Graph Classes

www.graphclasses.org

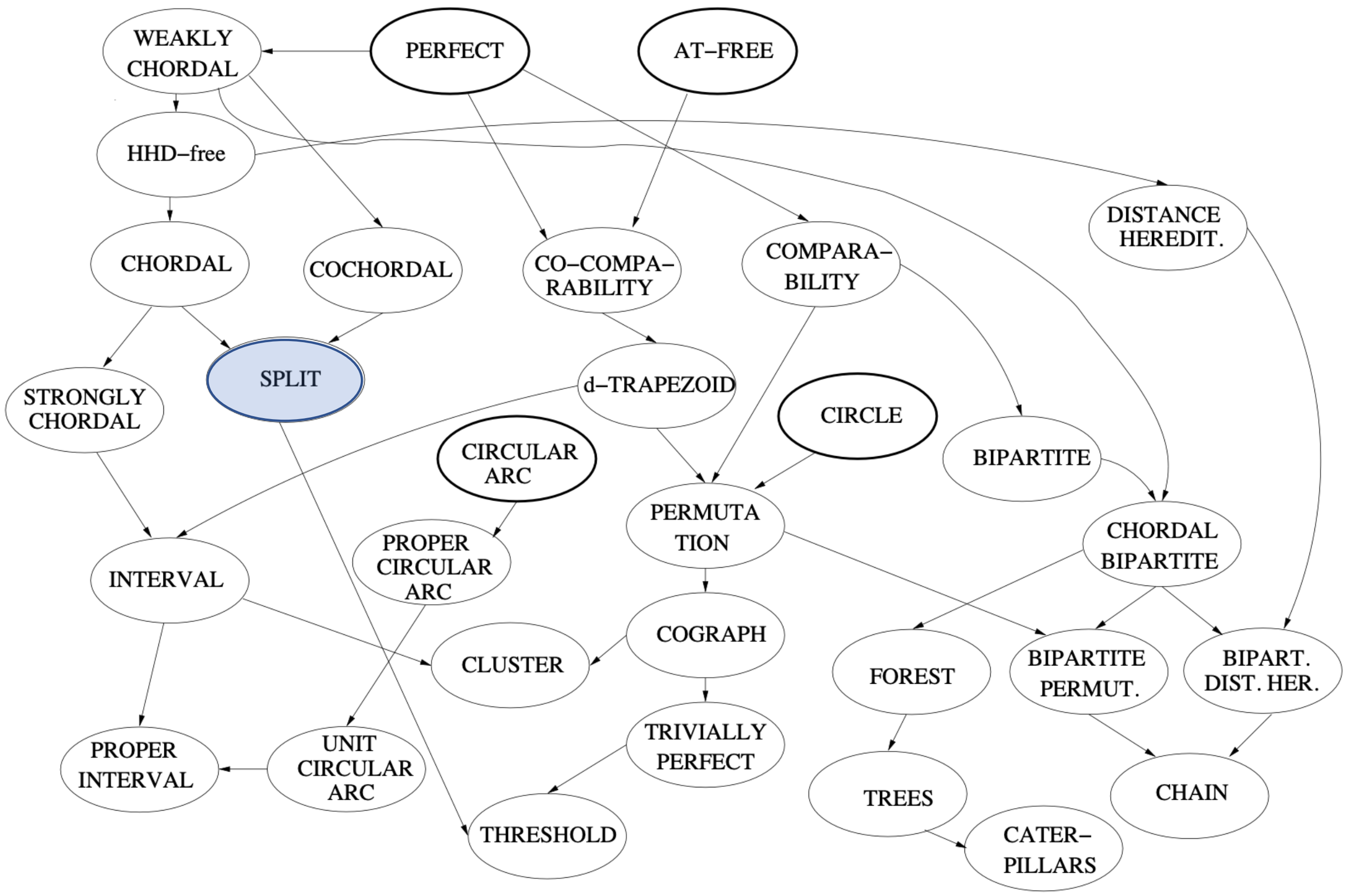
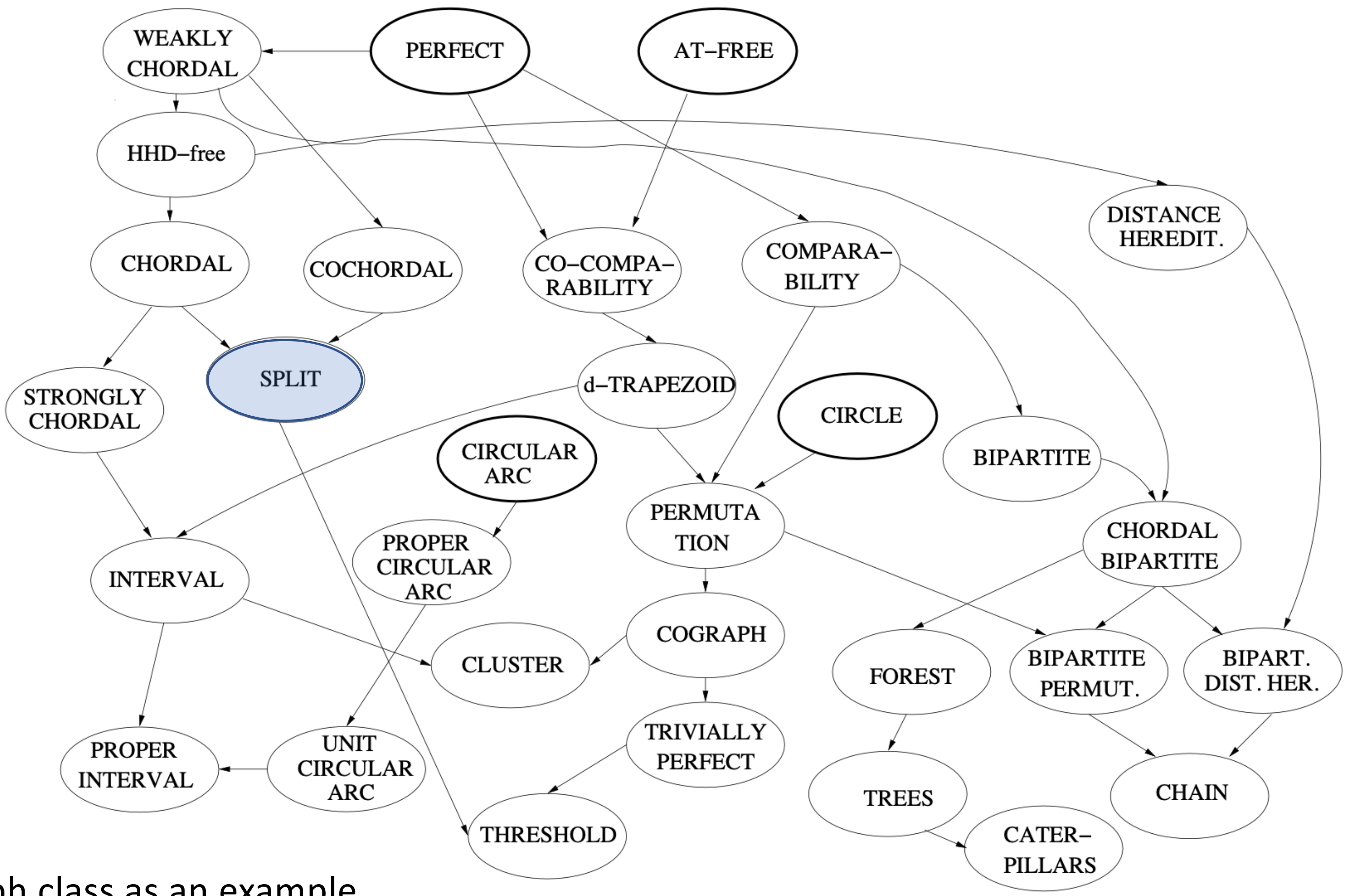


Figure 2.2: This diagram represents the inclusion relations among the graph classes we described in Chapter 2.1. An arrow from class A to class B means that $B \subset A$, and if there are arrows from two classes A and B to a class C, it means that $C \subseteq A \cap B$. The thicker ovals, represents the classes that are not perfect.

From Mancini thesis – this author prefers arrows drawn downwards rather than upwards as I use in Hasse Diagrams



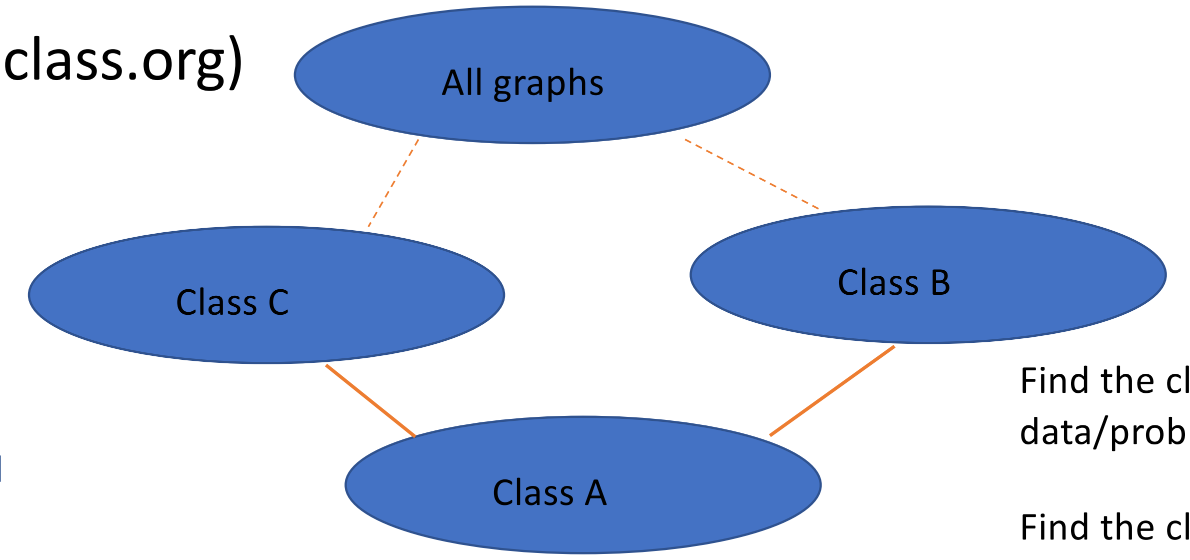
From Mancini thesis

Take split graph class as an example.
Tracing upwards, we can see every split graph is a chordal graph – but not visa versa. Continuing up, every chordal graph is a perfect graph…..etc.
Going downwards – we can see that *some* split graphs are threshold graphs (have extra properties that the “other” split graphs don’t have. Of course, all threshold graphs are split graphs.
The above is a TINY part of the of “atlas” of graph classes.

Graphs (and slightly less so, Hypergraphs) are so well studied…they provide a "ready" taxonomy/structure to a huge array of problems.

Graph taxonomy

(e.g., graphclass.org)



Class A is smaller than class C or B −

But inherits all the good properties of C or B +

Plus, may have extra good properties because of the specialized nature of this class +

Find the class that fits all your data/problem instance.

Find the class(es) that nearly fits all your data/problem instances

Find the class(es) that approximately fits some data instances and is VERY NICE

Graphclass: perfect

Definition:

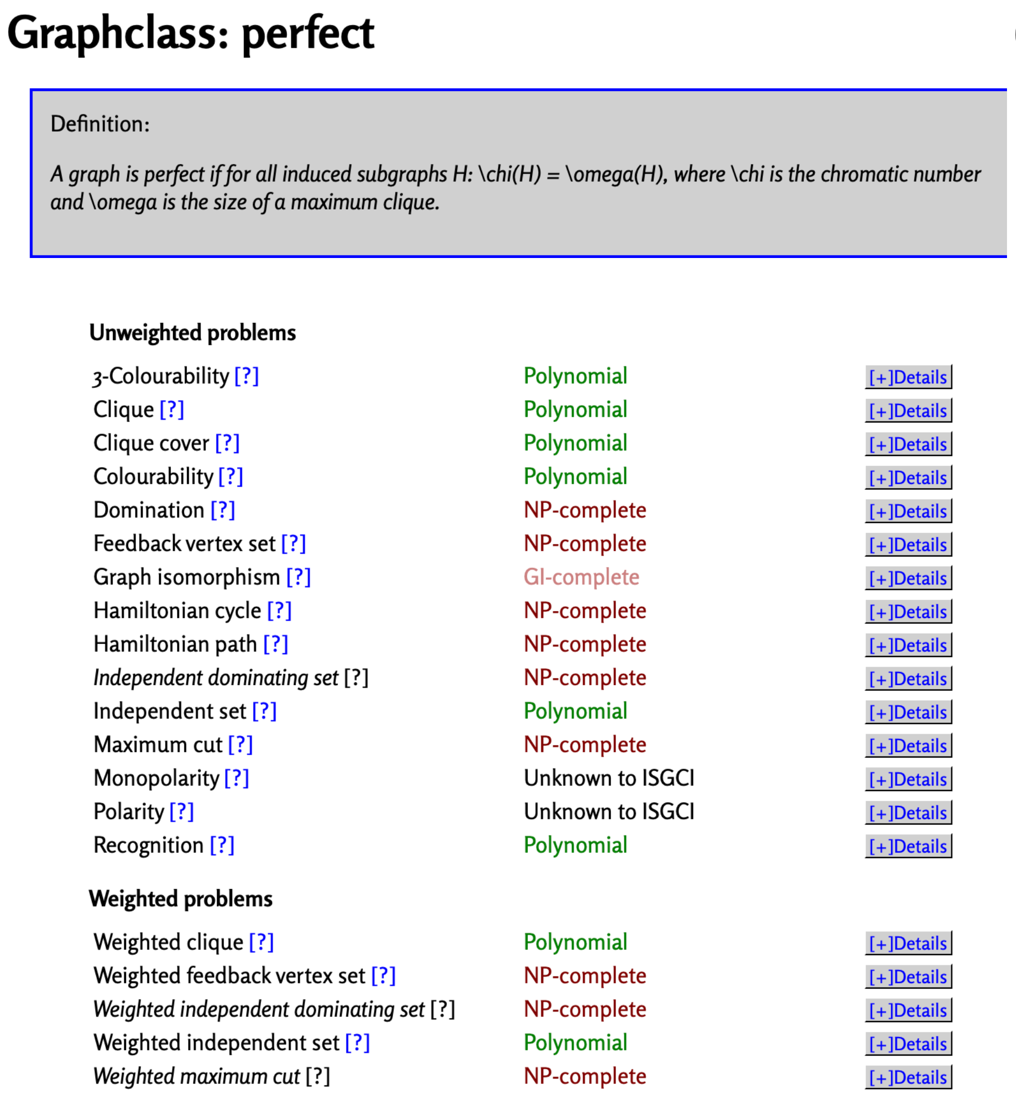
A graph is perfect if for all induced subgraphs H: \chi(H) = \omega(H), where \chi is the chromatic number and \omega is the size of a maximum clique.

