Geometric Monitoring with Safe-Zones

Large-scale Distributed Computation

Shonan Village Center, Jan. 11-15, 2012

- Assaf Schuster, Tsachi Sharfman, Guy Sagy, Amir Abboud, David Ben-David
- Technion

- Daniel Keren, Tsachi Sharfman
- Haifa University

Research supported by FP7-ICT Programme, Project "LIFT", *Local Inference in Massively Distributed Systems*
http://www.lift-eu.org/

2

Example: monitoring cloud health

- **Small computer cloud (two machines)**
- **Want to submit an alert when <u>average</u> load is < 80%.**
- **But – don't want to keep reporting individual loads.**
- **Trivial solution: every machine keeps silent as long as its load is ≥ 80%.**
- **Better – set different thresholds (robust machine reports when load < 90%, unstable machine when load < 70%).**

❑ The set in which the average of the local data (loads in this case) is allowed to be (in this case the interval **[0.8,1]**), will be called the *admissible region* and denoted by *G.*

❑ The sets in which the local data can roam freely, while their average is guaranteed to be inside the admissible region, are called *safe zones.*

❑ Many possibilities for choosing safe-zone: for example, both **{[0.8,1],[0.8,1]}** and **{[0.7,1],[0.9,1]}** are legal assignments.

❑ The safe zones should not be just legal, but also *large* (in a sense to be made precise later).

❑ In the general case, the admissible region and the safe-zones will be subsets of higher-dimensional Euclidean space.

We're scraping the surface of the *Distributed Monitoring Problem*

- Slightly modifying the problem makes it far harder… e.g. we want to monitor *uniformity*
- Three machines, loads are

$$L_1, L_2, L_3 \ \ (\overline{L} = \text{average load})$$

- Want to submit an alert when

$$\left(\overline{L} - L_1\right)^2 + \left(\overline{L} - L_2\right)^2 + \left(\overline{L} - L_3\right)^2 \geq T$$

- *Local conditions* for submitting an alert (i.e. safe-zones) are harder to define!
- Loads at all machines may be increasing but in a uniform fashion, hence no alert should be sent, etc.
- A huge range of problems…
- Simplest – average (linear function) of two scalar parameters.
- Most general and difficult – many nodes, each holds a (dynamic) vector of parameters, need to monitor a general function of them all. The value of this function may indicate a problem, an abnormality, or a phase change.

- No general solution yet.
- Required: a paradigm for compiling *local* conditions at the nodes, such that:
  - Every *global* event is captured (i.e. it results in the violation of at least one local condition).
  - Communication is minimal.
- Why? Save energy, overcome the communication bottleneck, maintain privacy…

Problem Definition – Streams

- A set of data sources
  - Distributed
  - Dynamic
- A data vector is collected from each stream
- Given:
  - A function over the union of data vectors
  - A threshold *T*
- Continuous query: alert when the GLOBAL function crosses *T*
- Goal: minimize communication during query execution

Problem Model – Monitored Function

- Need to define a problem which is more general than what was done so far (mostly, linear/monotonic functions, aggregates).

- But also a problem which is tractable and relevant to practical applications.

- A satisfactory choice is $$f\left(\frac{x_1+\ldots+x_k}{k}\right)$$

f a general function,

x_i the data vectors at the nodes

Problem Model – Monitored Function

- Broad enough to cover many interesting problems, including maximum, top-*k*, variance, effective dimension, mutual information, chi-square (local vectors = contingency tables).
- Maximum: augment local vectors by their powers and use the fact that

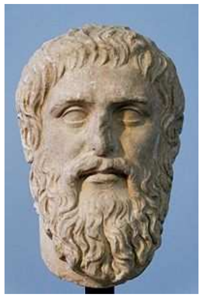
$$\max\{x_i\} \cong \left(x_1^n + \ldots + x_k^n\right)^{\frac{1}{n}}$$

- Variance: augment vectors by the squares of their components.

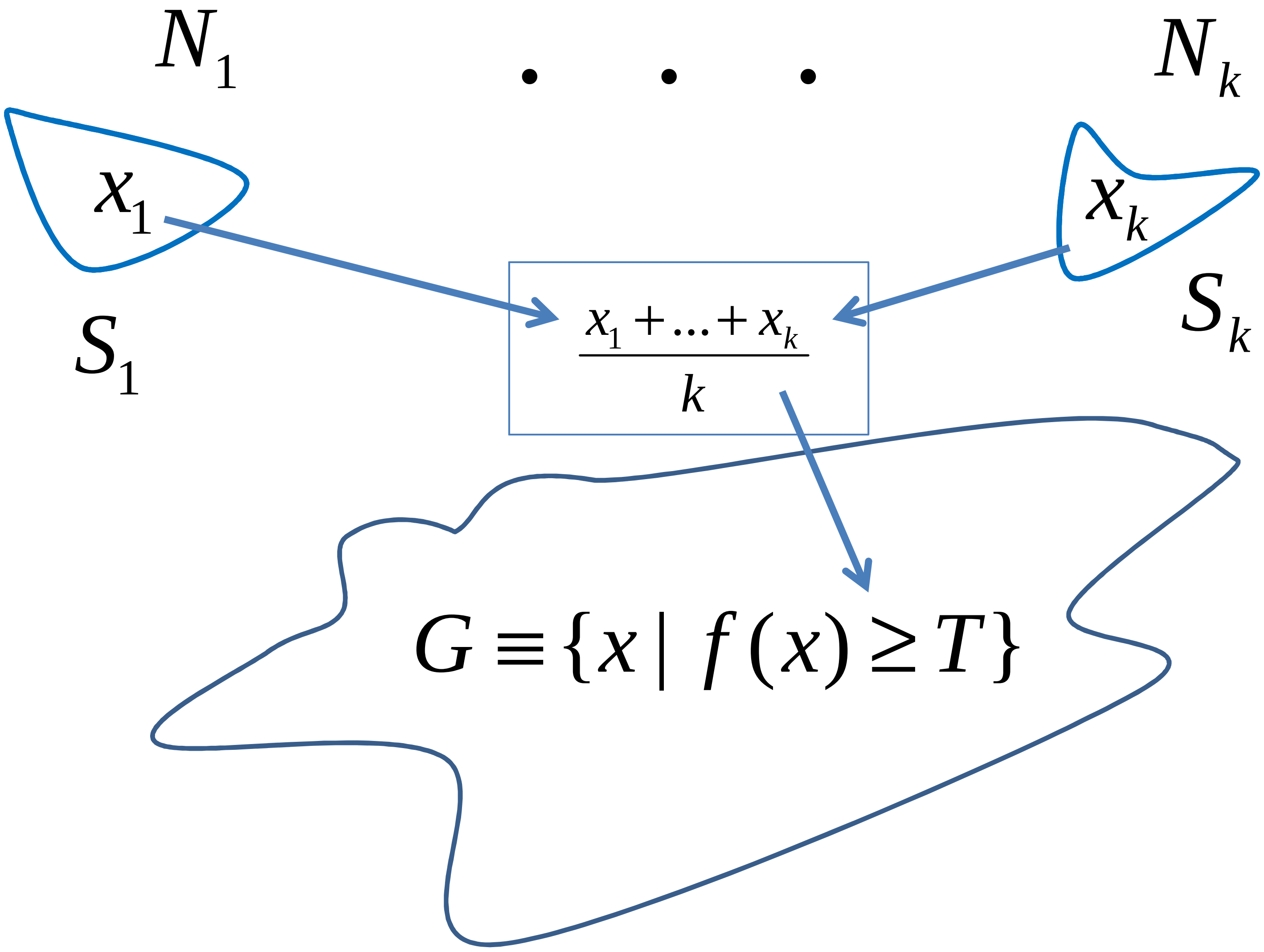
New Approach – Based on Geometry

ΑΓΕΩΜΕΤΡΗΤΟΣ ΜΗΔΕΙΣ ΕΙΣΙΤΩ

- “Let no one ignorant of geometry enter!”

