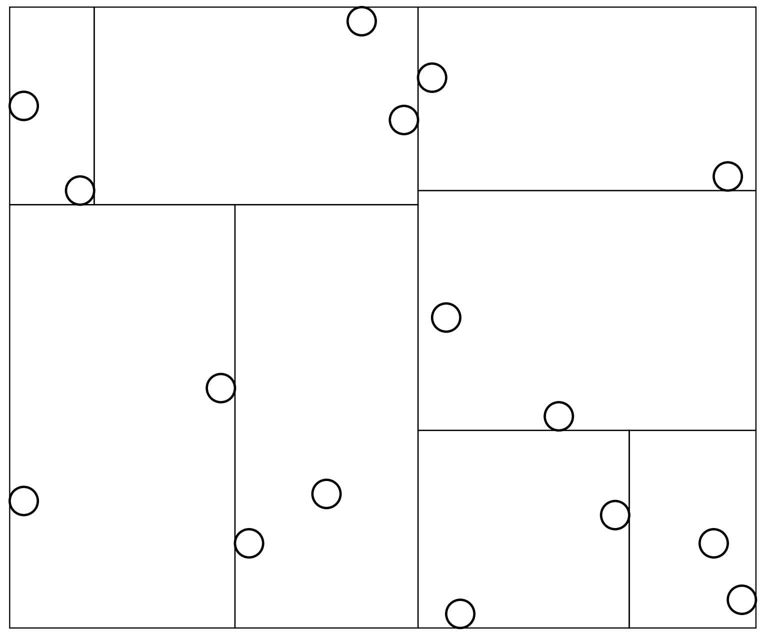
Construction of a *kd* tree



Four level 2 subnodes contain 4 positions each

Search for best-value (lowest-cost) points:

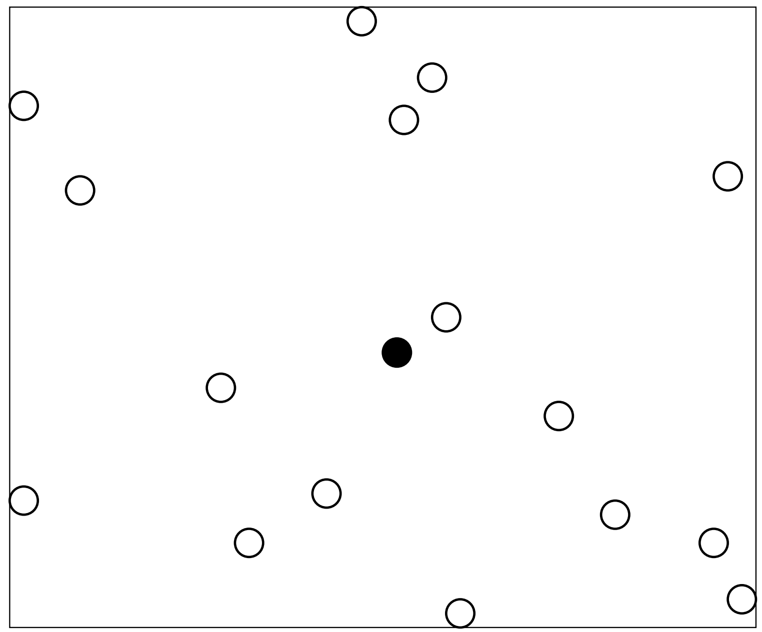
Construction of a *kd* tree



Eight level 3 subnodes contain 2 positions each

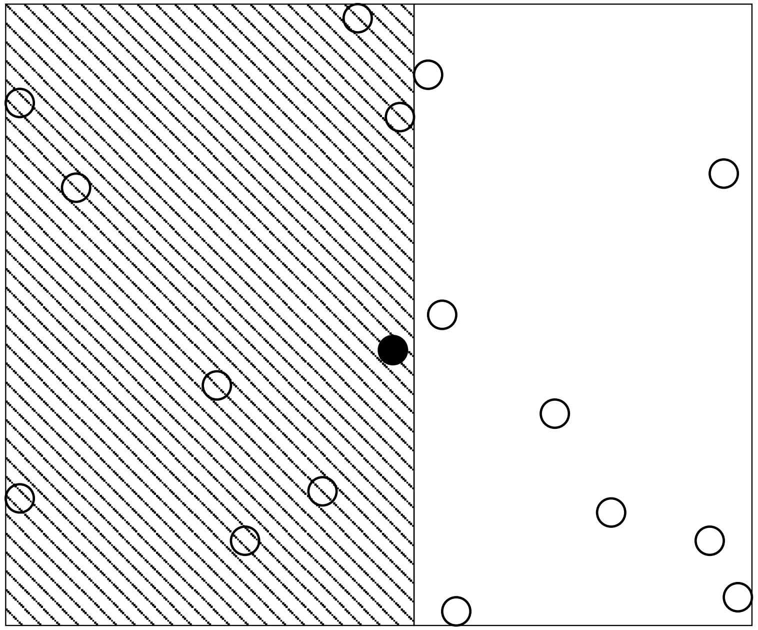
Search for best-value (lowest-cost) points:

Search in a *kd* tree



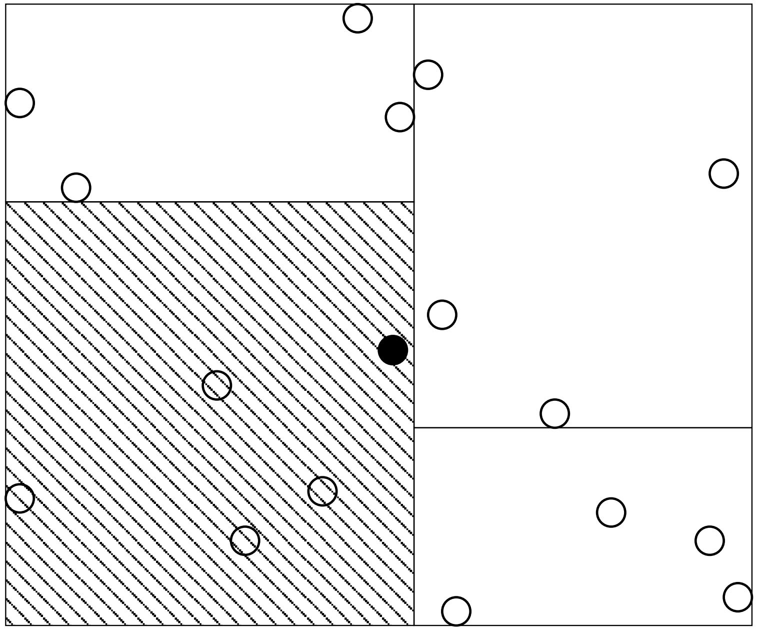
Search for best-value (lowest-cost) points:

Search in a *kd* tree



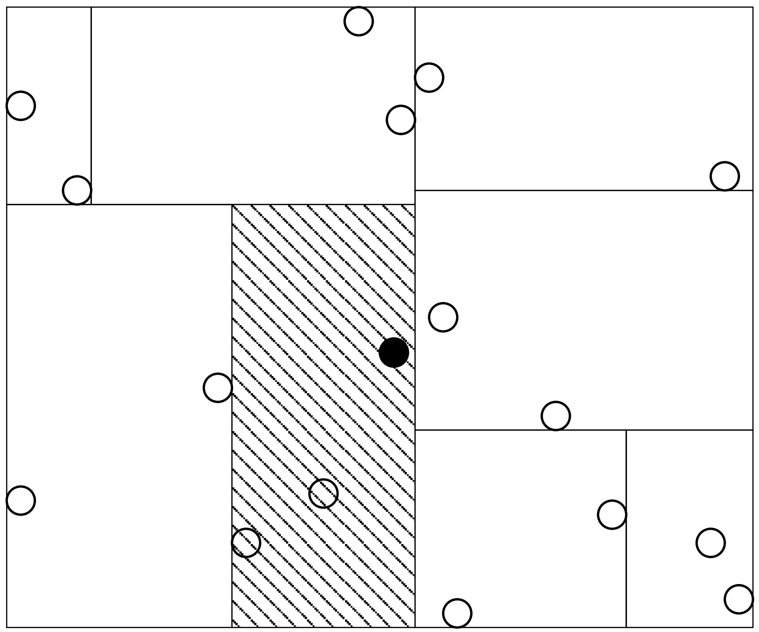
Search for best-value (lowest-cost) points:

Search in a *kd* tree



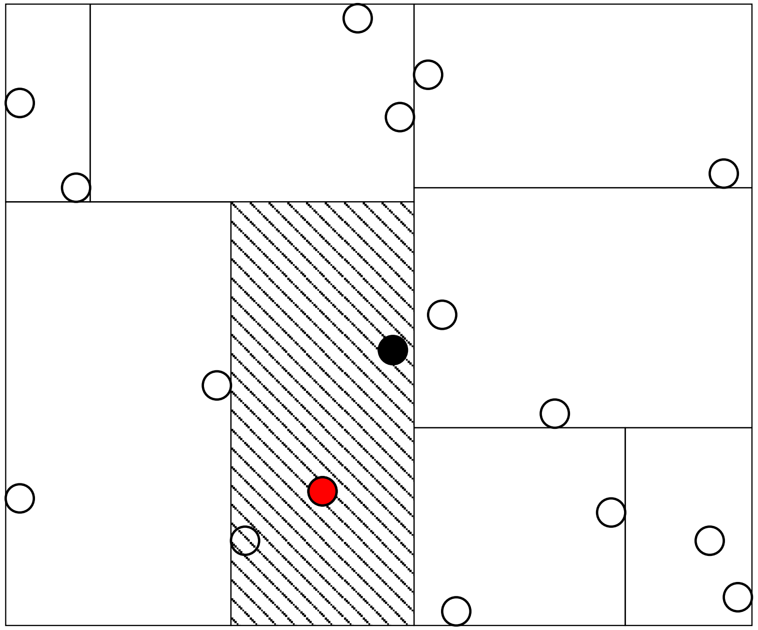
Search for best-value (lowest-cost) points:

Search in a *kd* tree



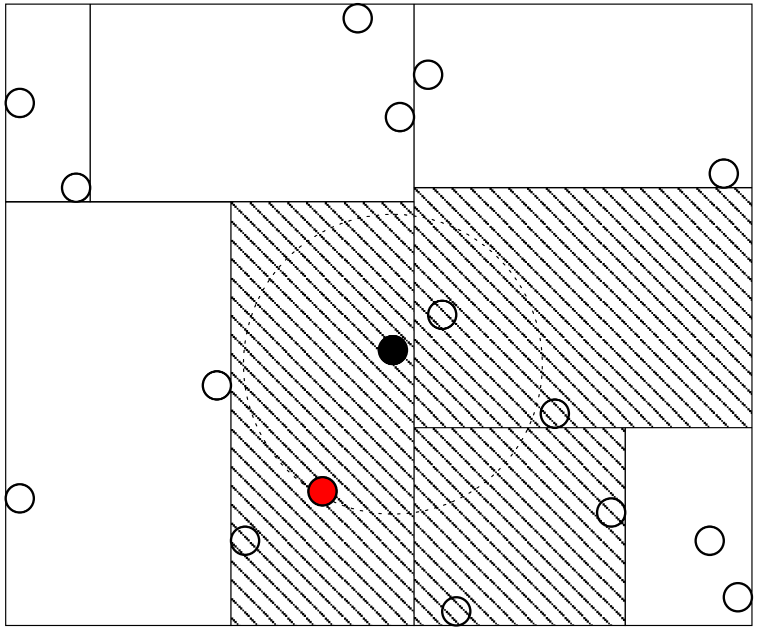
Search for best-value (lowest-cost) points:

Search in a *kd* tree



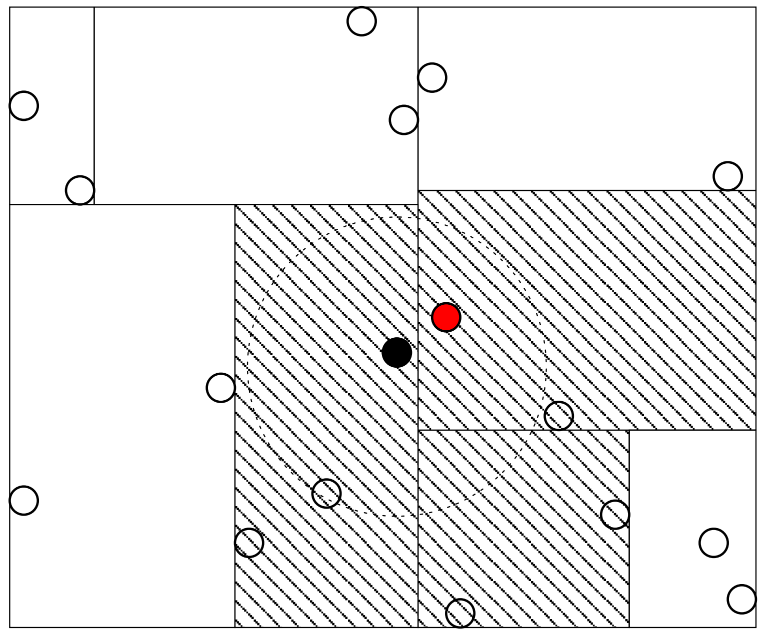
Search for best-value (lowest-cost) points:

Search in a *kd* tree



Search for best-value (lowest-cost) points:

Search in a *kd* tree



Search for best-value (lowest-cost) points:

To look for lowest-cost rather than closest points:

- keep for each subbox the price of the cheapest point in it
- add min prices to squared distances when deciding whether to discard a subbox

Redshift-space costs

q initial, x present position in comoving coordinates.

Cost function: $|x-q|^2 = (x-q_{\|})^2 + |q_\perp|^2$

Apparent redshift radius: $s = x + \dfrac{v_{\|}}{H}$

Radial peculiar velocity in the ZA: $v_{\|} = \beta H(x-q_{\|})$

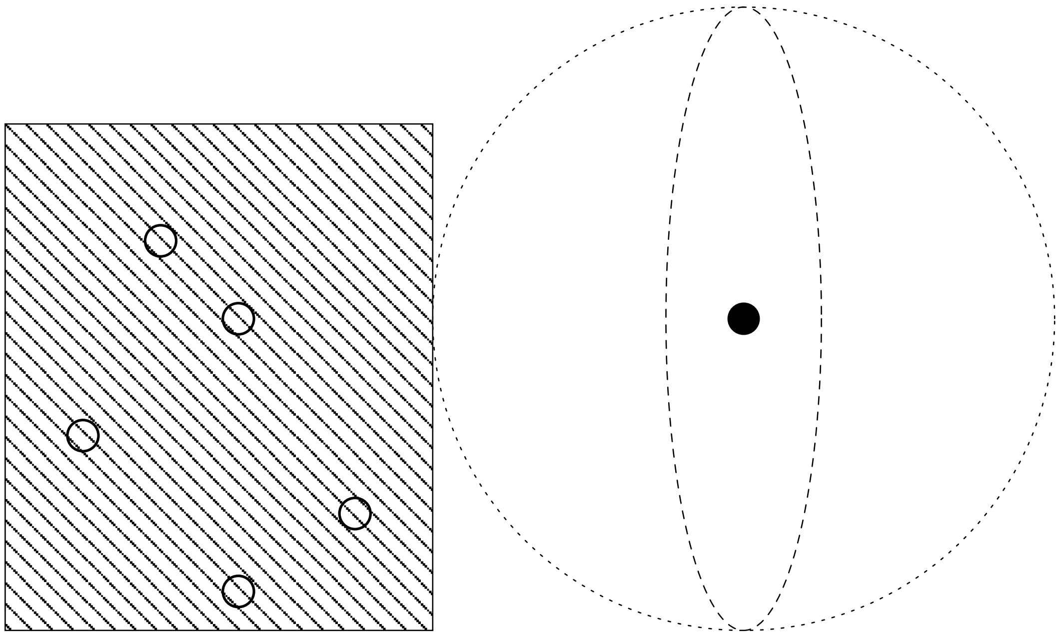
Therefore $s - q_{\|} = (1+\beta)(x-q_{\|})$

Redshift-space cost function: $\dfrac{1}{(1+\beta)^2}(s-q_{\|})^2 + |q_\perp|^2$

Thus "isodistant" surfaces in redshift space are *ellipsoids*

Redshift-space costs

Estimating redshift-space cost from below:



Enclosing ellipsoids into spheres