Unweighted problems

3-Colourability [?]	Polynomial	[+]Details
Clique [?]	Polynomial	[+]Details
Clique cover [?]	Polynomial	[+]Details
Colourability [?]	Polynomial	[+]Details
Domination [?]	NP-complete	[+]Details
Feedback vertex set [?]	NP-complete	[+]Details
Graph isomorphism [?]	GI-complete	[+]Details
Hamiltonian cycle [?]	NP-complete	[+]Details
Hamiltonian path [?]	NP-complete	[+]Details
Independent dominating set [?]	NP-complete	[+]Details
Independent set [?]	Polynomial	[+]Details
Maximum cut [?]	NP-complete	[+]Details
Monopolarity [?]	Unknown to ISGCI	[+]Details
Polarity [?]	Unknown to ISGCI	[+]Details
Recognition [?]	Polynomial	[+]Details

Weighted problems

Weighted clique [?]	Polynomial	[+]Details
Weighted feedback vertex set [?]	NP-complete	[+]Details
Weighted independent dominating set [?]	NP-complete	[+]Details
Weighted independent set [?]	Polynomial	[+]Details
Weighted maximum cut [?]	NP-complete	[+]Details



Graphclass: perfect

Definition:

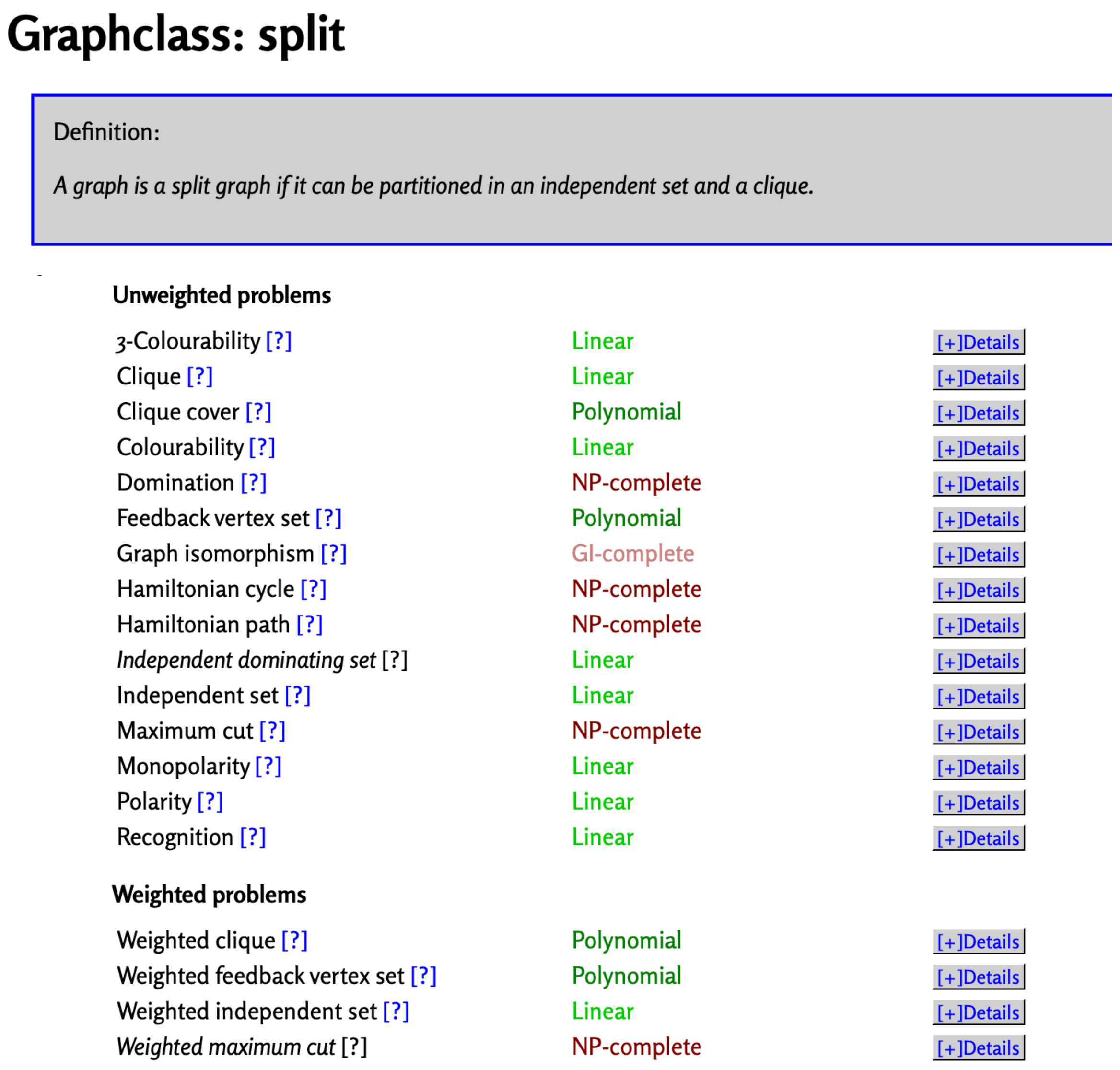
A graph is perfect if for all induced subgraphs H: \chi(H) = \omega(H), where \chi is the chromatic number and \omega is the size of a maximum clique.

Unweighted problems

3-Colourability [?]	Polynomial	[+]Details
Clique [?]	Polynomial	[+]Details
Clique cover [?]	Polynomial	[+]Details
Colourability [?]	Polynomial	[+]Details
Domination [?]	NP-complete	[+]Details
Feedback vertex set [?]	NP-complete	[+]Details
Graph isomorphism [?]	GI-complete	[+]Details
Hamiltonian cycle [?]	NP-complete	[+]Details
Hamiltonian path [?]	NP-complete	[+]Details
Independent dominating set [?]	NP-complete	[+]Details
Independent set [?]	Polynomial	[+]Details
Maximum cut [?]	NP-complete	[+]Details
Monopolarity [?]	Unknown to ISGCI	[+]Details
Polarity [?]	Unknown to ISGCI	[+]Details
Recognition [?]	Polynomial	[+]Details

Weighted problems

Weighted clique [?]	Polynomial	[+]Details
Weighted feedback vertex set [?]	NP-complete	[+]Details
Weighted independent dominating set [?]	NP-complete	[+]Details
Weighted independent set [?]	Polynomial	[+]Details
Weighted maximum cut [?]	NP-complete	[+]Details



Graphclass: split

Definition:

A graph is a split graph if it can be partitioned in an independent set and a clique.

Unweighted problems

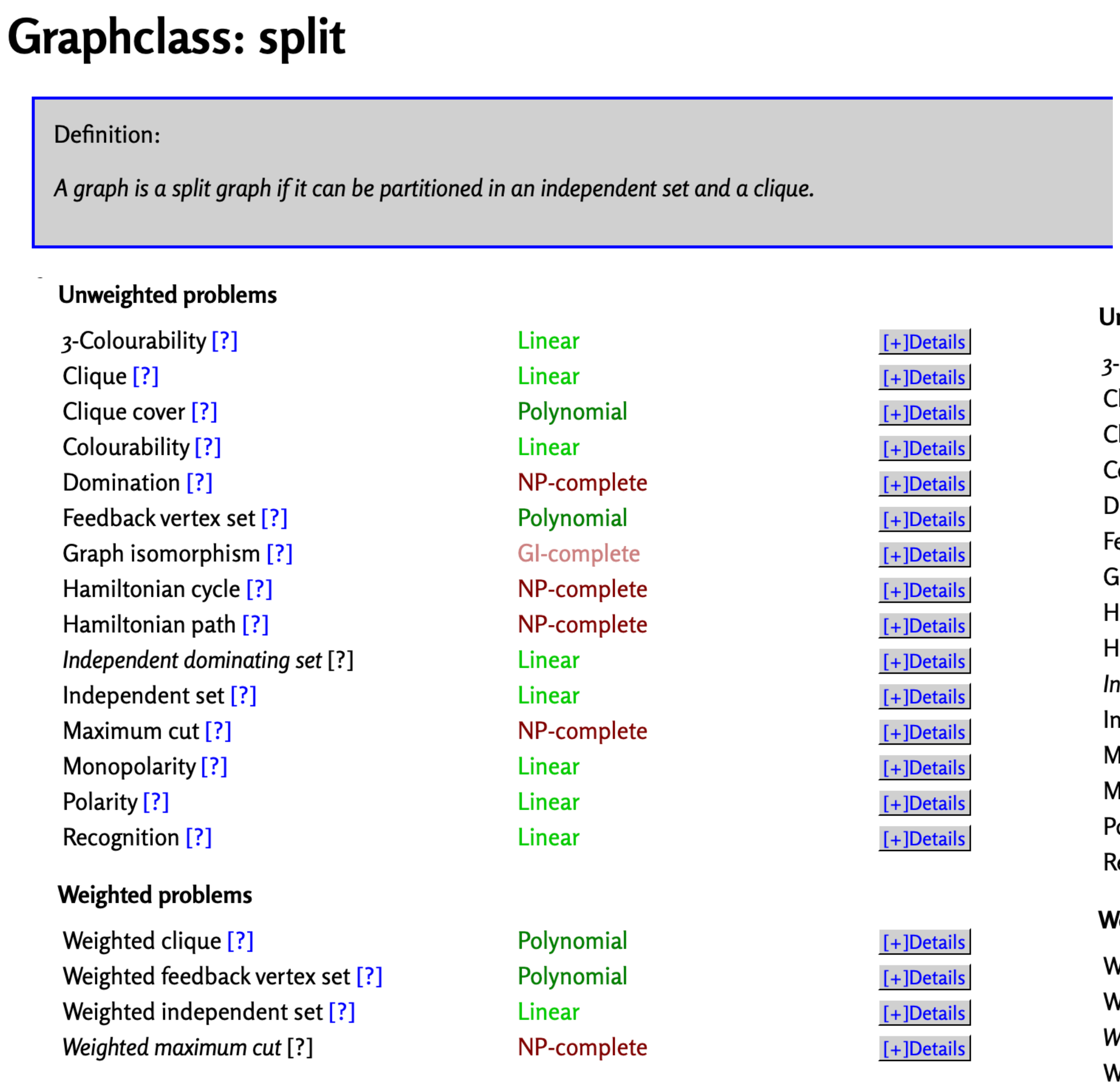
3-Colourability [?]	Linear	[+]Details
Clique [?]	Linear	[+]Details
Clique cover [?]	Polynomial	[+]Details
Colourability [?]	Linear	[+]Details
Domination [?]	NP-complete	[+]Details
Feedback vertex set [?]	Polynomial	[+]Details
Graph isomorphism [?]	GI-complete	[+]Details
Hamiltonian cycle [?]	NP-complete	[+]Details
Hamiltonian path [?]	NP-complete	[+]Details
Independent dominating set [?]	Linear	[+]Details
Independent set [?]	Linear	[+]Details
Maximum cut [?]	NP-complete	[+]Details
Monopolarity [?]	Linear	[+]Details
Polarity [?]	Linear	[+]Details
Recognition [?]	Linear	[+]Details

Weighted problems

Weighted clique [?]	Polynomial	[+]Details
Weighted feedback vertex set [?]	Polynomial	[+]Details
Weighted independent set [?]	Linear	[+]Details
Weighted maximum cut [?]	NP-complete	[+]Details

Wouldn’t it be nice if your data/instances were split graphs?

- Or some large number of your data instances…
- Or some identifiable and “important” set of your problem instances…
- Or if you’re a useful/important set of your data instances were close (in some sense) to split graphs…
- Well some of my interests are 🙂



Graphclass: split

Definition:

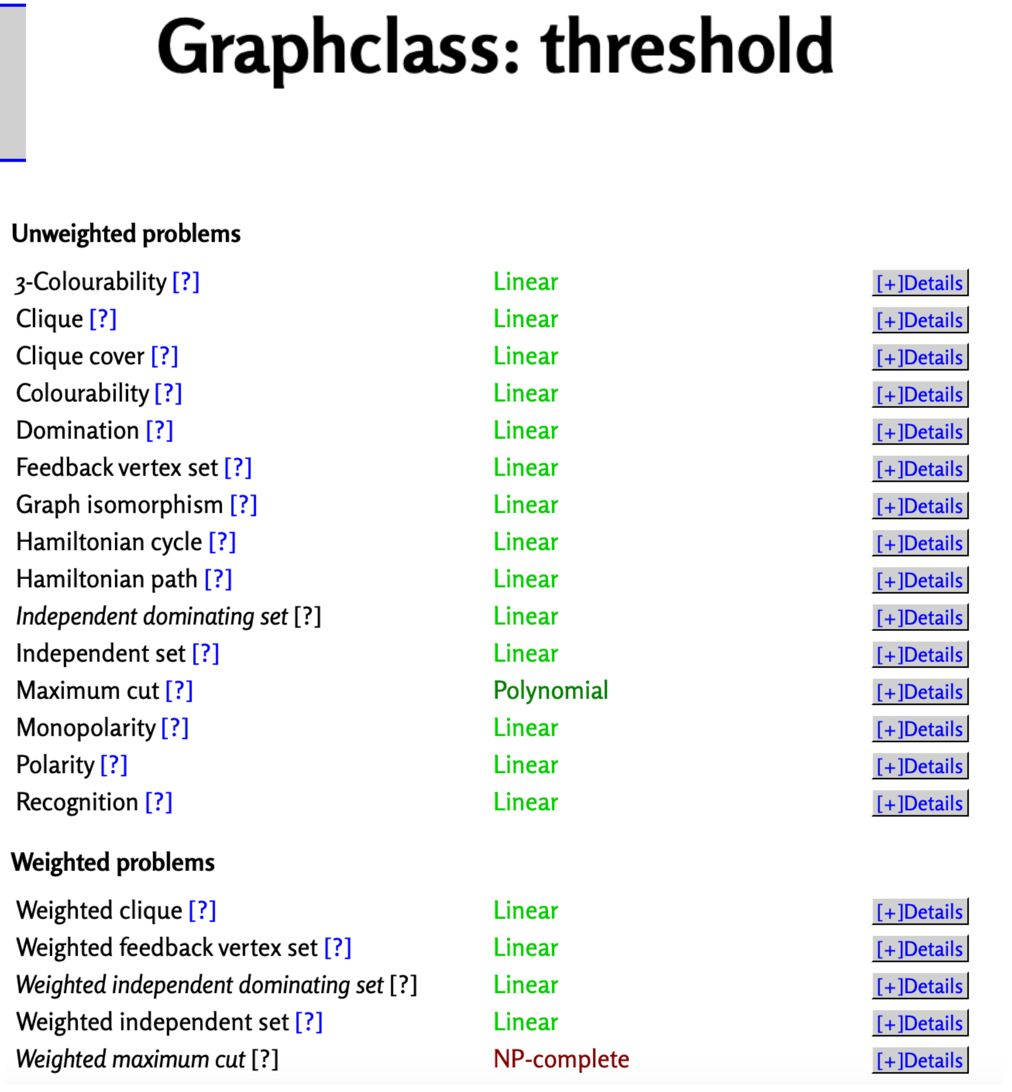
A graph is a split graph if it can be partitioned in an independent set and a clique.

Unweighted problems

3-Colourability [?]	Linear	[+]Details
Clique [?]	Linear	[+]Details
Clique cover [?]	Polynomial	[+]Details
Colourability [?]	Linear	[+]Details
Domination [?]	NP-complete	[+]Details
Feedback vertex set [?]	Polynomial	[+]Details
Graph isomorphism [?]	GI-complete	[+]Details
Hamiltonian cycle [?]	NP-complete	[+]Details
Hamiltonian path [?]	NP-complete	[+]Details
Independent dominating set [?]	Linear	[+]Details
Independent set [?]	Linear	[+]Details
Maximum cut [?]	NP-complete	[+]Details
Monopolarity [?]	Linear	[+]Details
Polarity [?]	Linear	[+]Details
Recognition [?]	Linear	[+]Details

Weighted problems

Weighted clique [?]	Polynomial	[+]Details
Weighted feedback vertex set [?]	Polynomial	[+]Details
Weighted independent set [?]	Linear	[+]Details
Weighted maximum cut [?]	NP-complete	[+]Details



Graphclass: threshold

Unweighted problems

3-Colourability [?]	Linear	[+]Details
Clique [?]	Linear	[+]Details
Clique cover [?]	Linear	[+]Details
Colourability [?]	Linear	[+]Details
Domination [?]	Linear	[+]Details
Feedback vertex set [?]	Linear	[+]Details
Graph isomorphism [?]	Linear	[+]Details
Hamiltonian cycle [?]	Linear	[+]Details
Hamiltonian path [?]	Linear	[+]Details
Independent dominating set [?]	Linear	[+]Details
Independent set [?]	Linear	[+]Details
Maximum cut [?]	Polynomial	[+]Details
Monopolarity [?]	Linear	[+]Details
Polarity [?]	Linear	[+]Details
Recognition [?]	Linear	[+]Details

Weighted problems

Weighted clique [?]	Linear	[+]Details
Weighted feedback vertex set [?]	Linear	[+]Details
Weighted independent dominating set [?]	Linear	[+]Details
Weighted independent set [?]	Linear	[+]Details
Weighted maximum cut [?]	NP-complete	[+]Details

But….``not so fast’’…linear in what? Also cost of constructing /reading graph?

Sometimes you don’t have the graph itself……..cost of deriving the graph….

Moreover, maybe linear **in #edges** which is typically **quadratic** in **#vertices**….unless sparse graph….

Yet even with those caveats, if you can show your data/problem instances are in the “special” class with “more nice properties” then maybe you can still devise more efficient algorithms than is currently known (because no-one else has realized these problem instances come from that special class).

Or maybe YOUR data/situation comes with even special “powers” in addition to the graph class/structure….

E.g., a cheap oracle for X (where a group of vertices is a clique or not…for example)

But….``not so fast’’…these are results for graphs – what about hypergraphs…

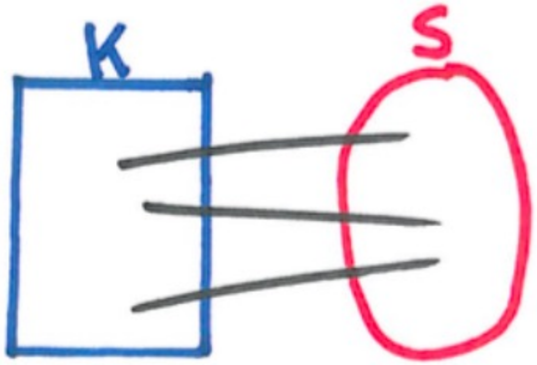
A k-uniform hypergraph is a graph when k=2.