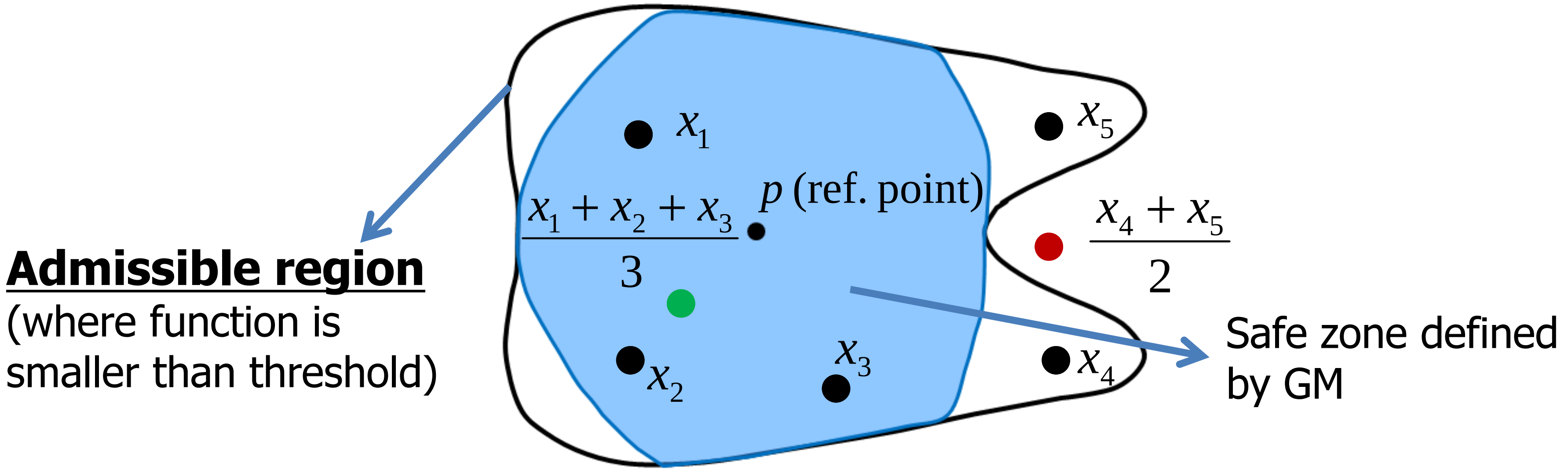


- We argue that for general functions, one must monitor the *domain* of the function and not its *range.*

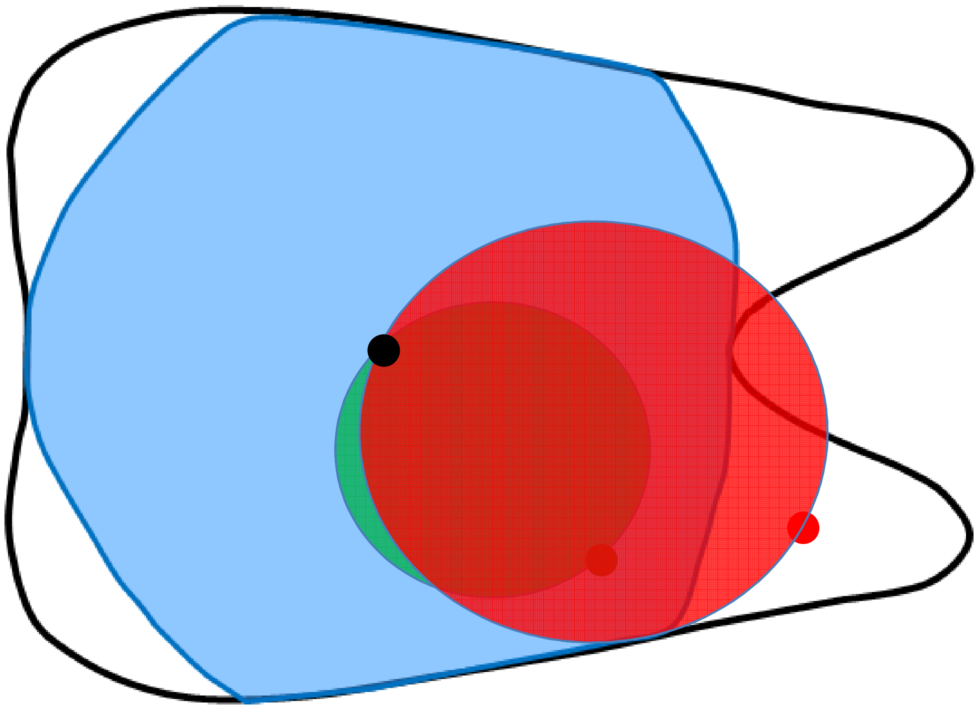

N_1
. . .
N_k
x_1
x_k
S_1
S_k
$\frac{x_1+\ldots+x_k}{k}$
$G \equiv \{x \mid f(x) \geq T\}$

Convexity

- The spherical constraints (previous work, Geometric Monitoring, “GM”) define regions – “safe zones” – in which the local vectors can roam freely (no alerts required). It is guaranteed that as long as the local vectors are in their regions, the function computed at their global average did not cross the threshold.
- **These regions turn out to be *convex* – and vice-versa: it is easy to see that any convex subset of the admissible region will do the job (since it is closed under averages).**

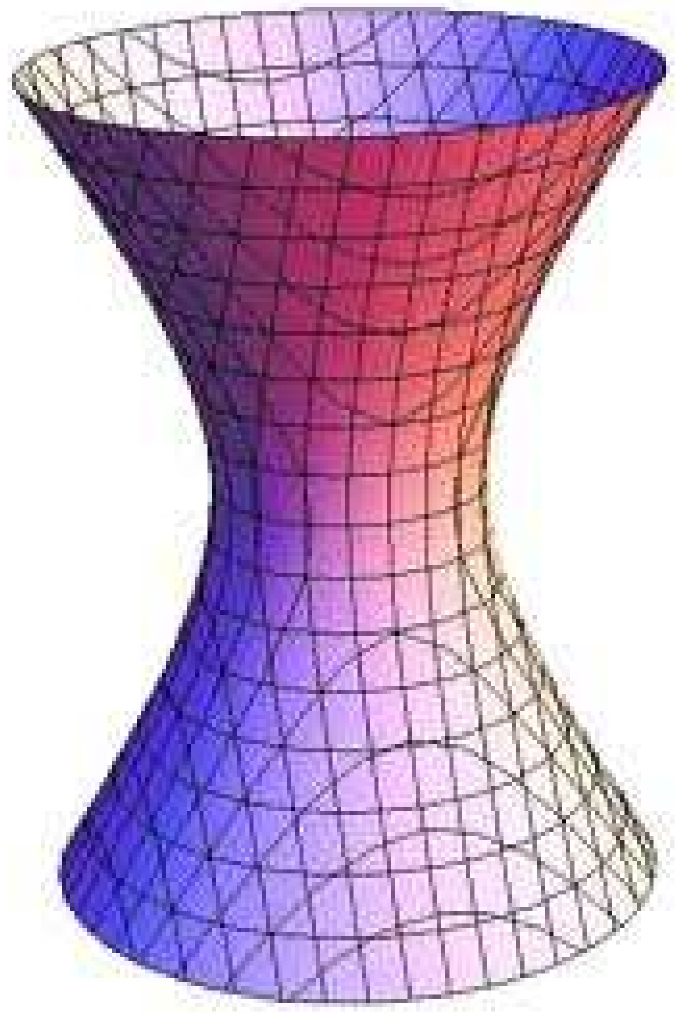


How “safe zones” are defined in GM: bounding spheres



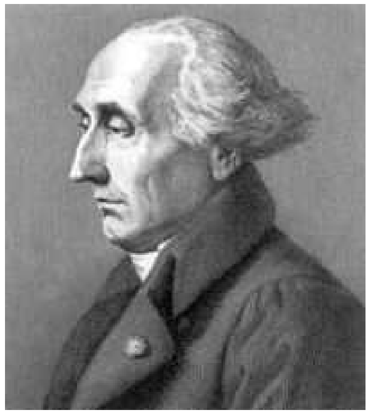
So – why not look for an *optimal* convex region?