For k>2, generally the situation is more complex and the “attractive properties” don’t always hold in the “hypergraph generalization**(s)**”

However the basic principle of course apply – a hierarchy of hypergraph classes with more favourable characteristics (but less data/instance coverage) as one moves down the hierarchy (to more specialized classes).

Definition: A *split graph* is a graph G whose vertex set can be partitioned as $V(G) = K \cup S$ where K is a clique and S is a stable set.

Such a partition is a KS-partition.



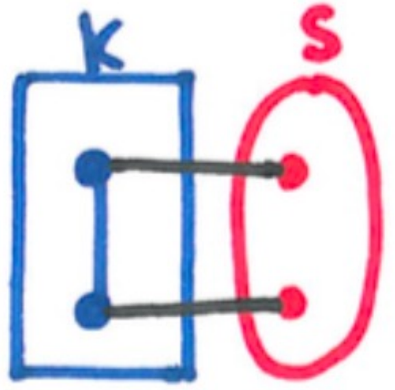
Definition: A KS-partition of a split graph G is *K-max* if $|K| = \omega(G)$ and *S-max* if $|S| = \alpha(G)$.

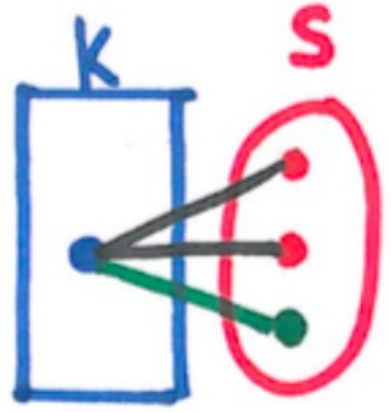
(From Trenk DIMACS talk)

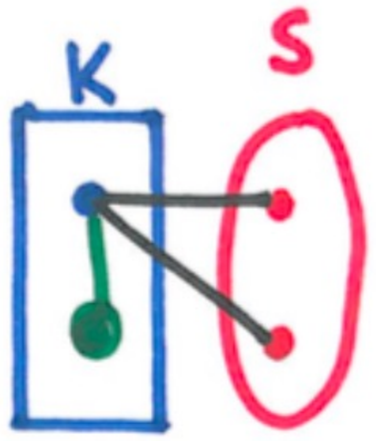
Why might split graphs be of interest?

- MANY reasons (discovered in the 1970’s when studying optimization and Linear programming in particular).
- Complex to explain some of these reasons…but look it up if interested!
- For my (our?) purposes – they describe a “perfect community”. The vertices (people) in the (maximal)clique are perfectly connected to each other. The vertices (people) not in the clique are not connected to each other (but may be connected to some of the people in the clique). More generally – perfect cluster…

P_4 $K_{1,3}$







P_4 has a unique KS-partition
It is both K-max and S-max.

$K_{1,3}$ has two KS-partitions
One is S-max, the other is K-max.

(From Trenk DIMACS talk)

Two kinds of split graphs

Theorem (Hammer, Simeone: 1977) For any KS-partition of a split graph G, exactly one of the following holds.

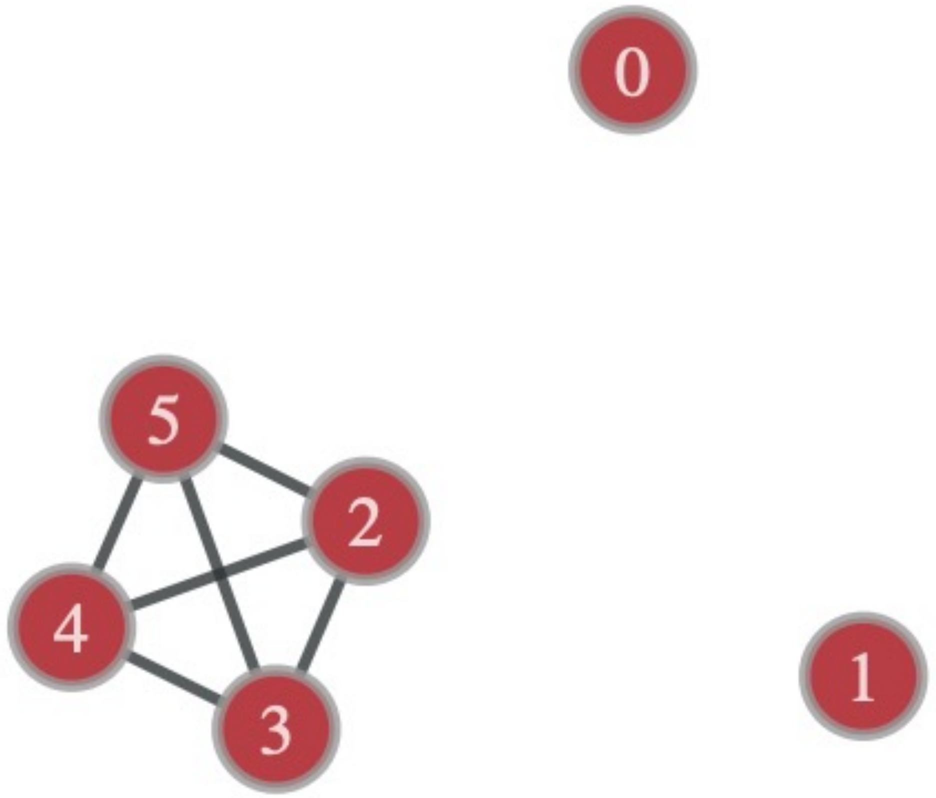
1. $|K| = \omega(G)$ and $|S| = \alpha(G)$. (K-max, S-max)
2. $|K| = \omega(G) - 1$ and $|S| = \alpha(G)$. (S-max)
3. $|K| = \omega(G)$ and $|S| = \alpha(G) - 1$. (K-max)

Moreover, in
(1.) the partition is unique, in
(2.) there exists $s \in S$ so that $K \cup \{s\}$ is complete, and in
(3.) there exists $k \in K$ so that $S \cup \{k\}$ is a stable set.

Theorem (Cheng, Collins, Trenk: 2016) Let G be a split graph with degree sequence $d_1 \geq d_2 \geq \cdots \geq d_n$ and let $m = \max\{i : d_i \geq i - 1\}$. Then G is unbalanced if $d_m = m - 1$ and balanced if $d_m > m - 1$.

(From Trenk DIMACS talk)

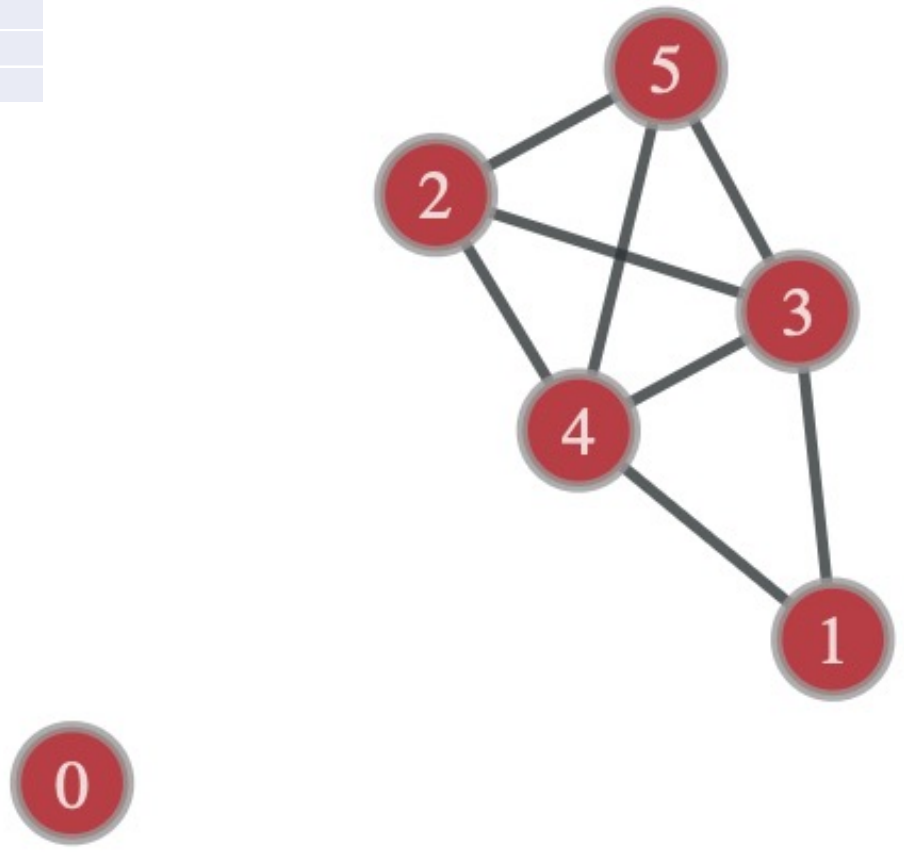
Degree Sequence	3	3	3	3	0	0
i-1	0	1	2	3	4	5
m	3					
m(m-1)	6					
sum_di_to_m	9					
sum_di_m+1_to_n	3					
splittance	0					



"UnBalanced" – all members of clique are swing vertices

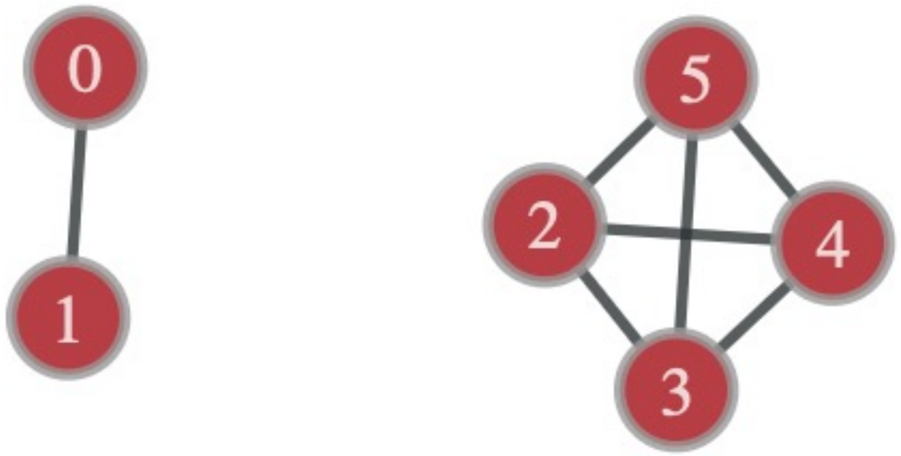
Degree Sequence	4	4	3	3	2	0
i-1	0	1	2	3	4	5
m	3					
m(m-1)	6					
sum_di_to_m	11					
sum_di_m+1_to_n	5					
splittance	0					

Input

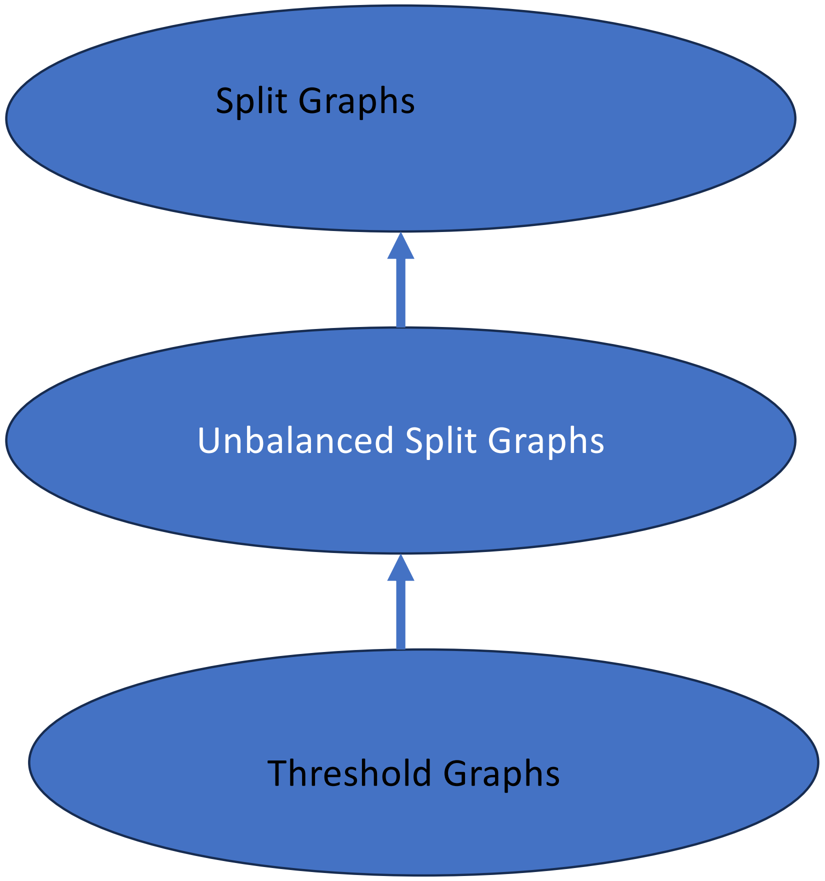


Unbalanced – members 2 and 5 of the clique are swing vertices

Degree Sequence	3	3	3	3	1	1	
i-1	0	1	2	3	4	5	
m	3						
m(m-1)	6						
sum_di_to_m	9						
sum_di_m+1_to_n	5						
splittance	1						



All threshold graphs are unbalanced split graphs (but not visa versa)



{0,1}* constructible graphs

- Threshold graphs have a fun/simple description – one that says how they can be constructed.
- Order your vertices – then, one by one, in order, add these vertices choosing either:
  - Add an isolated vertex (totally unconnected to any previous vertex) – if 0 in construction string
  - Add a dominating vertex (totally connected to every previous vertex) – if 1 in construction string

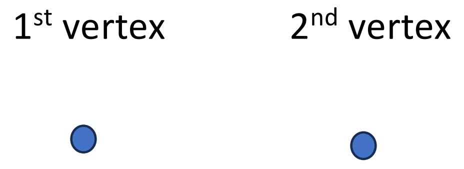
Draw these graphs given the “construction strings”

- 00111
- 11100
- 11010

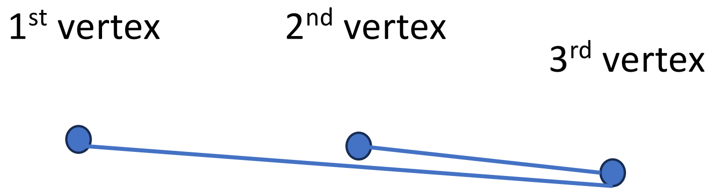
00111

1st vertex

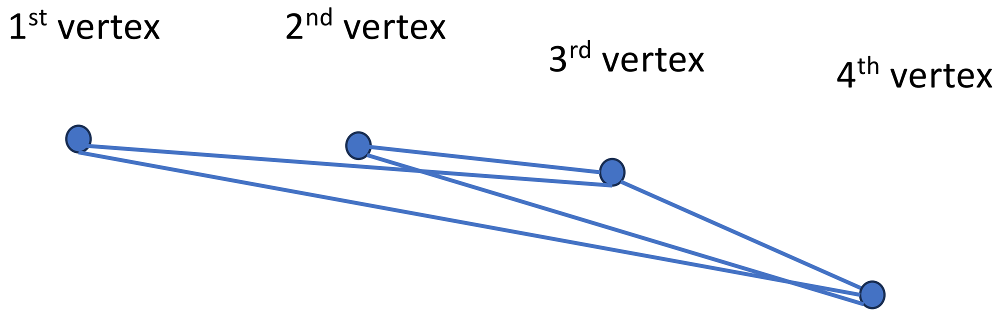
00111



00111

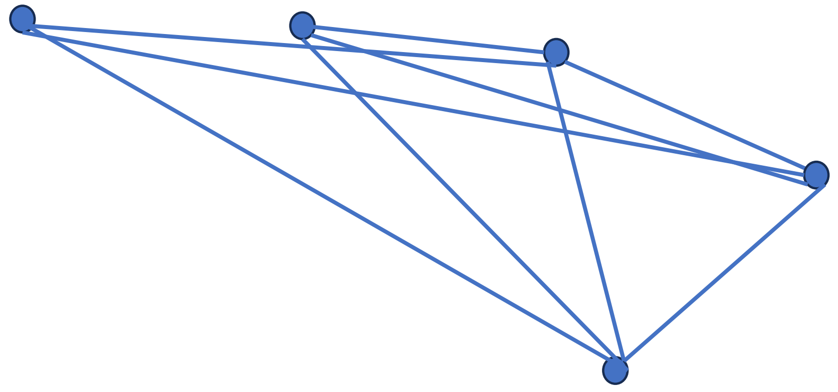


00111

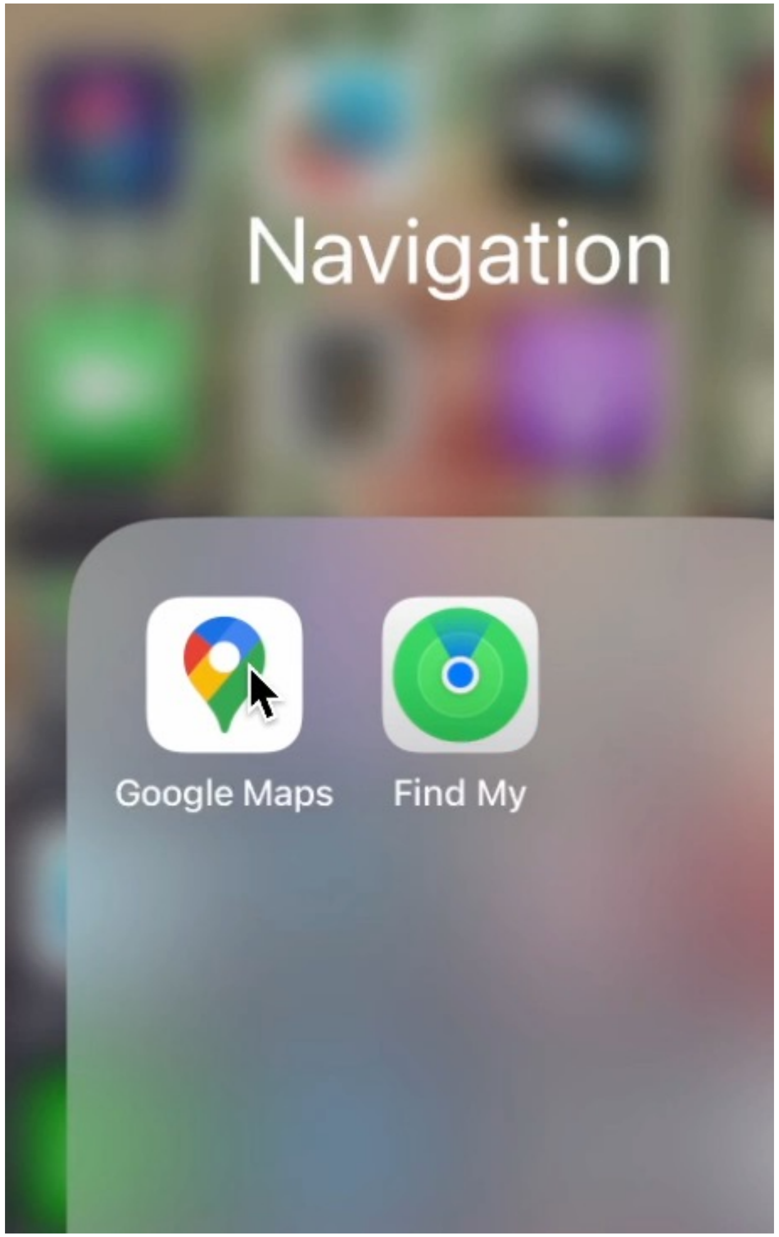


00111

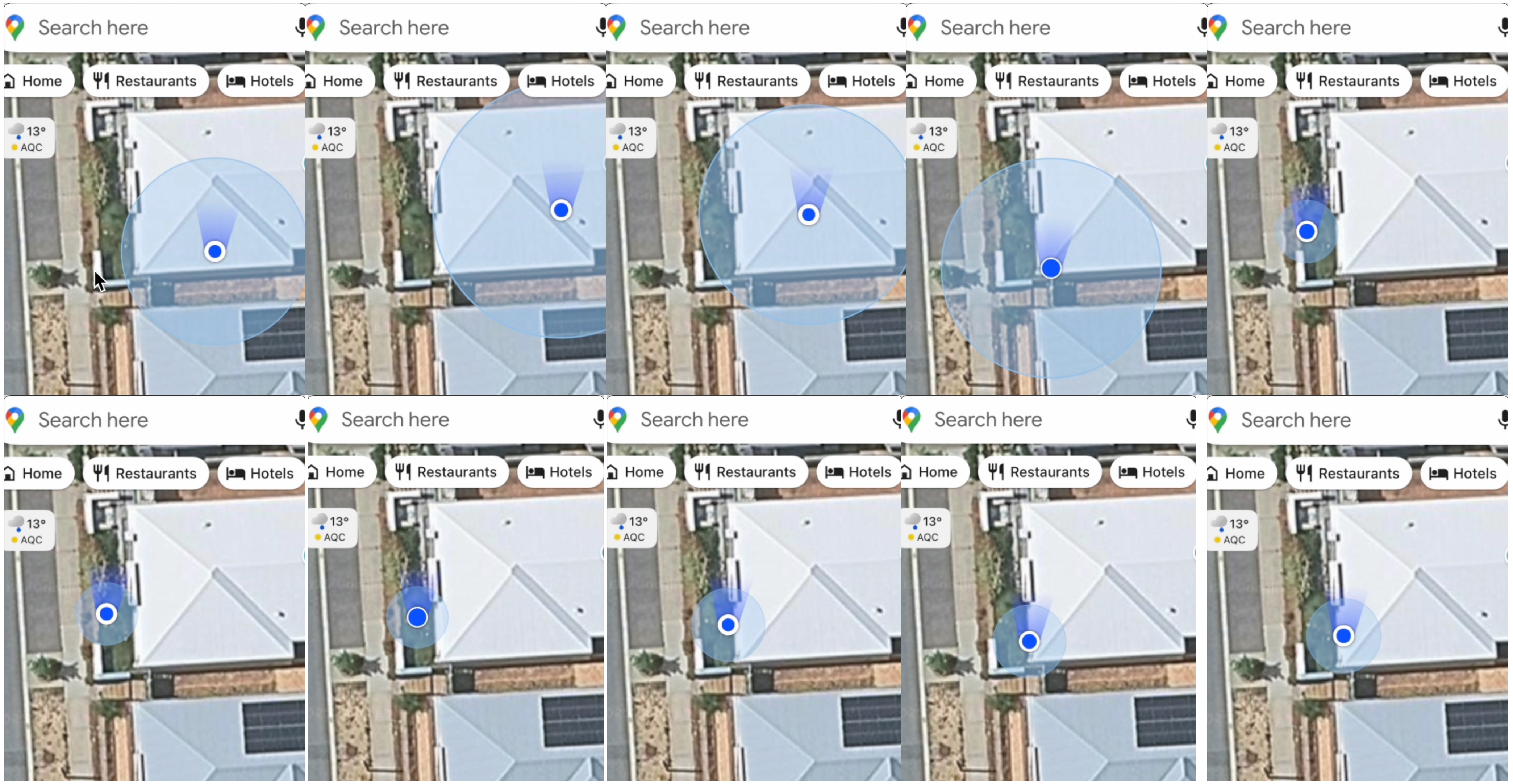
1st vertex 2nd vertex 3rd vertex 4th vertex 5th vertex



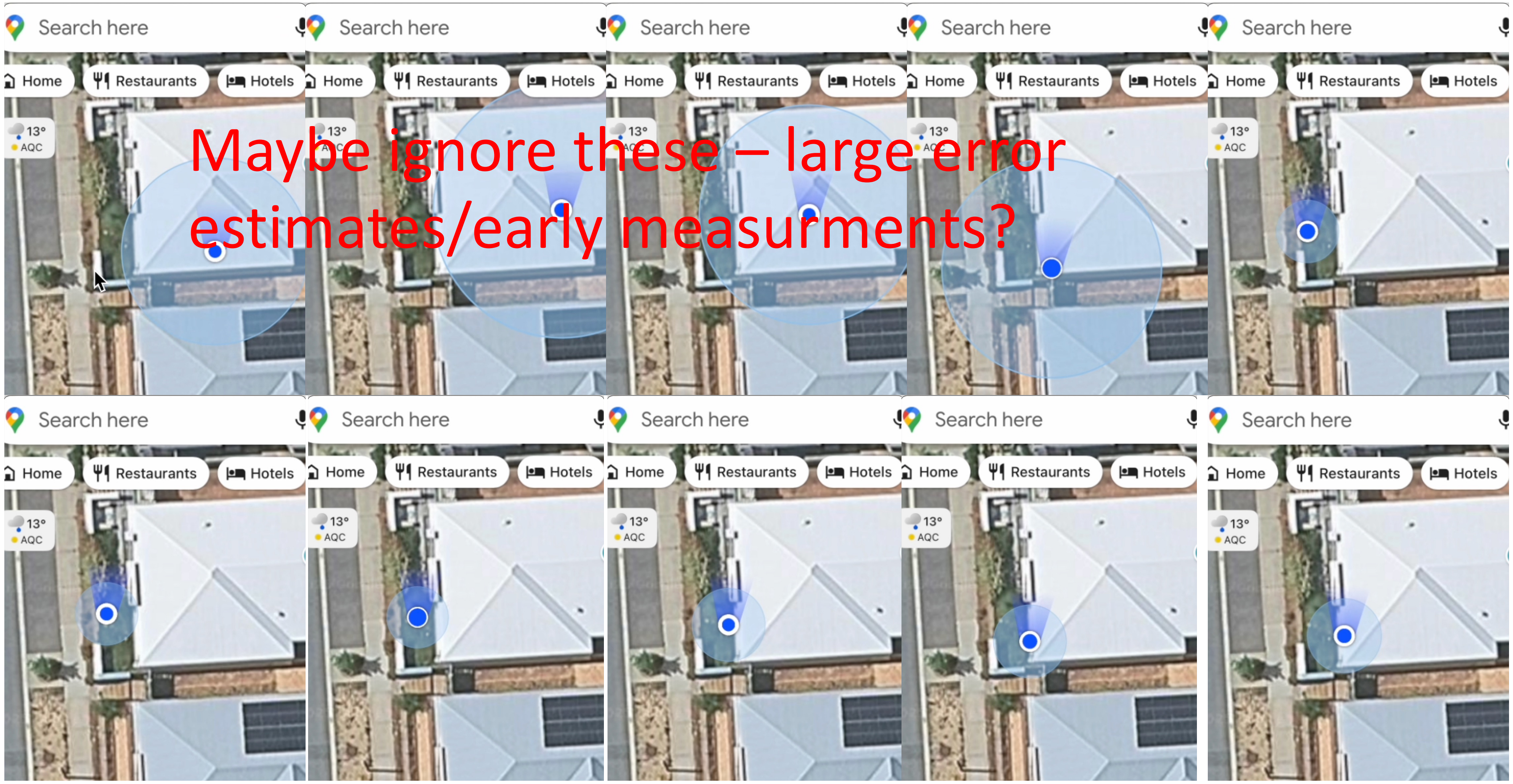
Draw it better! It's an independent set of size 1, a clique of size 3 and every member of the clique is joined to the two independent vertices. A split graph with two swing vertices….an unbalanced split graph – as all threshold graphs are!


Navigation
Google Maps
Find My

Where am I? 2D location estimation

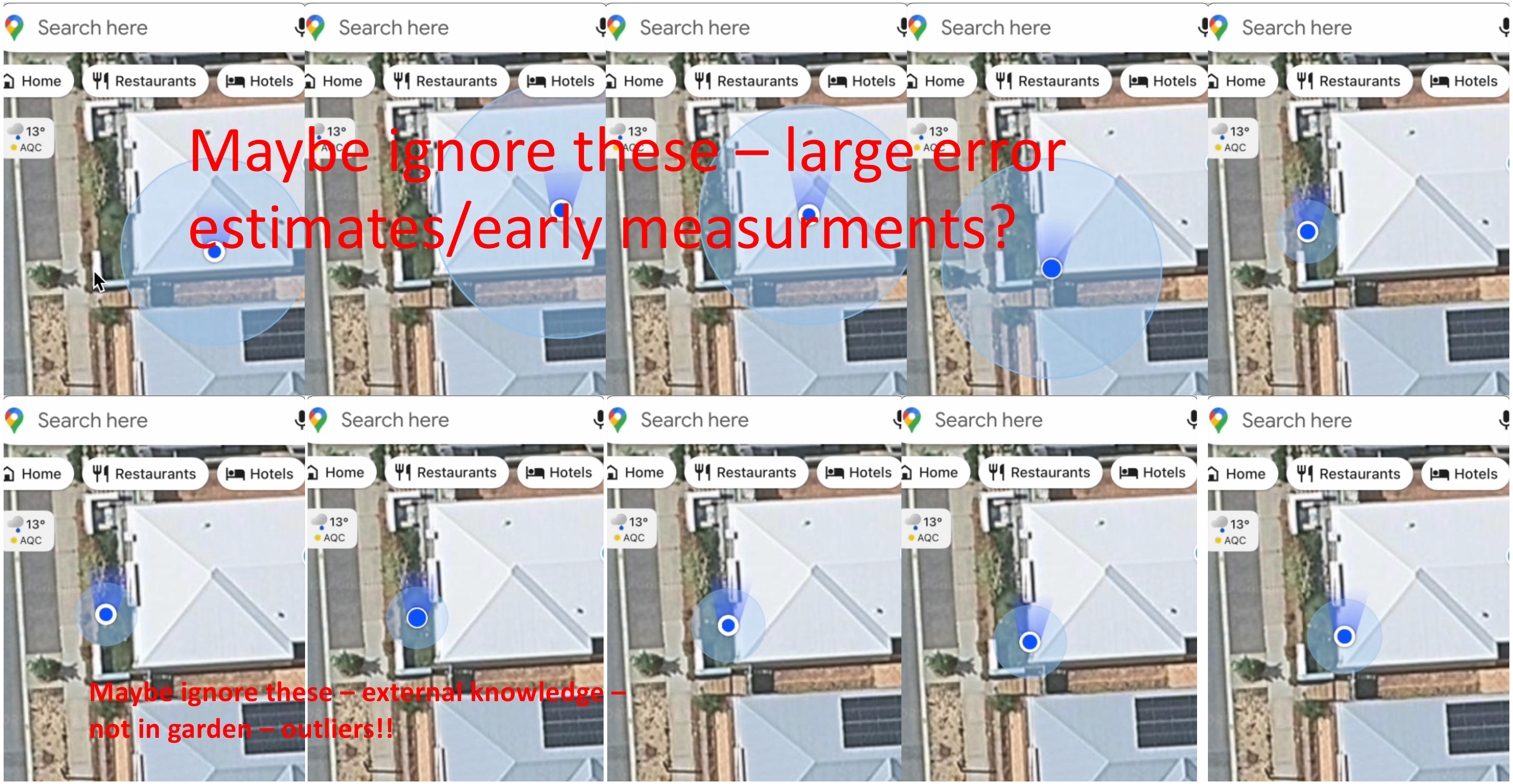


Where am I?   2D location estimation



Maybe ignore these – large error estimates/early measurments?