- Very difficult (infinite dimensional, non-linear) optimization problem: find a maximal (in the probabilistic sense) convex subset.
- But – can sometimes find sets which are provably (and/or practically) better than the geometric method!
- Nasty (alas important) example: inner product. Admissible region is a hyperboloid – every boundary point is non-convex and non-concave.

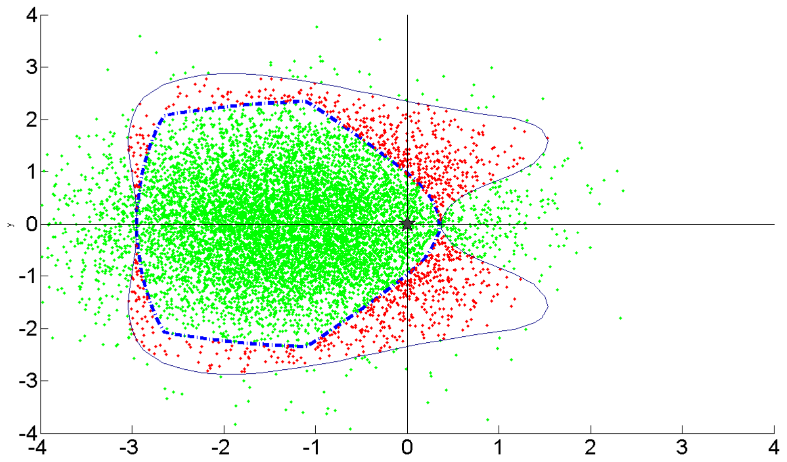


- A more general approach, requiring additional machinery (optimization, probability, algorithms).

"As long as algebra and geometry have been separated, their progress have been slow and their uses limited, but when these two sciences have been united, they have lent each mutual forces, and have marched together towards perfection."

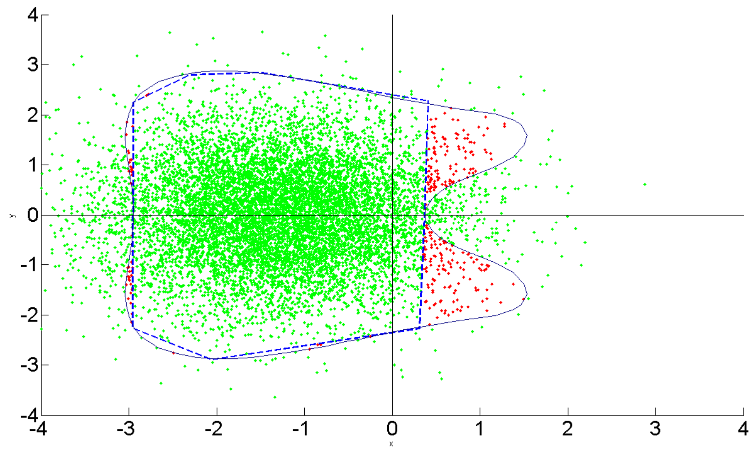


geometric method



(TKDE, 2012)

optimal convex octagon



GENERAL SOLUTION

- So far, we assumed all nodes have the **same distribution**, and therefore they are all assigned the **same** safe-zone.
- General solution – assign each node its own safe-zone, such that:

i. Each node's safe-zone fits its data distribution.

ii. The average of vectors from the distinct safe-zones satisfies the legality constraint (it lies inside the admissible region).

iii. Maximize the time during which the local vectors remain in their safe-zones.

Minkowski sum:

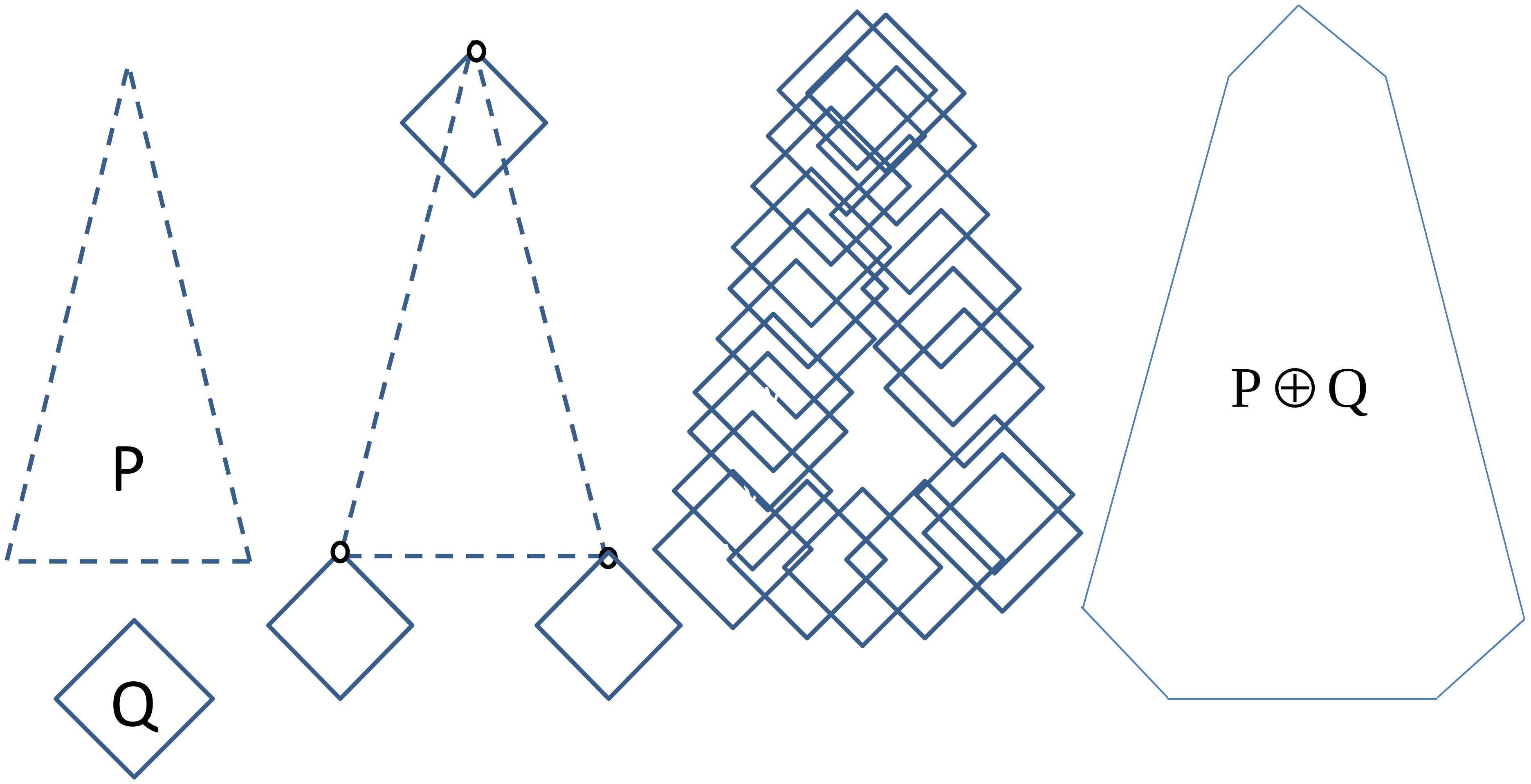
$$A \oplus B = \{a + b \mid a \in A, b \in B\}$$