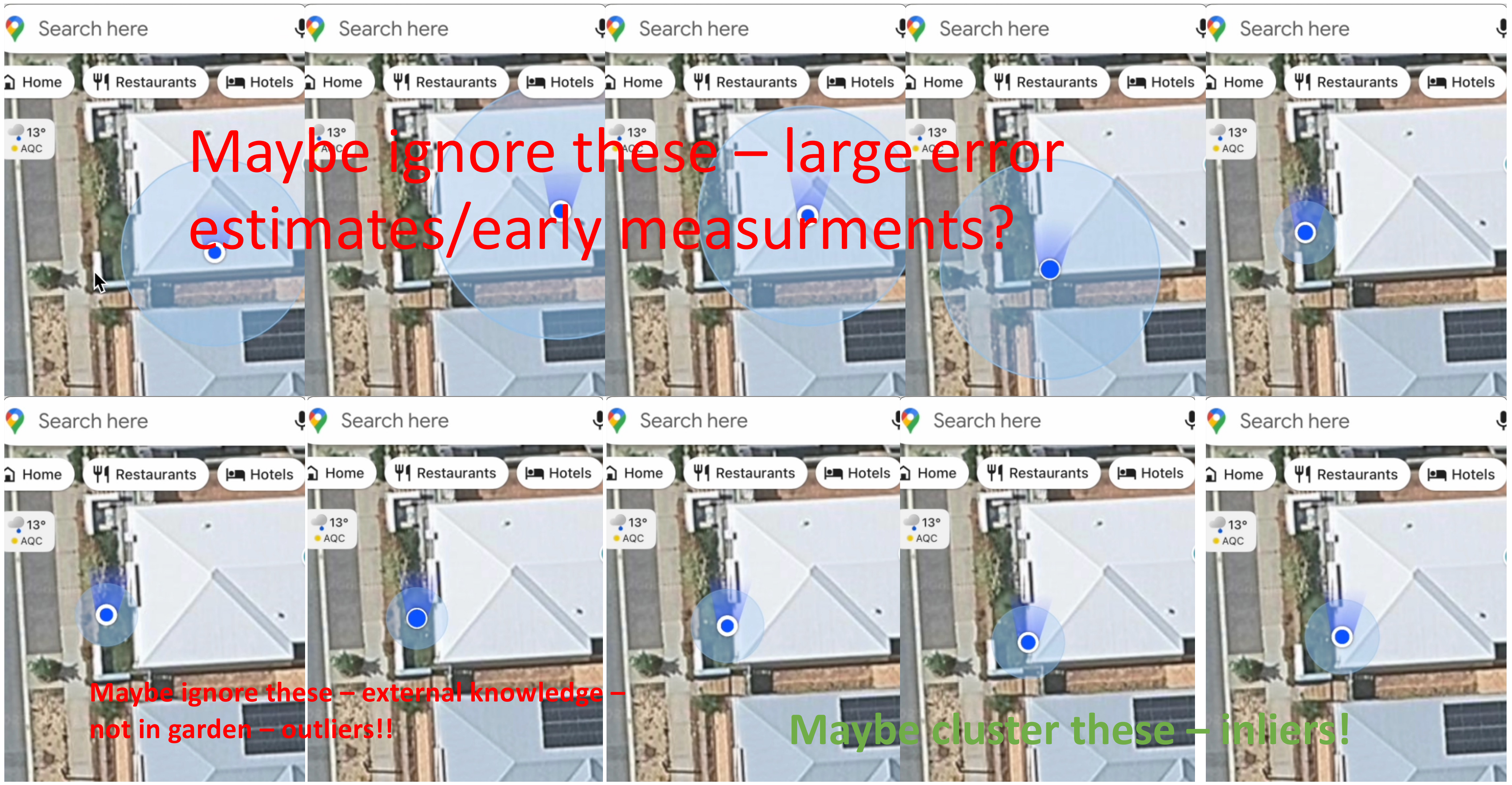
Where am I?   2D location estimation



Maybe ignore these – large error estimates/early measurments?

Maybe ignore these – external knowledge – not in garden – outliers!!

Where am I?   2D location estimation

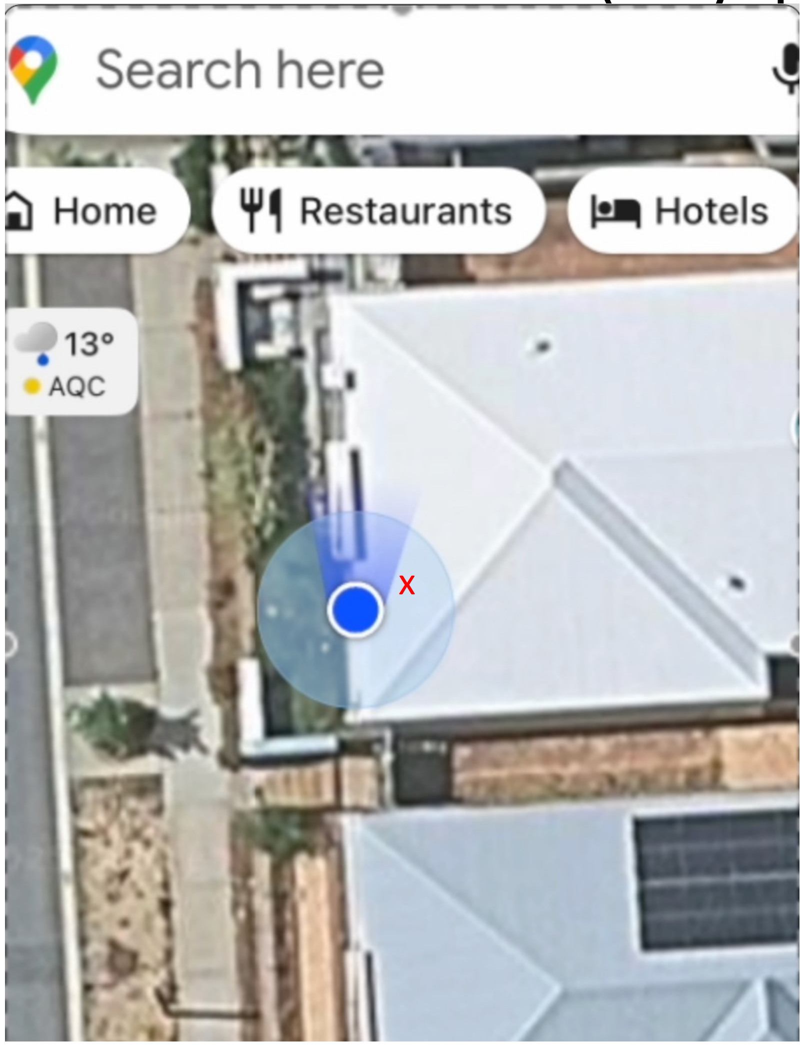


Maybe ignore these – large error estimates/early measurments?

Maybe ignore these – external knowledge – not in garden – outliers!!

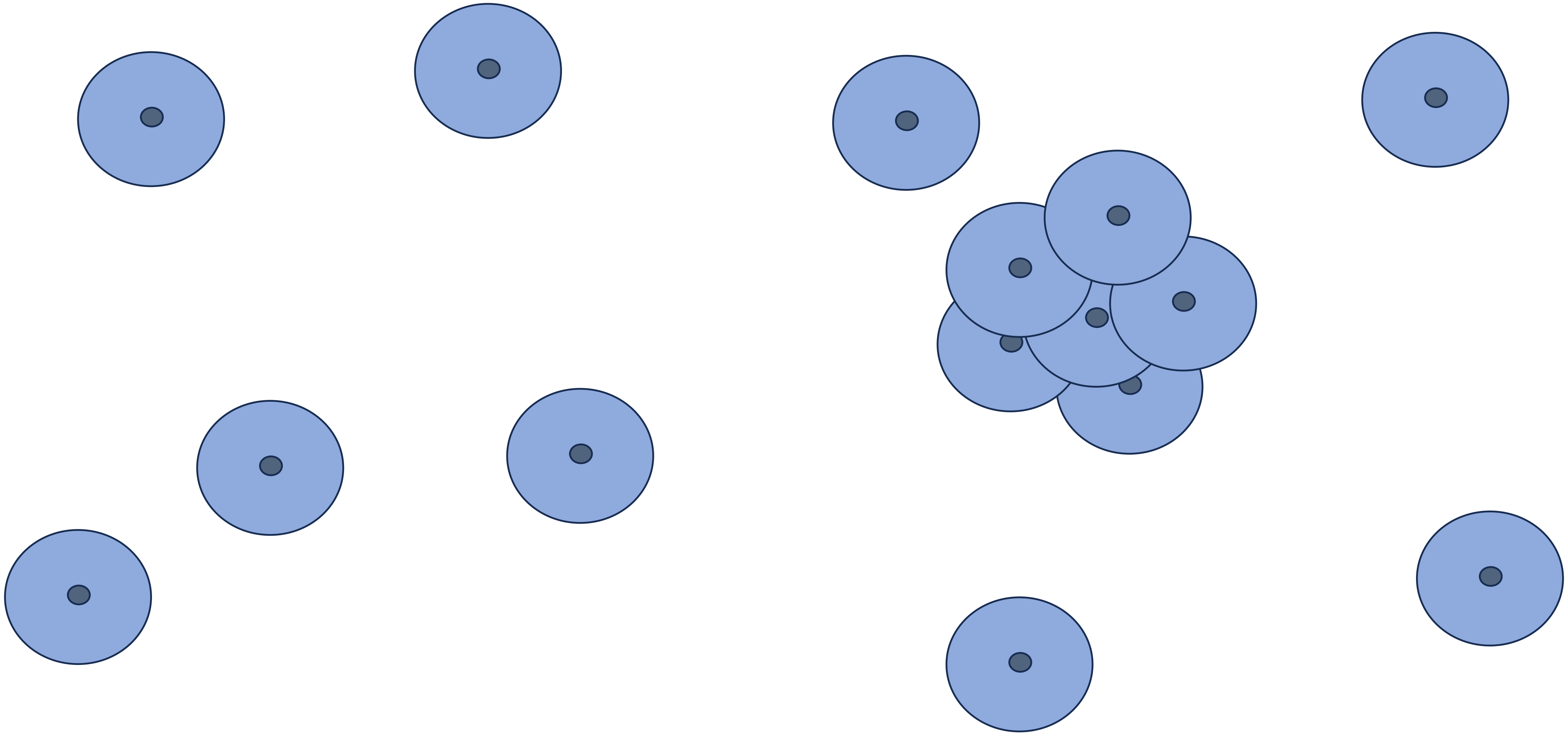
Maybe cluster these – inliers!

Where was I? (My phone)

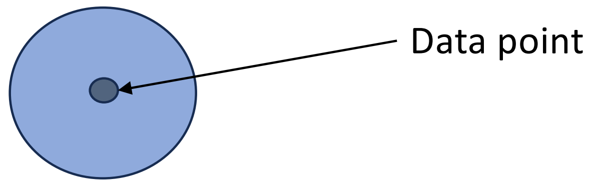


(my best guess from knowledge of where desk is and interior of house)

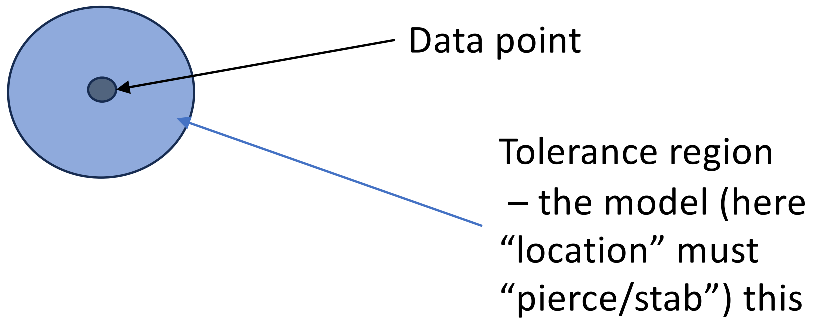
2D location estimation – model problem



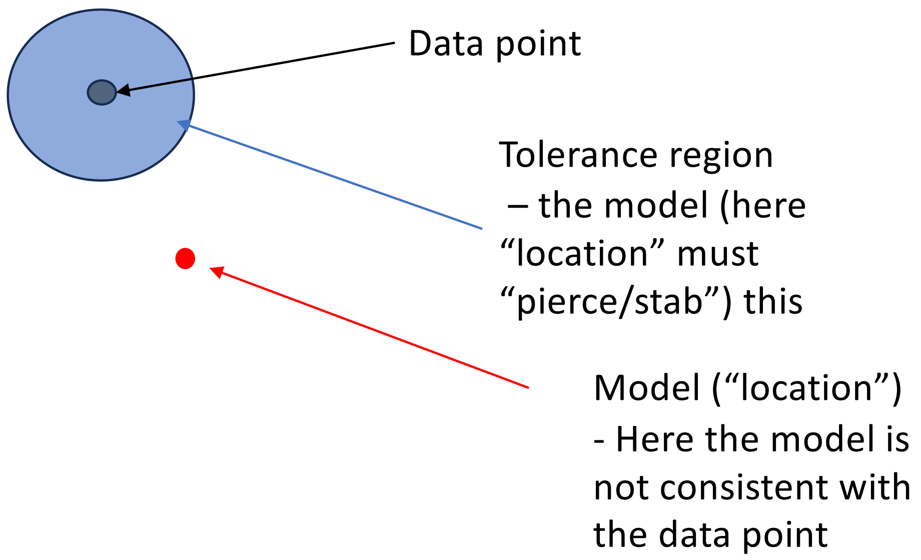
Models/hypotheses, data, and metrics (tolerances)



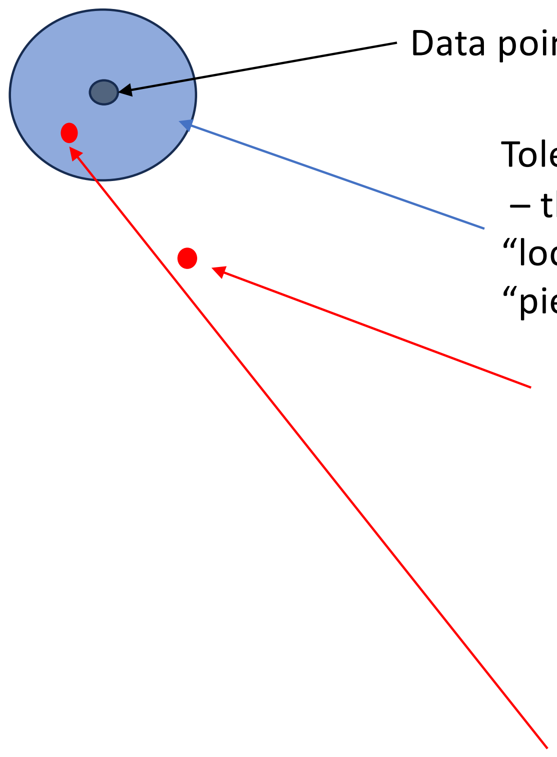
Models/hypotheses, data, and metrics (tolerances)



Models/hypotheses, data, and metrics (tolerances)


Data point
Tolerance region – the model (here “location” must “pierce/stab”) this
Model (“location”) - Here the model is not consistent with the data point

Models/hypotheses, data, and metrics (tolerances)



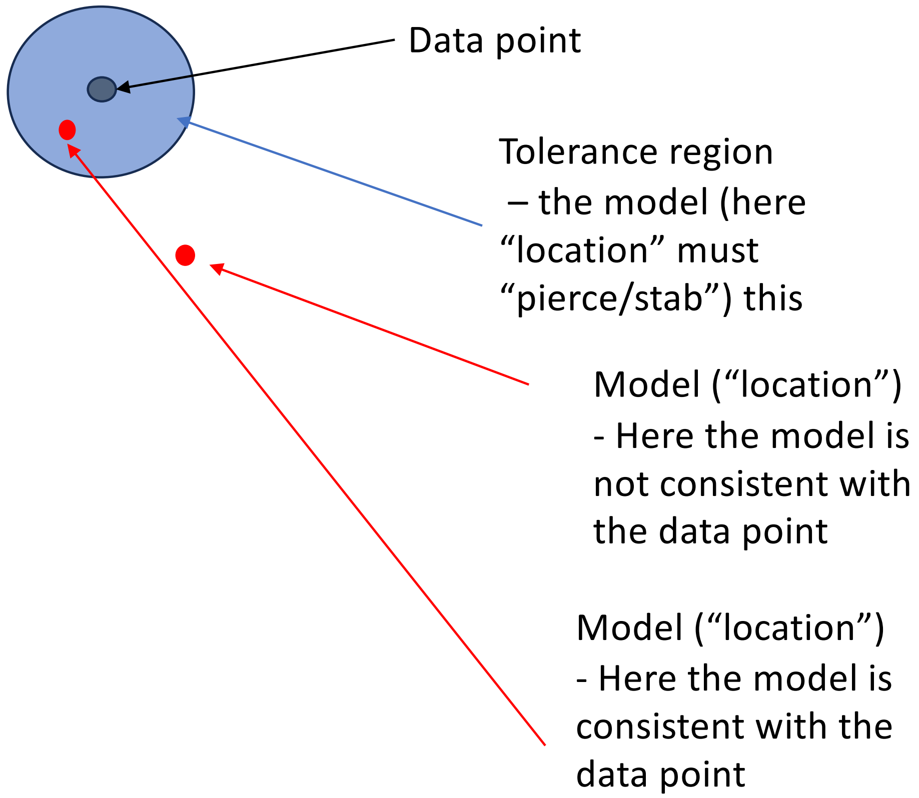
Data point

Tolerance region
– the model (here "location" must "pierce/stab") this

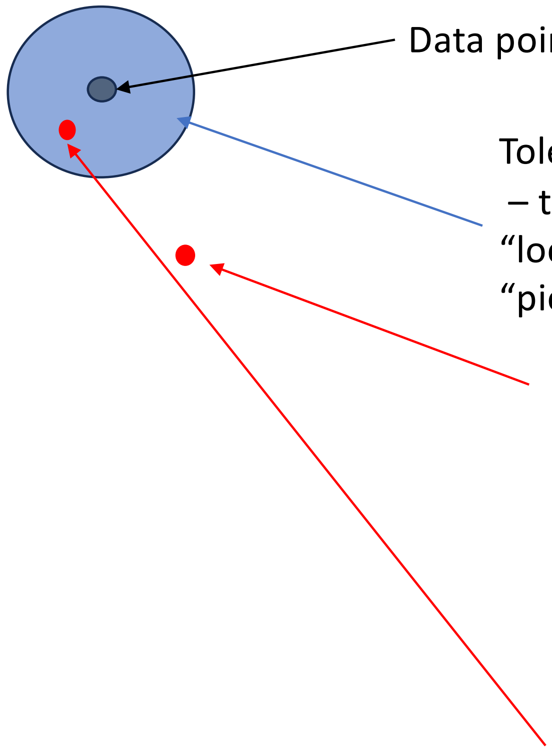
Model ("location")
- Here the model is not consistent with the data point

Model ("location")
- Here the model is consistent with the data point

Models/hypotheses, data, and metrics (tolerances)


Data point
Tolerance region – the model (here "location" must "pierce/stab") this
Model ("location") - Here the model is not consistent with the data point
Model ("location") - Here the model is consistent with the data point

Models/hypotheses, data, and metrics (tolerances)



Data point

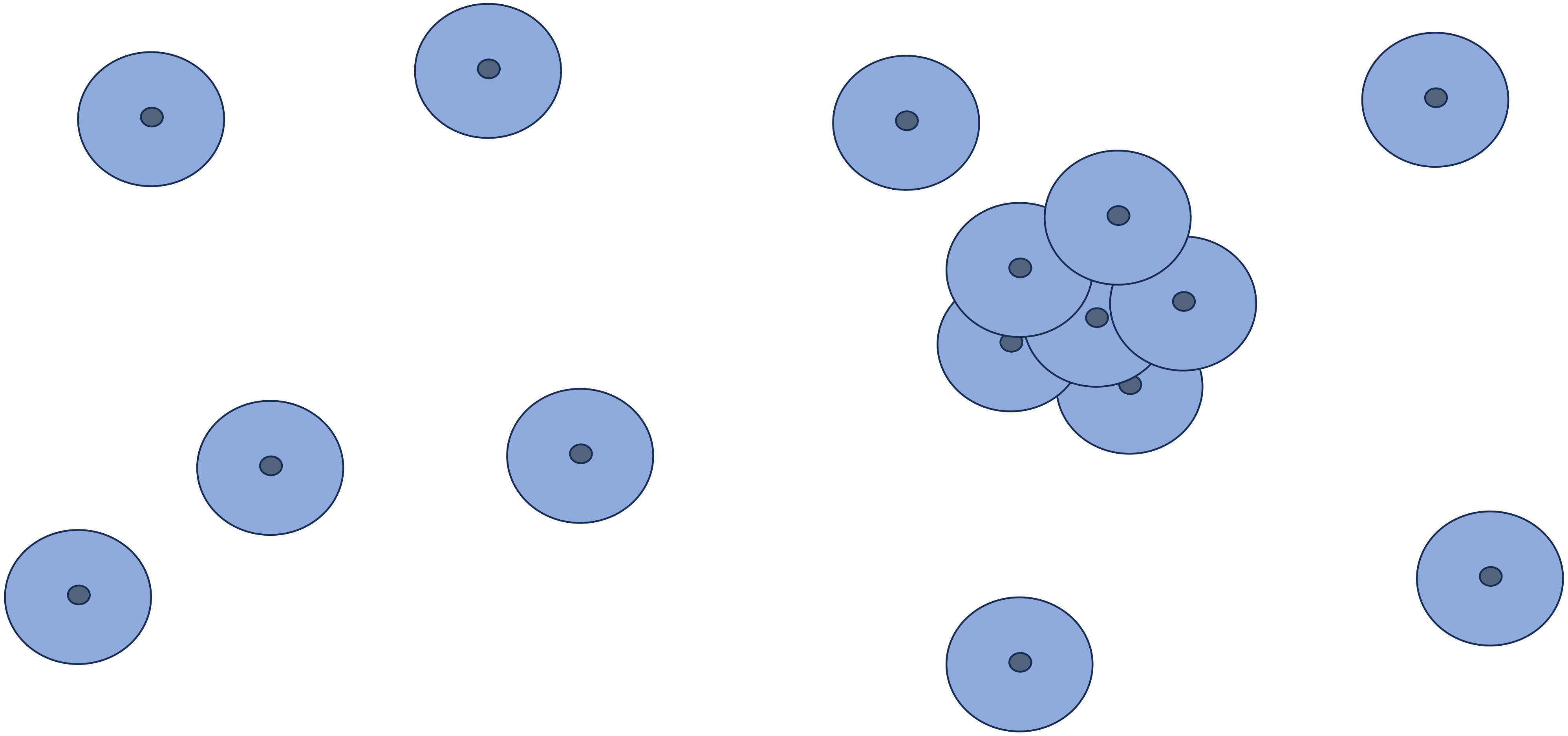
Tolerance region
– the model (here “location” must “pierce/stab”) this

Model (“location”)
- Here the model is not consistent with the data point

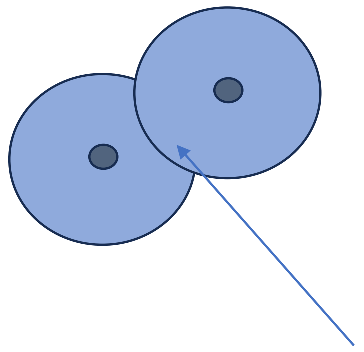
Model (“location”)
- Here the model is consistent with the data point

Tolerance region (around a data point) is the set of all models consistent with the data point

2D location estimation – model problem



2D location estimation – model problem



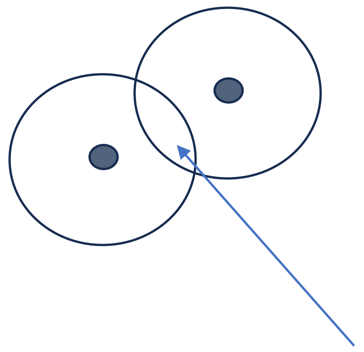
Common models – models they are with are in the intersection

2D location estimation – model problem

So…we can take every pair of data points and decide if they “agree” or not.
This would form a graph. Each vertex in the graph is a data point. There is an edge between each two “if they agree” and no edge otherwise.

So models “supported by the data” are associated with many “data points that agree” – clusters/cliques in graphs.

MaxCon (maximum concensus) is essentially clique finding.



Common models – models they are with are in the intersection

Maximum Clique in Unit Disk graphs

- One has to distinguish two situations:
  - One has the data – then one can use the “data space” geometry/information
  - One has only the graph (it is usually quite hard to then derive a data configuration that would generate that graph – so one has to use ”pure” graph algorithms
- For most of the problems/scenarios I’m interested in – one has the data. Moreover, for hypergraphs (later) especially, one doesn’t even want to construct the (hyper)graph (though going from data to (hyper)graph is usually much easier than the other way around!)

MaxClique on Unit Disk graphs given the data

- A fairly recent survey of state of the art algorithms can be found in https://arxiv.org/pdf/2506.21926 (June 27 2025).
  - (1990) $O(N^{4.5})$ algorithm
  - (1991) $O(N^{3.5} \log N)$ algorithm
  - (2007) $O(N^{3} \log N)$ algorithm
  - (2023) $O(N^{2.5} \log N)$ algorithm
  - (2025) $O(N^{7/3} \log N)$ algorithm
- What if the best one can ever do? That’s yet to be determined!

What if you are NOT given the data, and what about if the graph is NOT a unit disk graph (you want to find max clique in a given graph)..

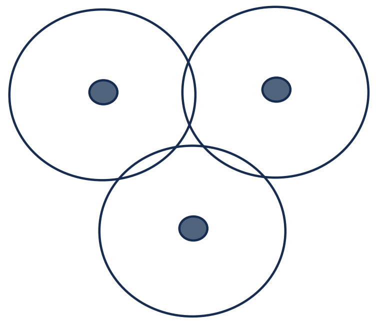
- The naive algorithm is to test every subset of vertices to see if is a clique and to return the largest such. This is a terrible algorithm for “large” n – as the number of subsets is 2^N (so the runtime of the naïve algorithm is roughly $p(N)2^N$ for some (uninteresting for very large N) polynomial p(N).
- It has been known since the 1970’s (e.g., 1976 Stanford technical report of Tarjan) that one can *in principle* shave the exponent (that technical report shaves to N/3) – but as that report admits, this only “buys” a factor of 3 in the size of the graphs one can handle and at the expense of a complicated algorithm – it’s more a theoretical than a practical algorithm.

What if you are NOT given the data, and what about if the graph is NOT a unit disk graph (you want to find max clique in a given graph)..

- As we saw, we can generally do much better if we know the class of graphs (unit disk graph) AND have the data that gave rise to that graph – somewhere between O(N^2) and O(N) in that case.
- But how much of the gain comes from the restricted graph class and how much comes from having access to the data that generated the graph?
- This set of notes is around those sort of questions…

MaxClique on Unit Disk graphs given the data

Note: there are subtley different versions of what one might mean by MaxClique – for Unit Disk tolerances (and more generally for our “zoo” of problems….). It has to do with what is known as the Helly number/Helly’s theorem.



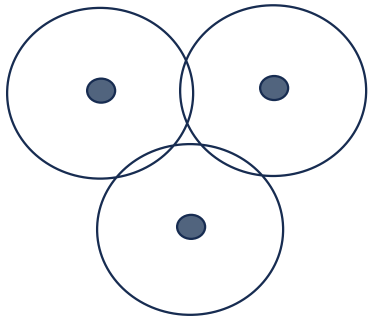
Pairwise agreement amongst all pairs of 3 data points.

But is there (at least one) MODEL they all agree on??

MaxClique on Unit Disk graphs given the data

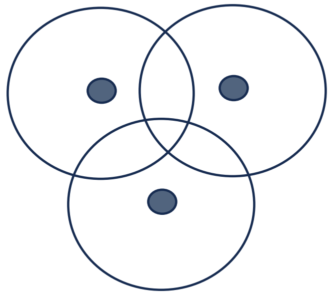
Three (or more) data points that only *pairwise* agree may not agree at each triple (or more) on a *common* model.

Helly: if all (d+1)-sized subsets of a set of convex bodies in R^d intersect, then they ALL intersect (at a common point – or set of common points). Here d=2 so (2+1)-set intersections are what you need.



Pairwise agreement amongst all pairs of 3 data points.

But is there (at least one) MODEL they all agree on??



3-wise agreement

Helly's theorem states that you need to enforce 3-wise agreements to ensure that ALL data in some subset "agree" – stronger notion of "agree". Leads to **hyper**graph versions of unit disk "agreement" in R^2

Unit Disk Graphs

Theorem 4.1 *For every integer $k \geq 1$, complement($K_2 + C_{2k+1}$) is a minimal non-UDG.*
Theorem 4.3 *For every integer $k \geq 4$, complement(C_{2k}) is a minimal non-UDG*
Theorem 4.4 *For every integer $k \geq 4$, C^*_{2k} is a minimal non-UDG. (treat as bipartite graph and complement each induced subgraph (empty!) on the two vertex sets.*
Also the following are minimal forbidden induced subgraphs: complements of $S_{3,3,3}$, F_1, F_2, F_3; the graphs F_1^*, F_2^*, F_3^*; and those over the page

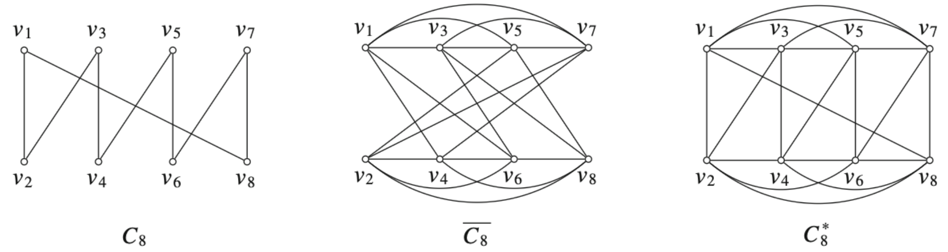


Fig. 6 Graphs C_8, $\overline{C_8}$, and C_8^*

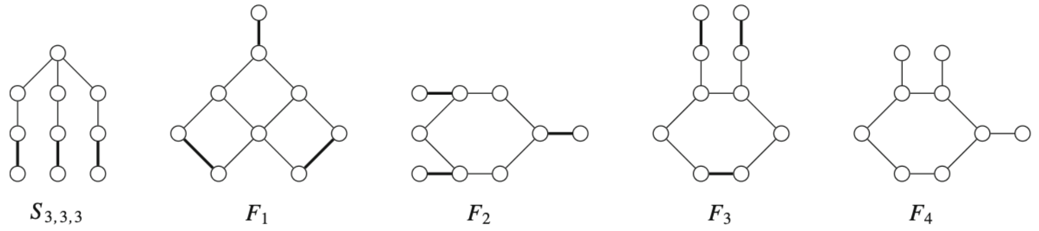


Fig. 7 Bipartite graphs $S_{3,3,3}$, F_1, F_2 and F_3 contain an edge-asteroid triple. Graph F_4 is the bipartite complementation of F_1