Optimization problem – optimal safe-zones:

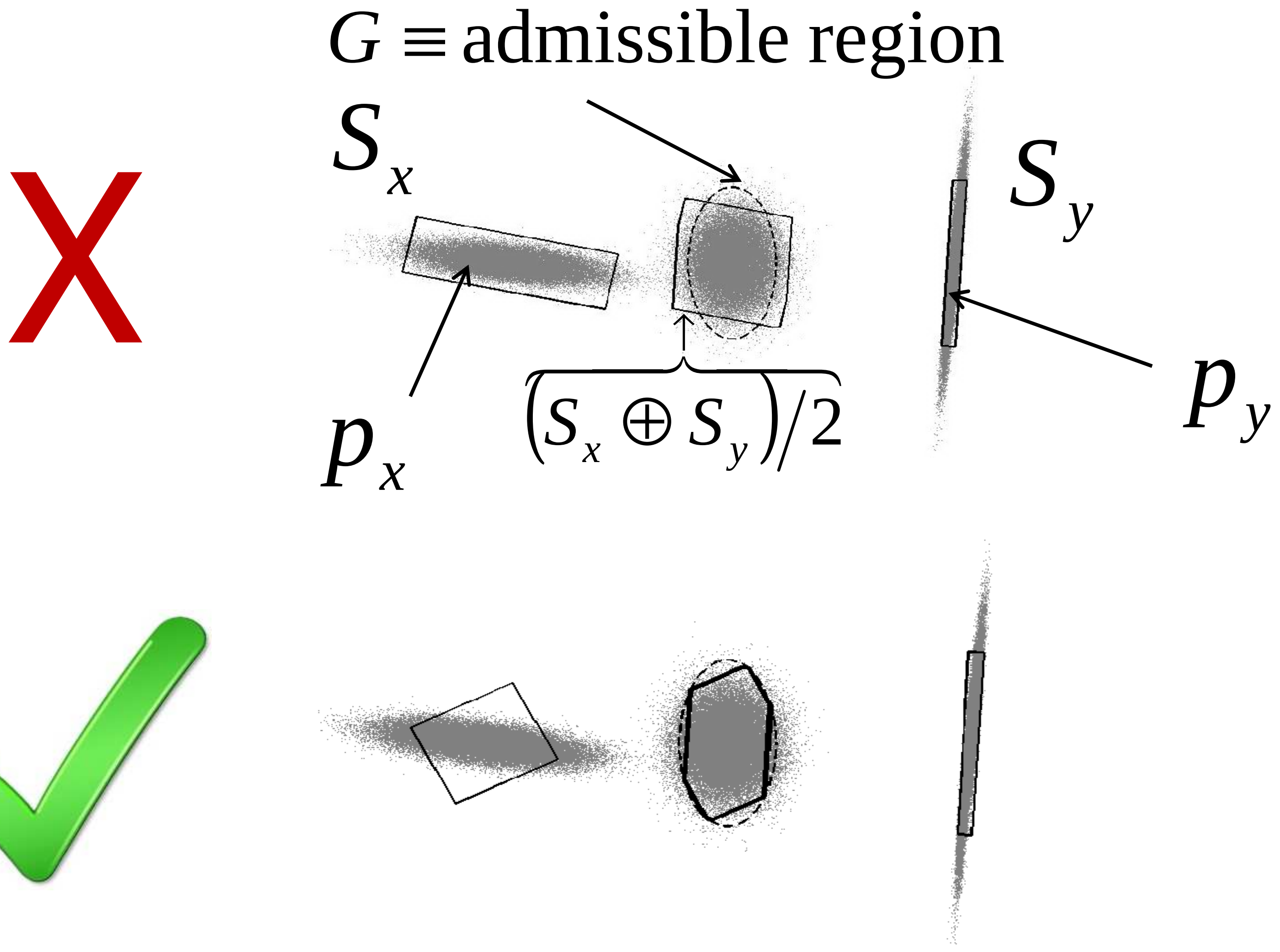
Maximize $\int_{S_x} p_x dx \int_{S_y} p_y dy$ subject to $\frac{S_x \oplus S_y}{2} \subset G$

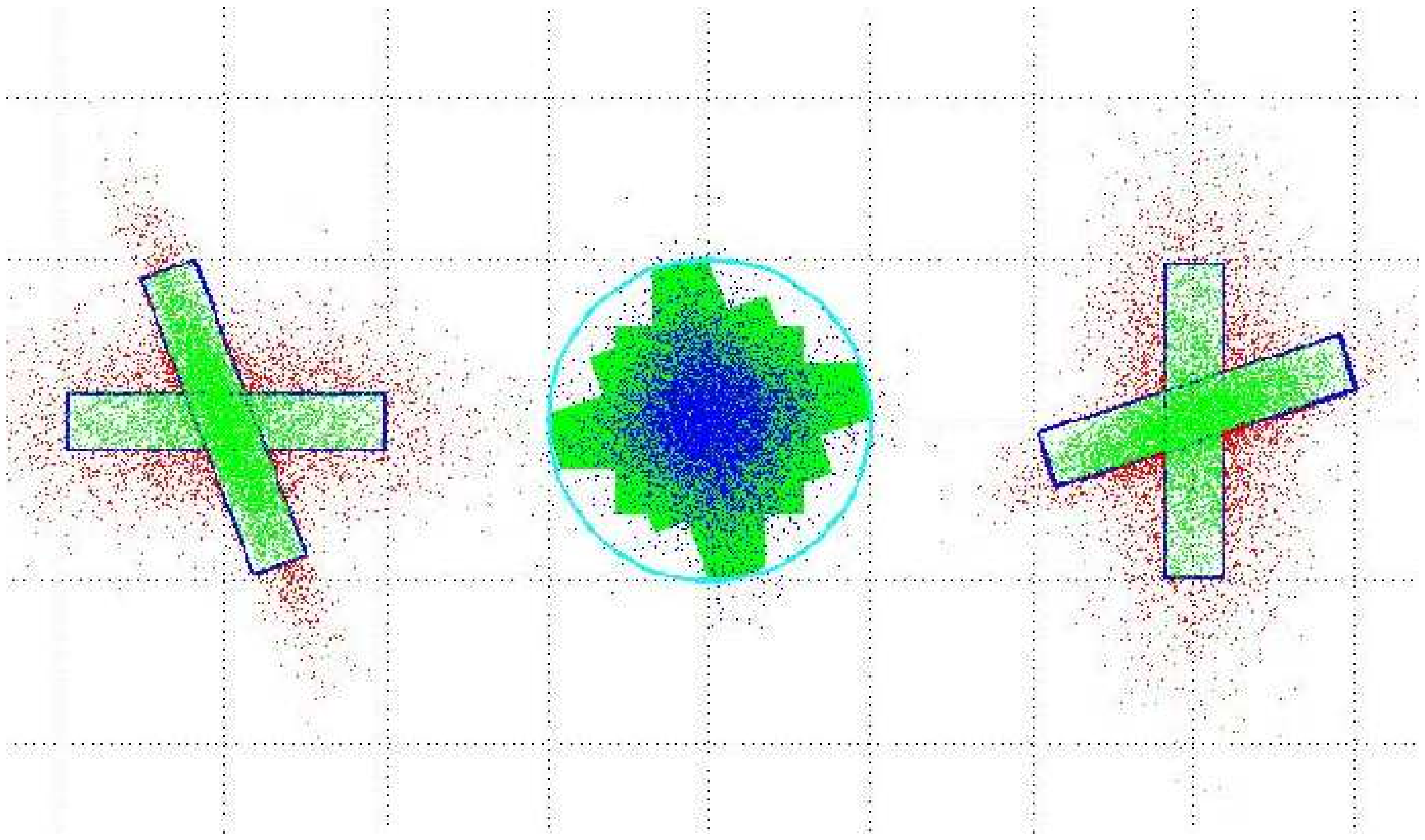
$p_x, p_y =$ p.d.f's at the nodes, $S_x, S_y =$ safe zones

Minkowski sum – the geometric intuition



To construct the Minkowski sum of P and Q, move a copy of Q all over P and take the union.


$G \equiv$ admissible region
S_x
S_y
p_x
p_y
$(S_x \oplus S_y)/2$



Geometric method vs. Safe-Zones: example

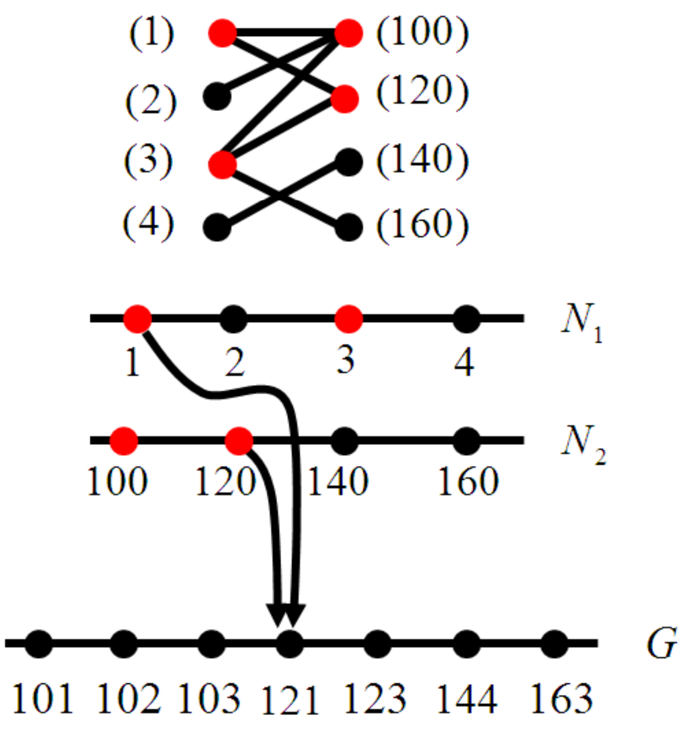
- ❑ Very simple case: two nodes, *d*-dimensional data
- ❑ Node 1 with a p.d.f = uniform over a ball with radius $1+\varepsilon$, Node 2 with the same but radius $1-\varepsilon$.
- ❑ *G* is a ball with radius 1.
- ❑ No alerts with general safe-zones.
- ❑ With the geometric method, each node cannot be assigned a safe-zone larger than *G*, hence Node 1 will submit an alert with probability at least $1-\left(1/(1+\varepsilon)\right)^d \approx 1-\exp(-\varepsilon d)$.
- ❑ This suggests that the improvement of the general approach over the geometric method increases with the dimension – and this is borne out in the experiments.

Geometric method vs. Safe-Zones – cont.

❑ The safe-zone approach is theoretically better – seeks a general solution which satisfies an optimization criteria.
❑ Testing the local conditions is typically easier with the safe-zone approach.
❑ The geometric method assigns the same constraints at all nodes, hence it cannot adapt when data distributions at the distinct nodes differ.
❑ Safe-zones can be *larger* than the admissible region – impossible, of course, with the geometric method.

BUT –
❑ **The geometric method requires no optimization.**

- **The safe-zone problem is NP-hard even for the simplest case: two nodes, one-dimensional data (reducible to *maximum edge biclique*):**



NP-hard in other cases – even one-dimensional data and *interval* safe-zones; two nodes and convex safe-zones (for dimension > 3).

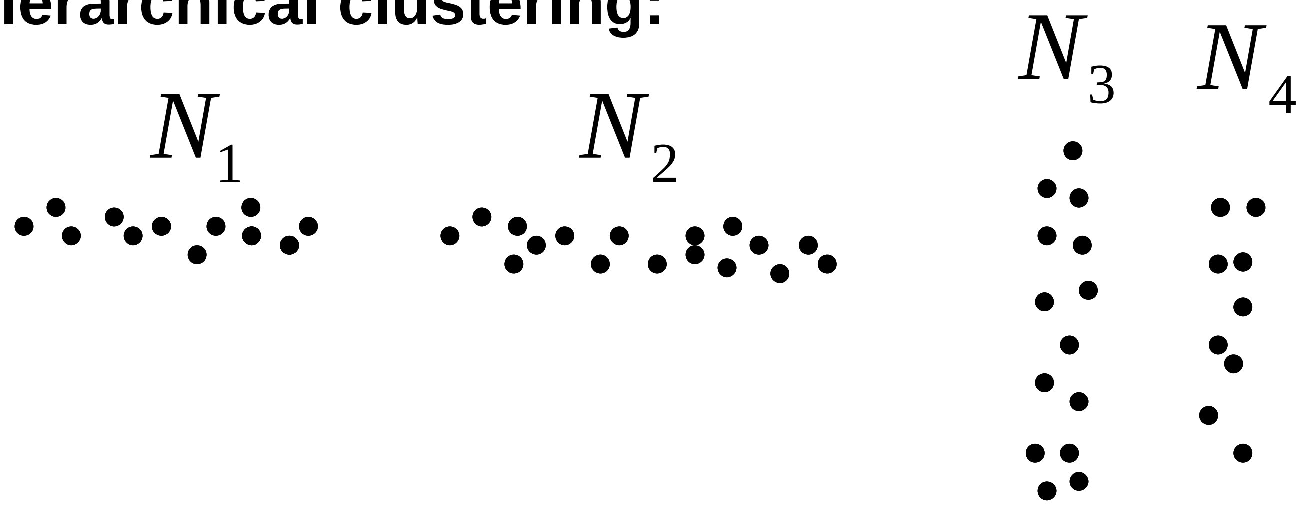
Making it work:

- ❑ Hierarchical clustering of nodes.

- ❑ Computational techniques to quickly test the constraints and compute the target function.