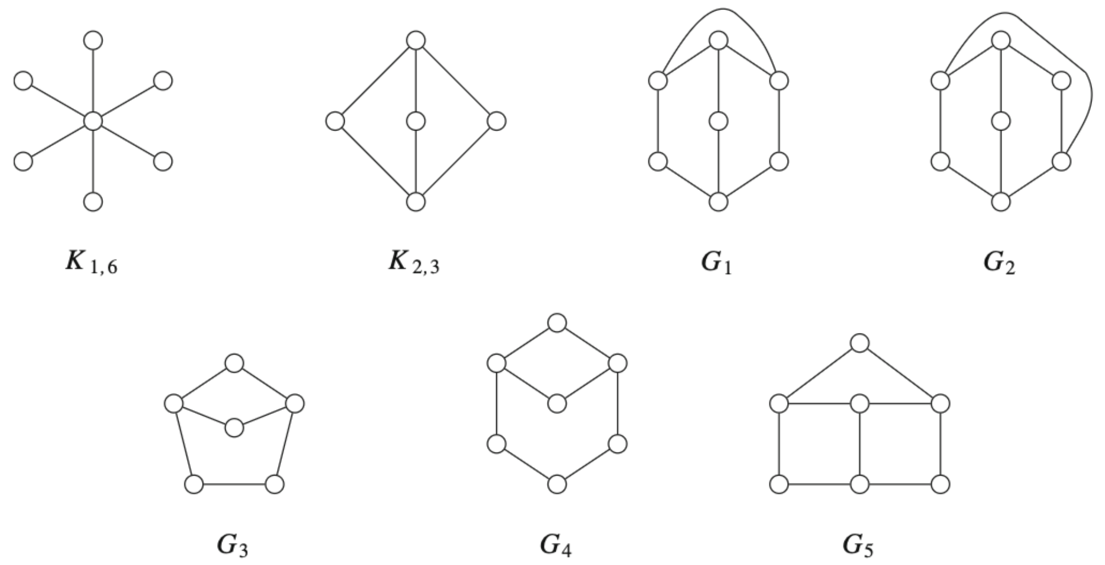
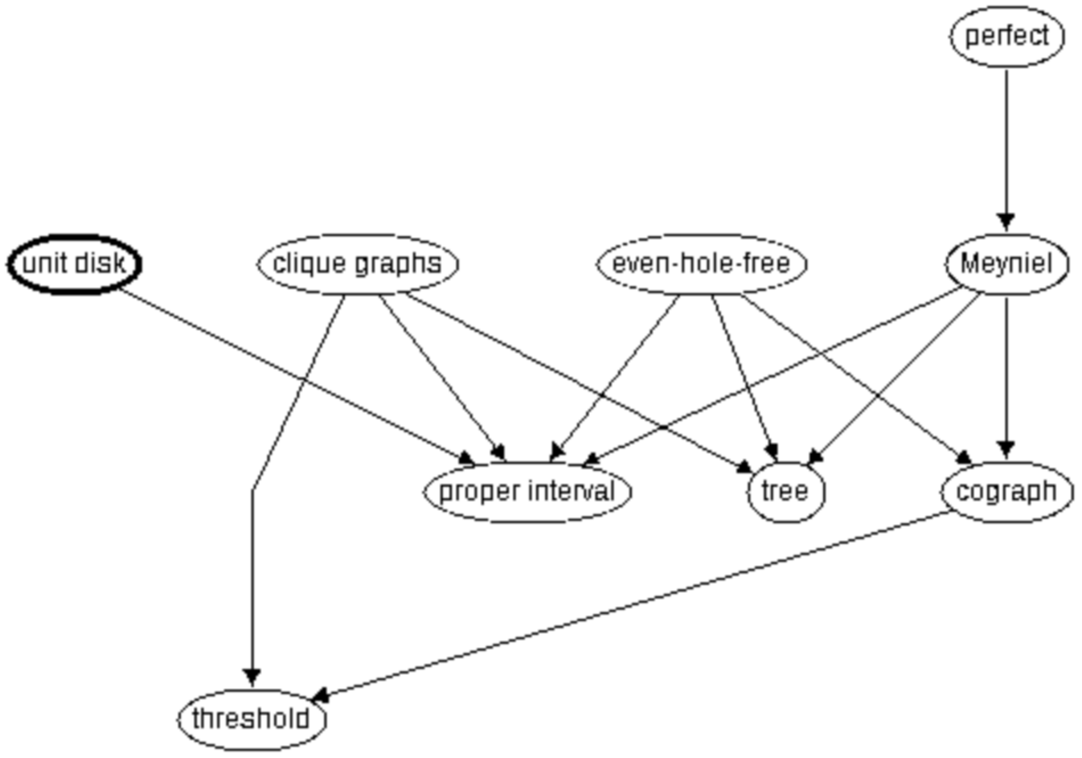
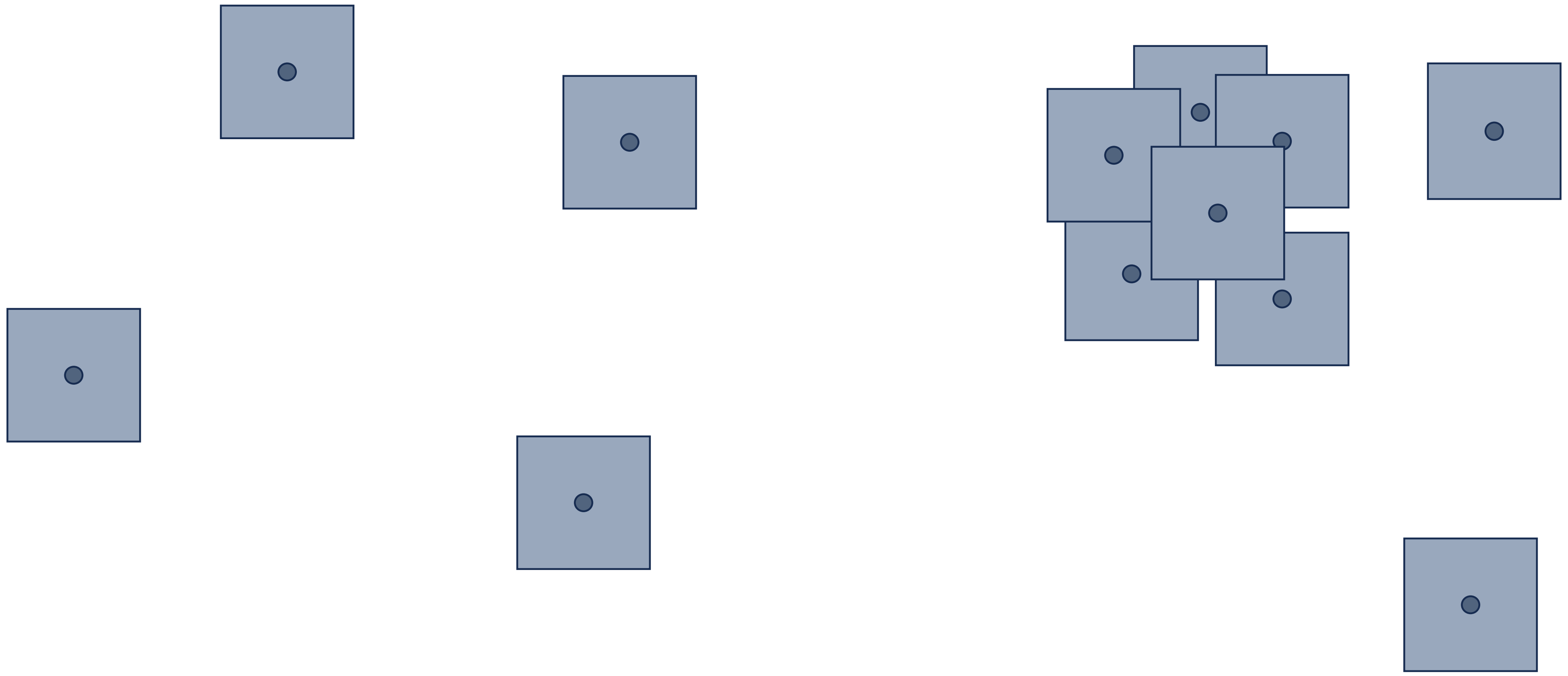


Fig. 1 Known minimal non-unit disk graphs

Map: [-]Hide details

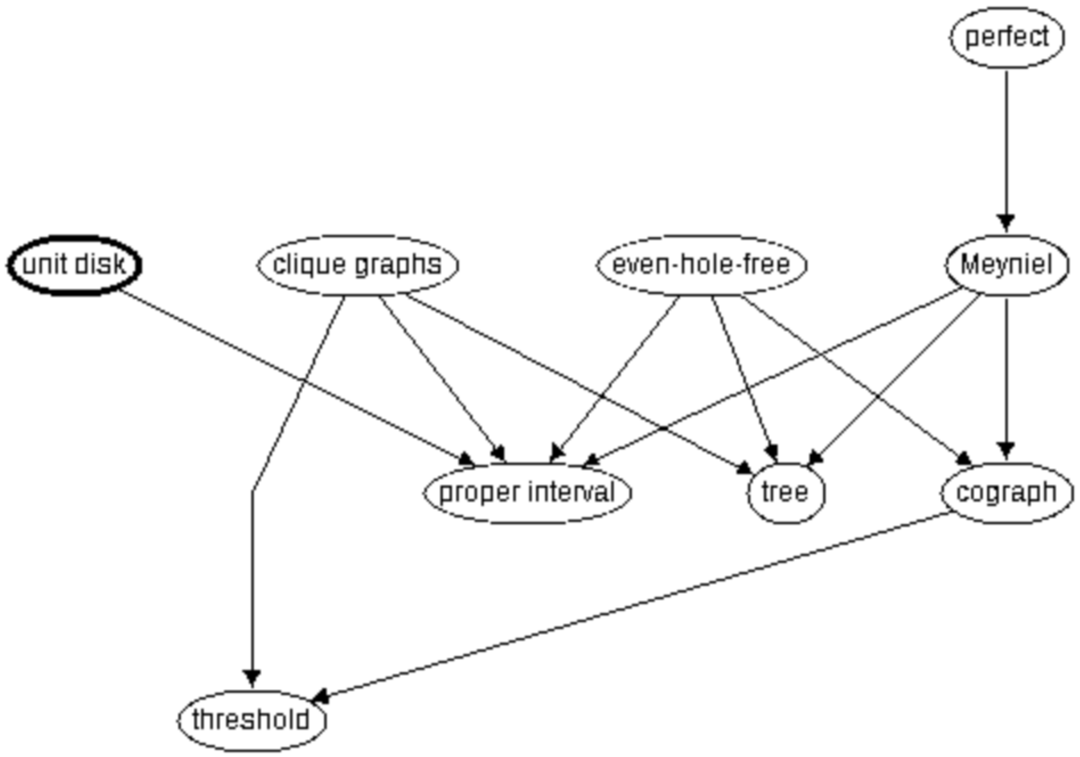


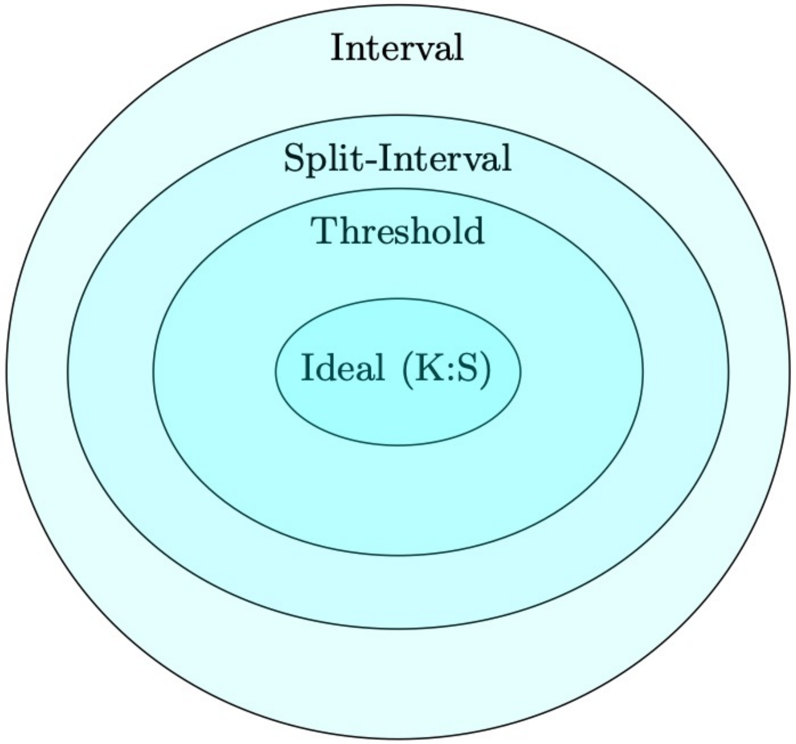
2D location estimation – Manhattan distance



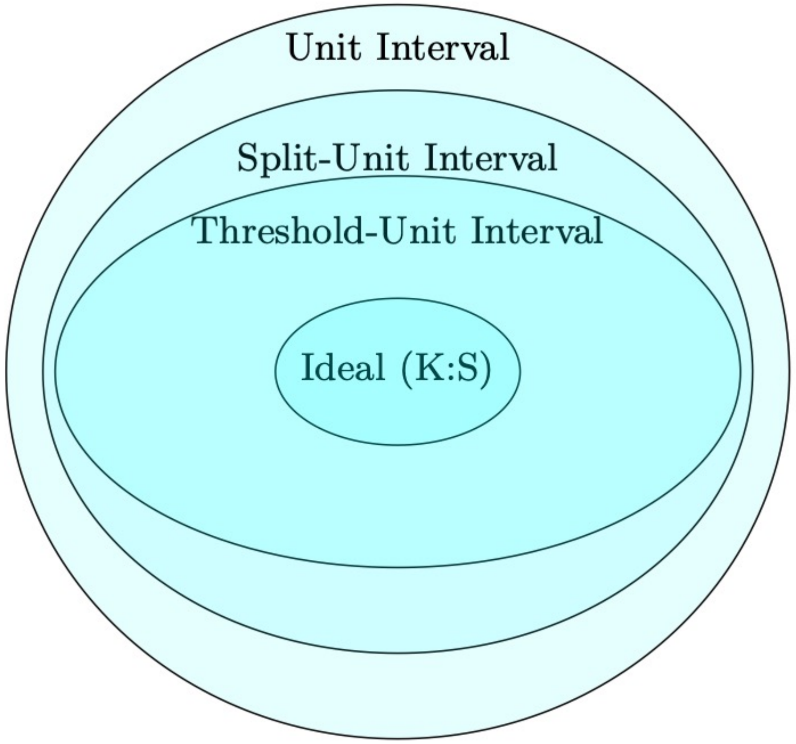
(Maybe more appropriate for images/rectangular grid data..”to the nearest pixel”)

Map: [-]Hide details





Graph Classes - 1D subspace regression



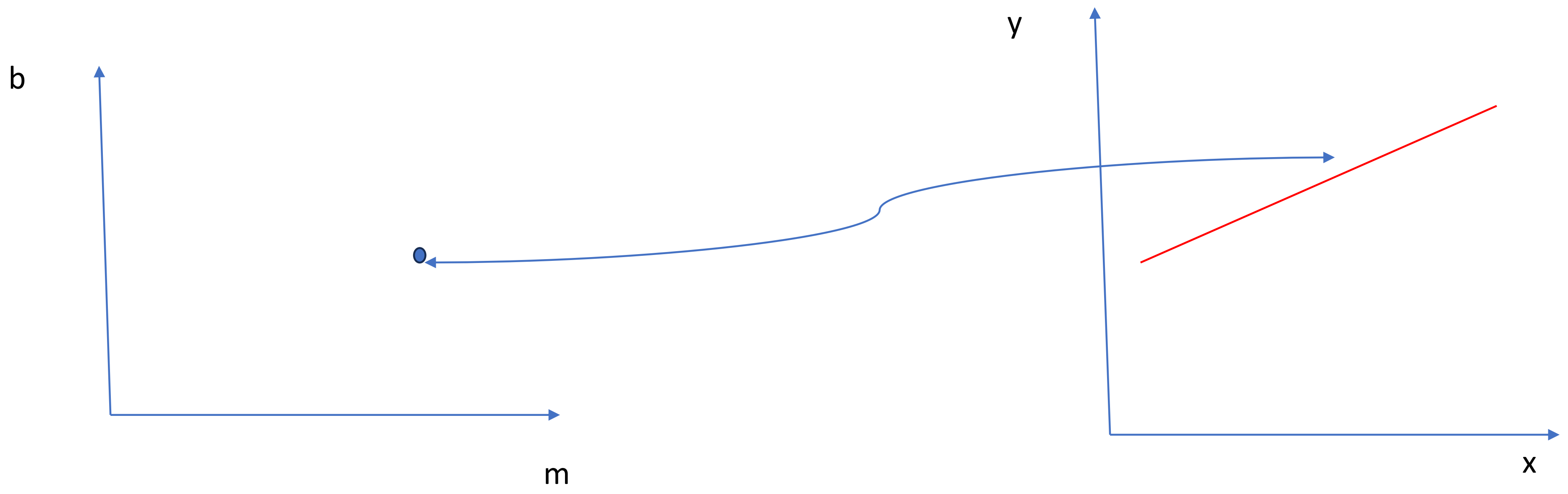
Graph Classes - 1D location

Much "nicer" data…caveat is that tolerance is uniform….otherwise becomes similar to 1D subspace regression