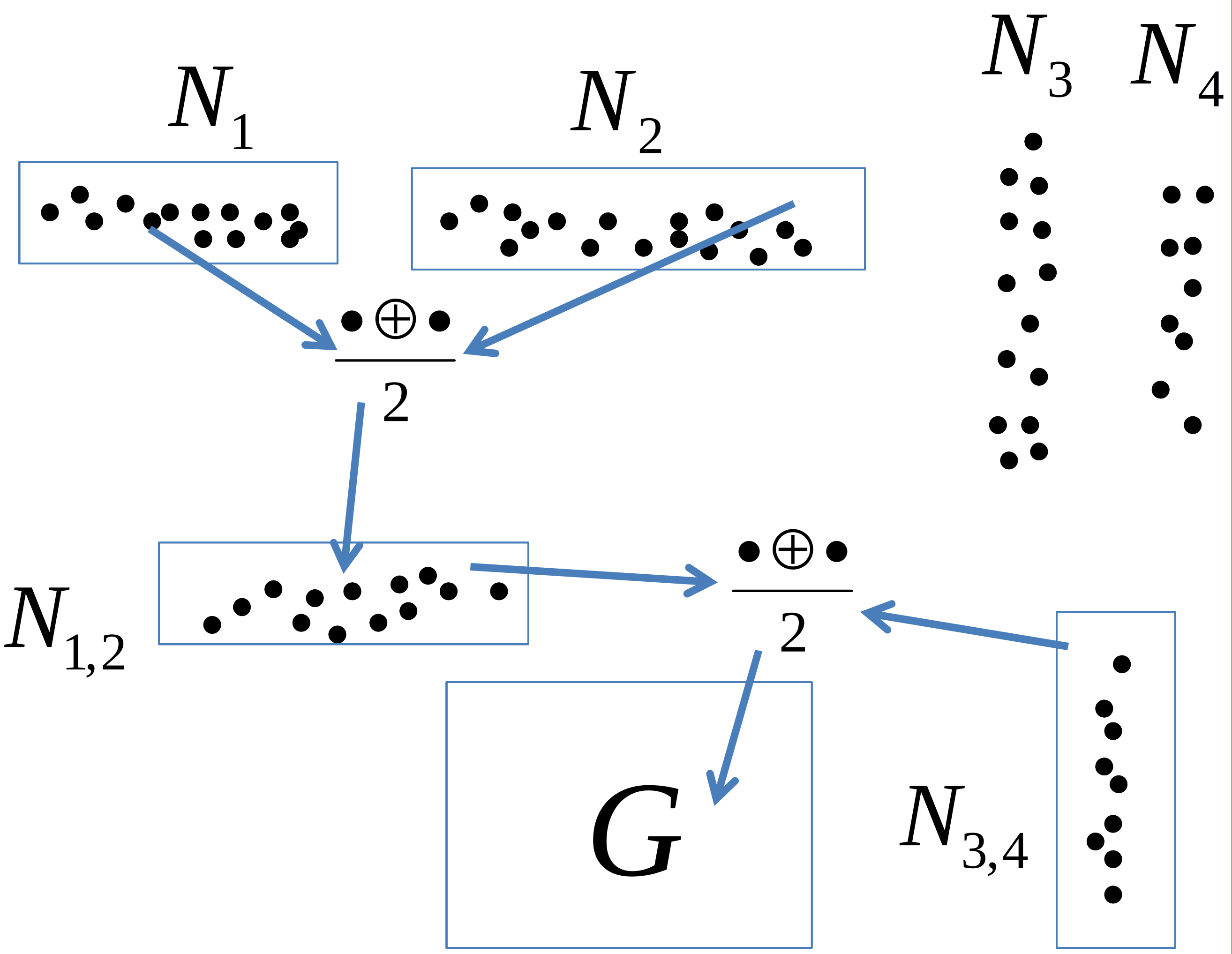
- ❑ Practically it boils down to solving an optimization problem over *shapes,* e.g. the location of the vertices which define a polytope.

Hierarchical clustering:



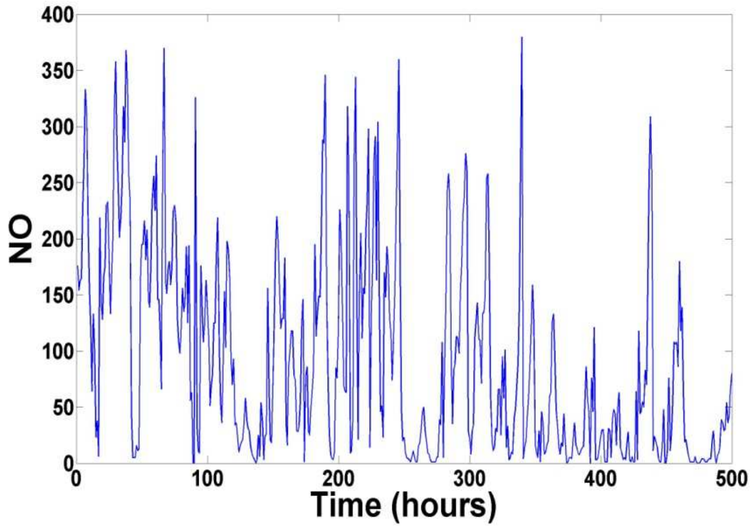
❑ Fitting rectangular safe-zones: optimize over 16 parameters.

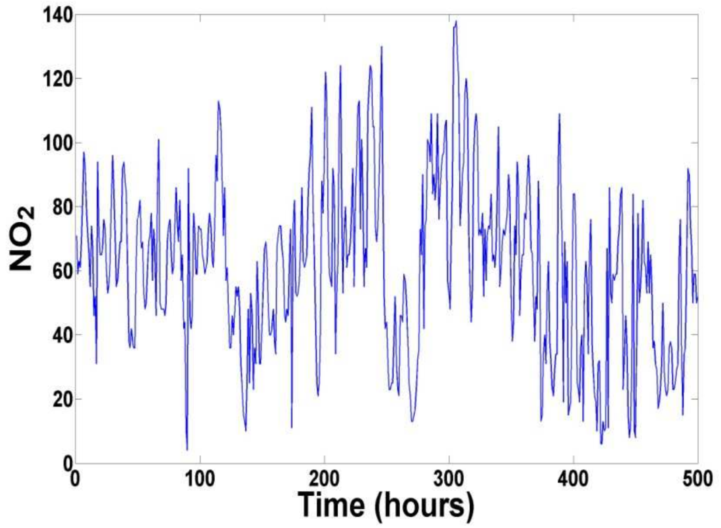
❑ Instead – model N_1 , N_2 by a single "super-node", same for N_3 , N_4; fit rectangles to these "super-nodes" and then fit rectangles to the data at the original nodes by using the "super-node" rectangles as admissible regions.

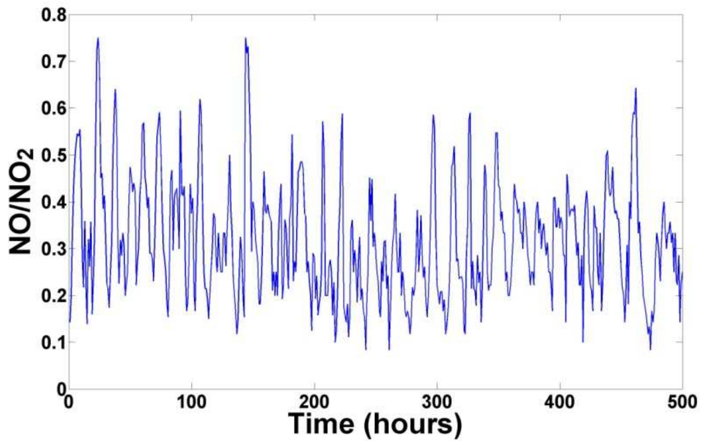

N_1
N_2
N_3
N_4
$\frac{\bullet \oplus \bullet}{2}$
$N_{1,2}$
$\frac{\bullet \oplus \bullet}{2}$
$N_{3,4}$
G

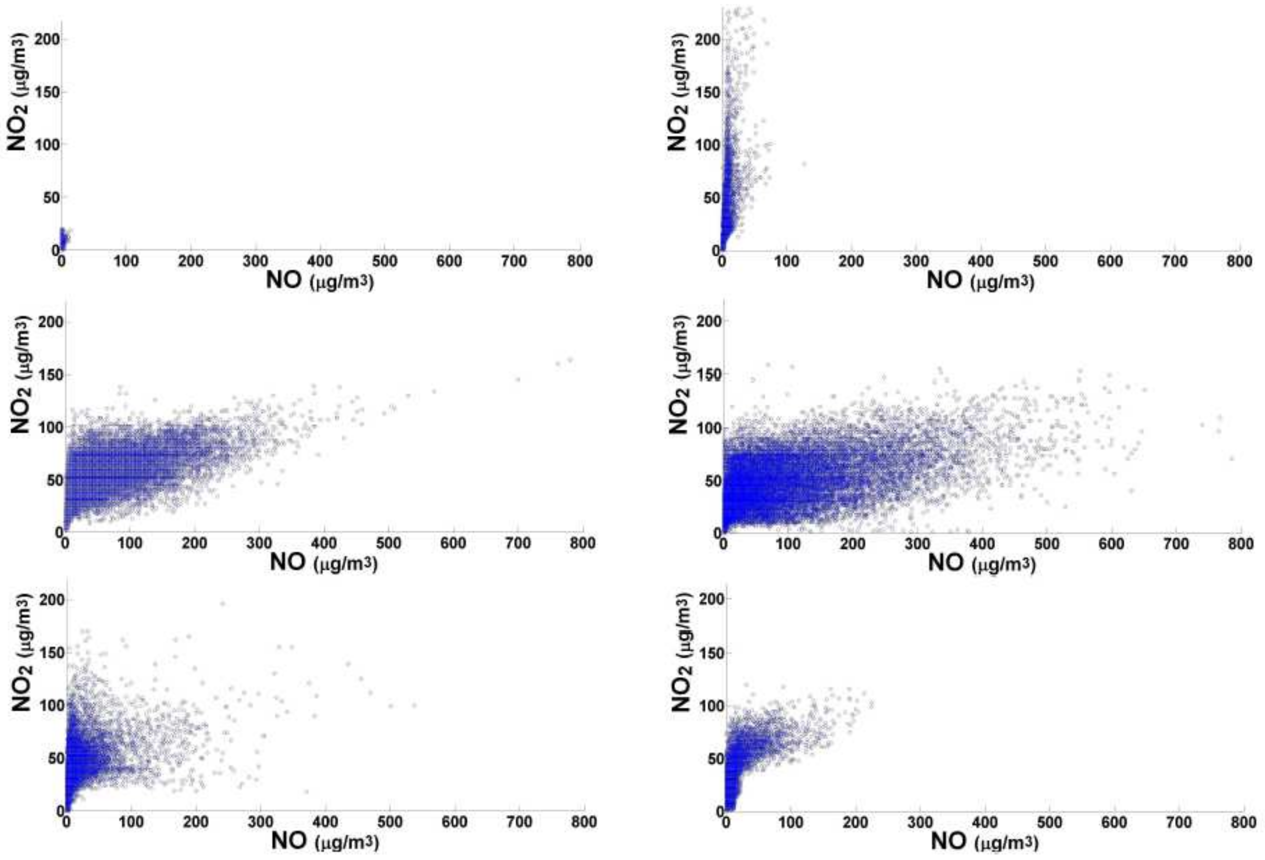
- ❑ **What did we gain?**
- ❑ **Solve more problems, but with a smaller number of parameters!**
- ❑ **Saves a great deal of time and memory, since the complexity of complicated optimization problems increases super-linearly with the number of parameters.**

DATA

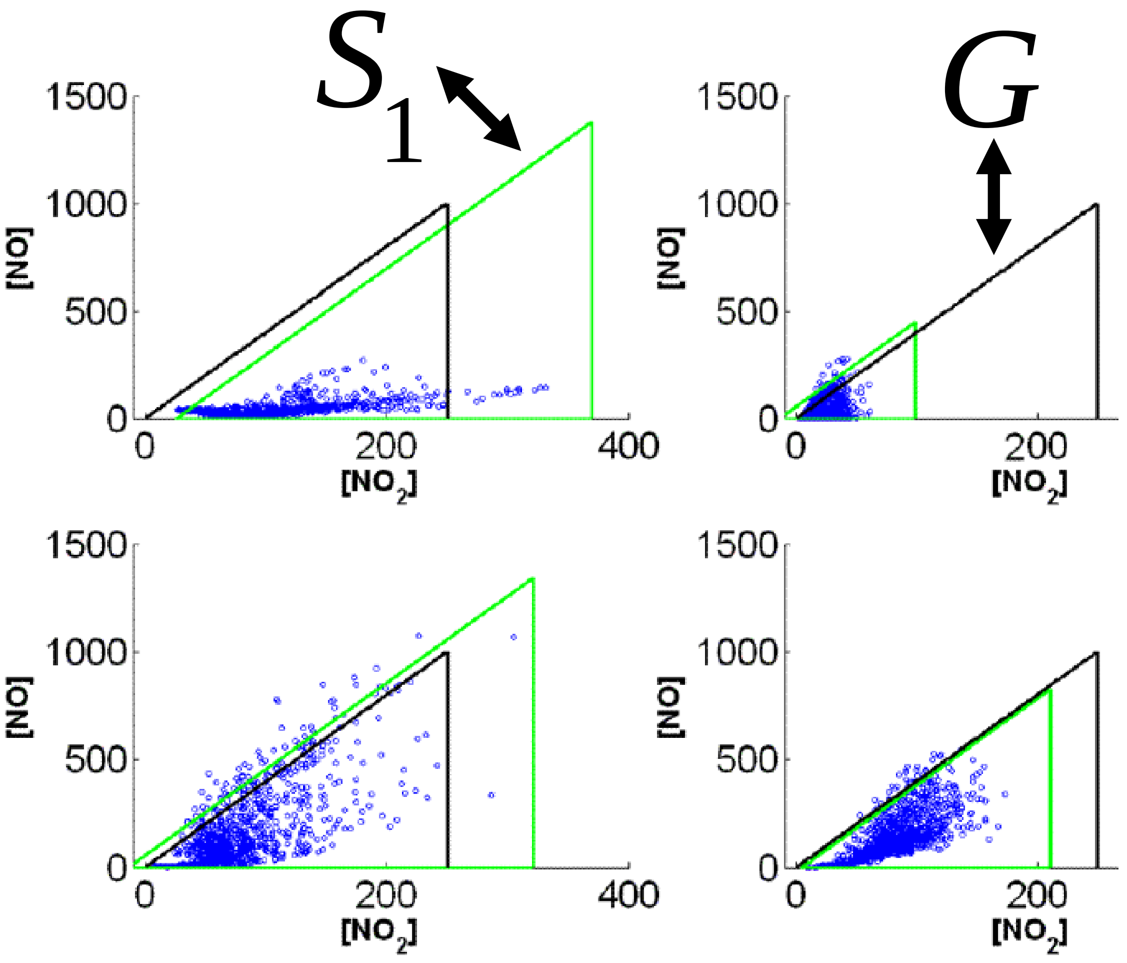




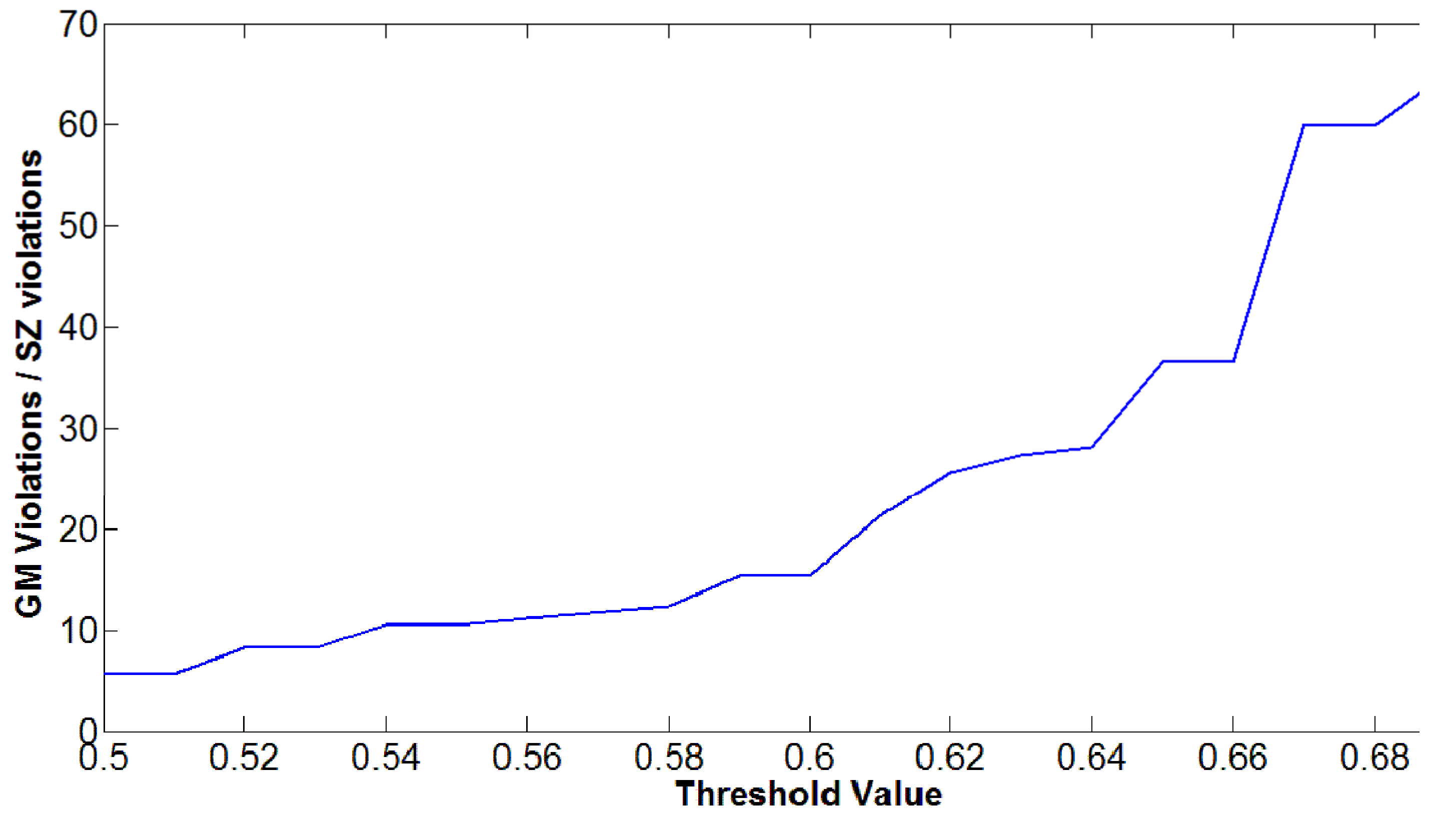




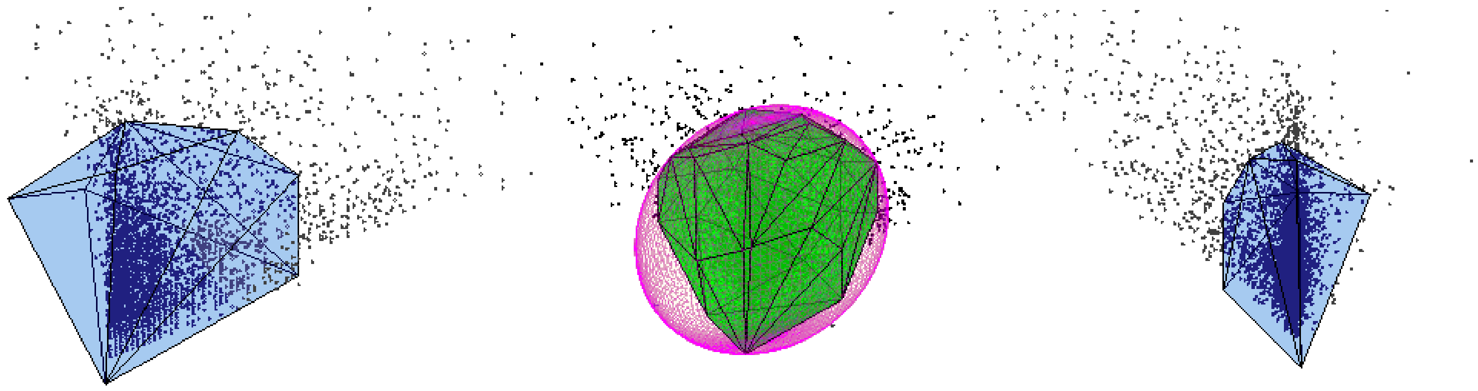
clustering nodes



example with triangular safe-zones

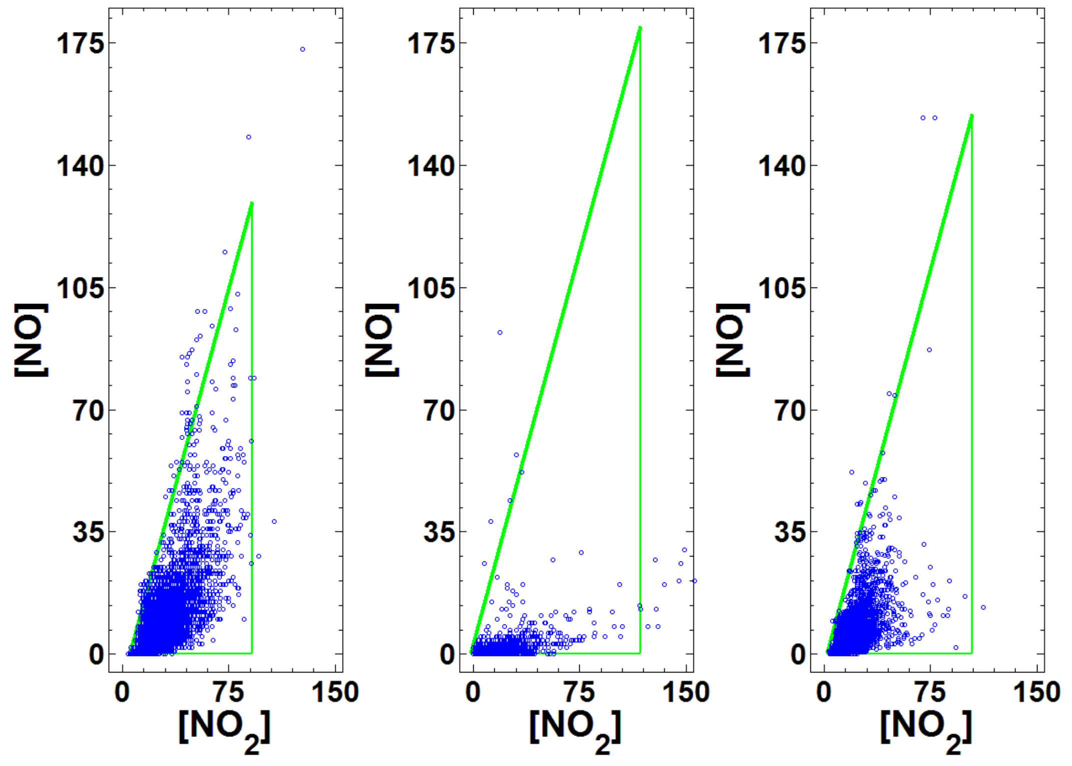


improvement over geometric method for chi-square monitoring



3D data and safe-zones

❑ Violation recovery – find optimal pairs of nodes which “balance” each other.


[NO]
$[NO_2]$