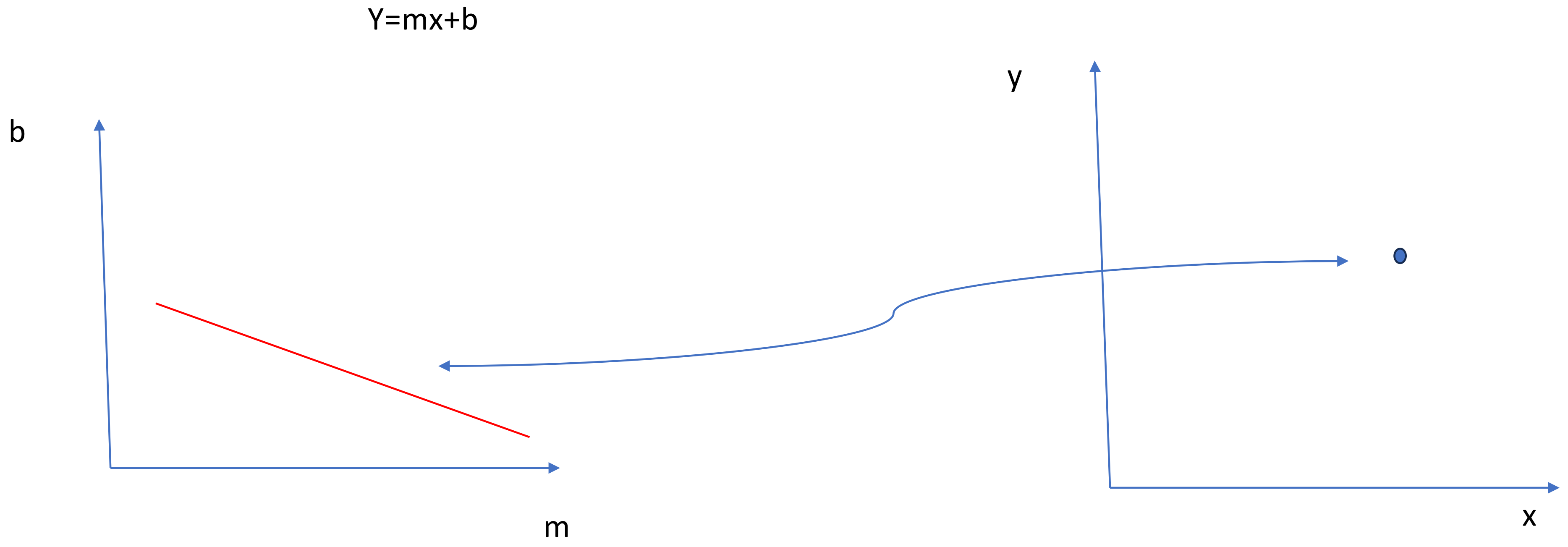
3-Uniform Hypergraphs for line fitting

"Model Space" – Point-line duality

Y=mx+b

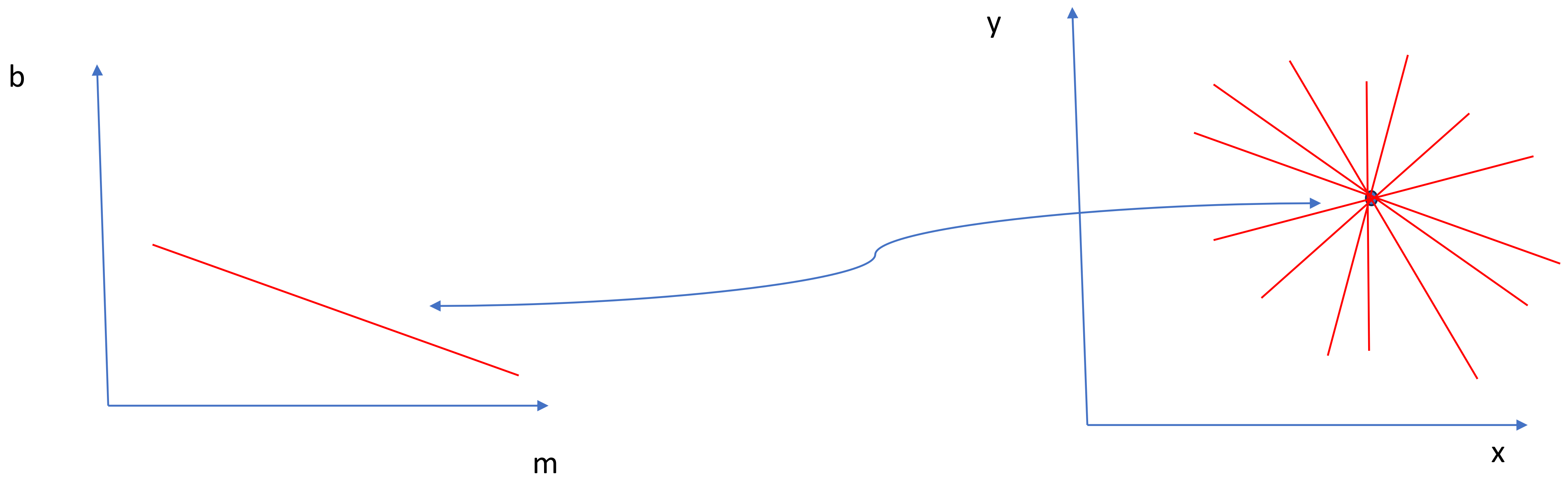


“Model Space” – Point-line duality



Actually a “pencil of lines”

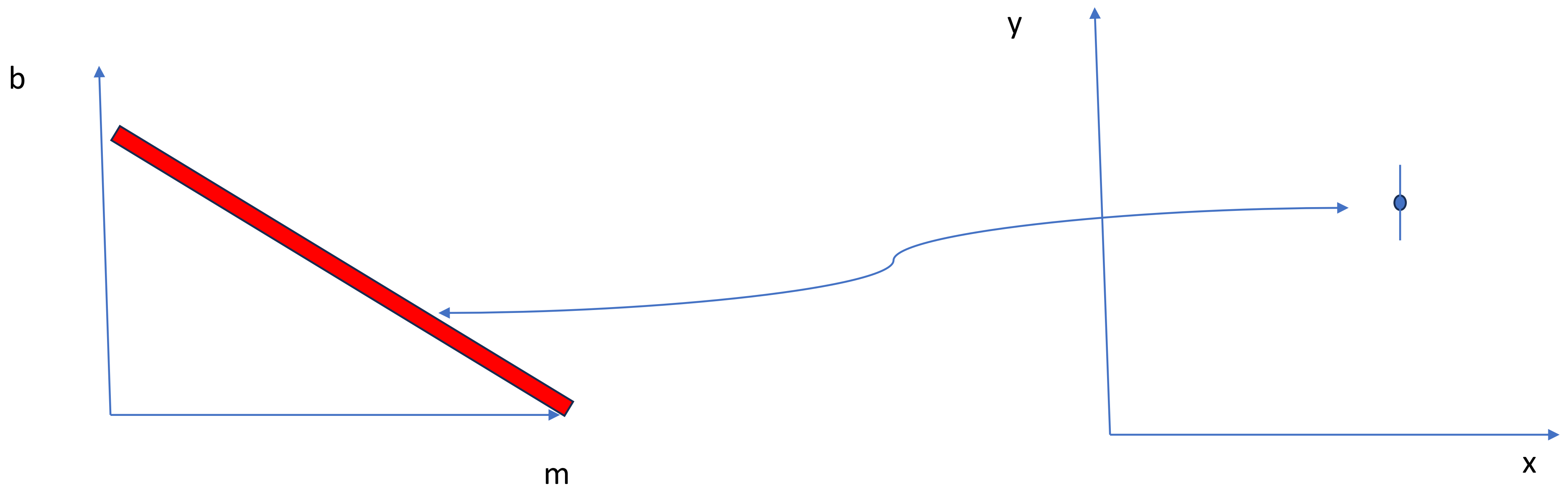
Y=mx+b



All the lines (in “data space”) that go through a point (in data space)

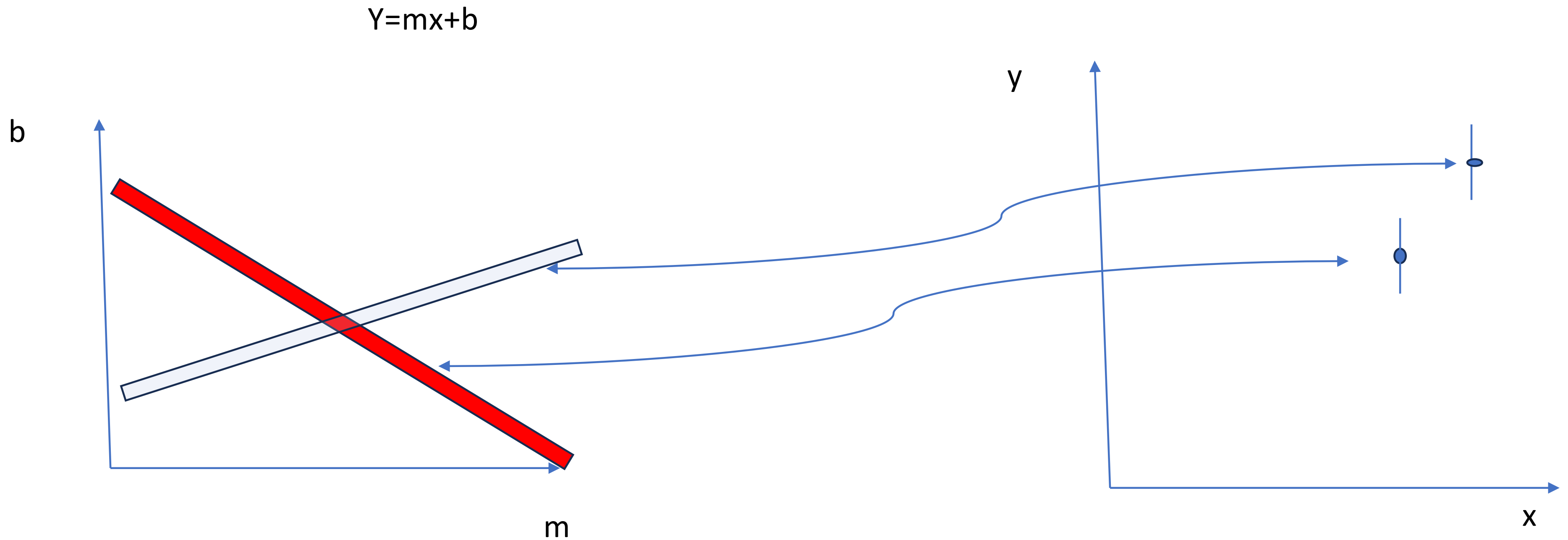
A (vertical) line segment of "pencils of lines"

Y=mx+b



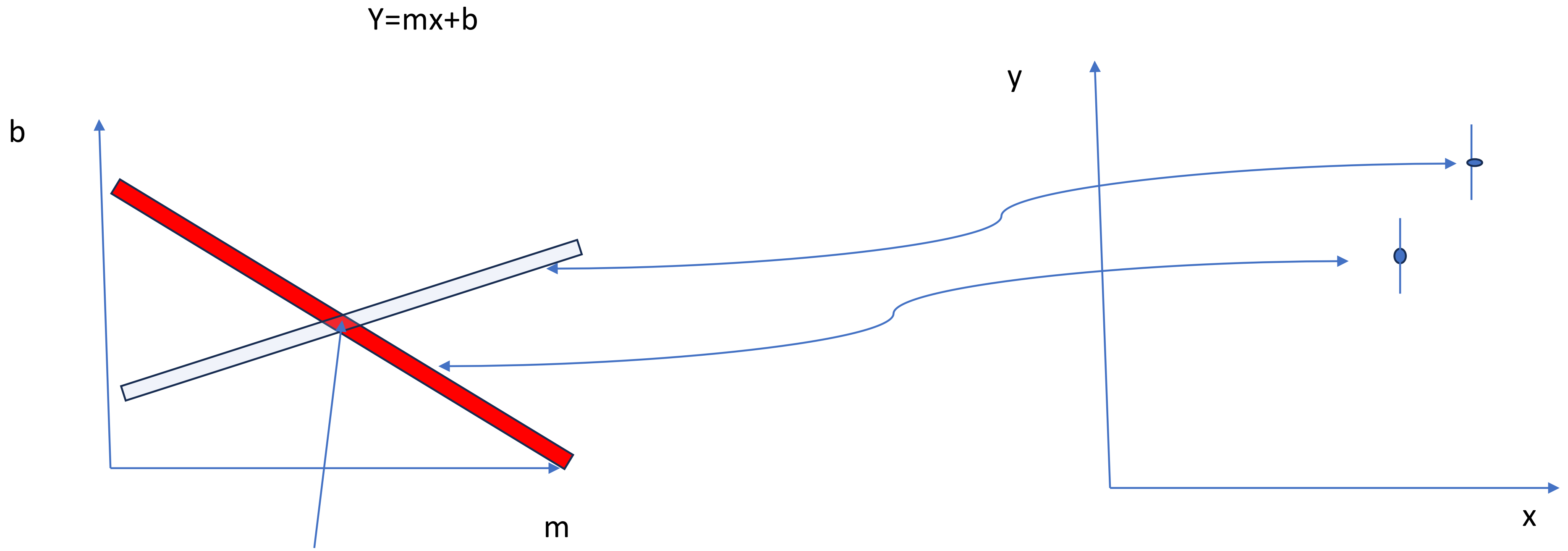
All the lines (in "data space") that go through a vertical line segment (in data space)

A (vertical) line segment of "pencils of lines"



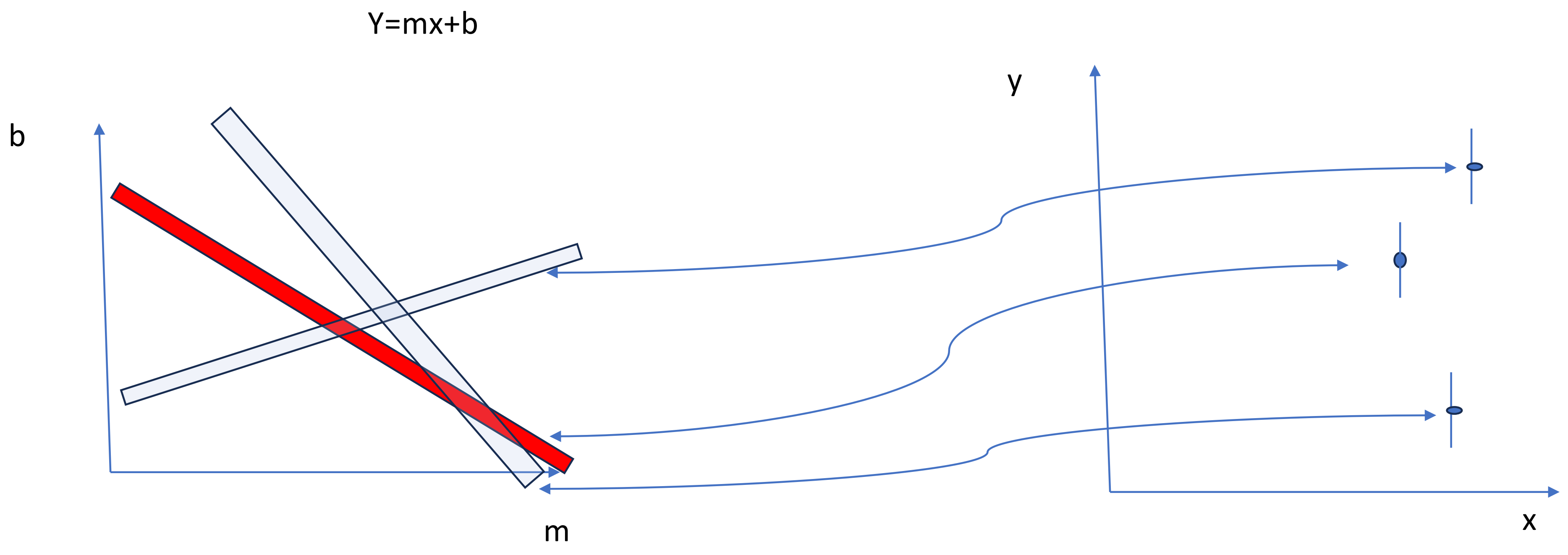
All the lines (in "data space") that go through a vertical line segment (in data space)

A (vertical) line segment of “pencils of lines”



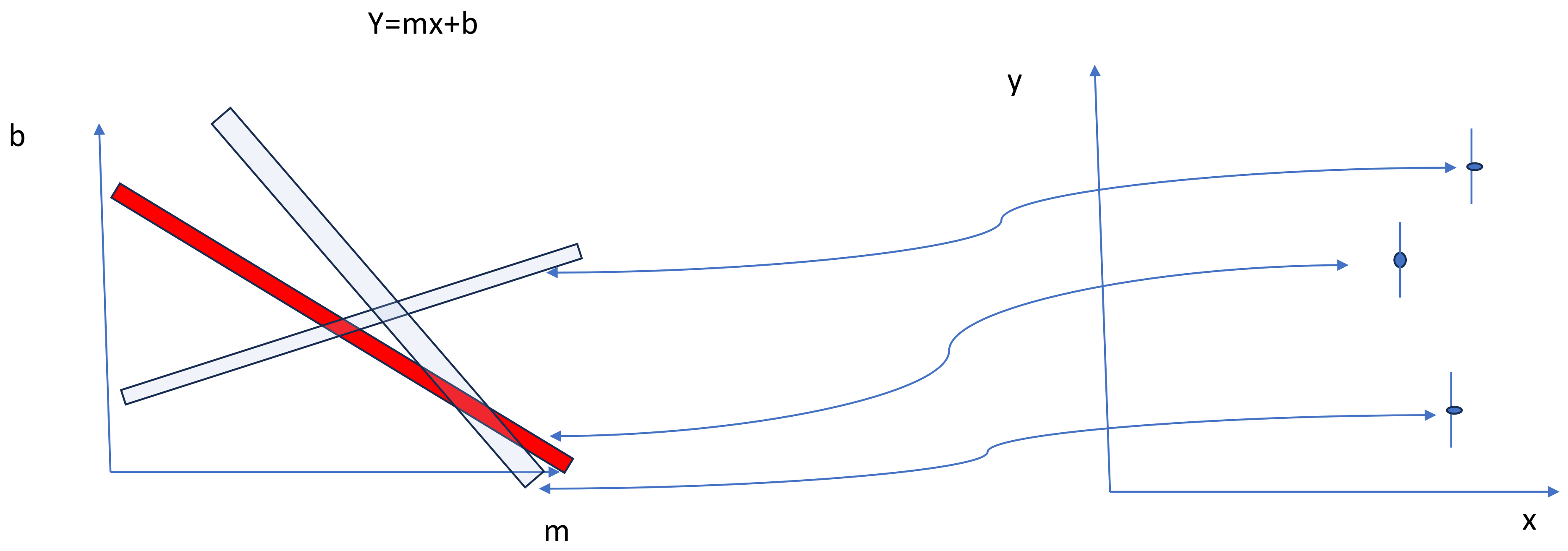
All the lines (in “data space”) that can pass within tolerance (using regression error) two points in data space

A (vertical) line segment of "pencils of lines"



The intersection model of (affine) line fitting (regression error) within tolerance "fat" line arrangements!

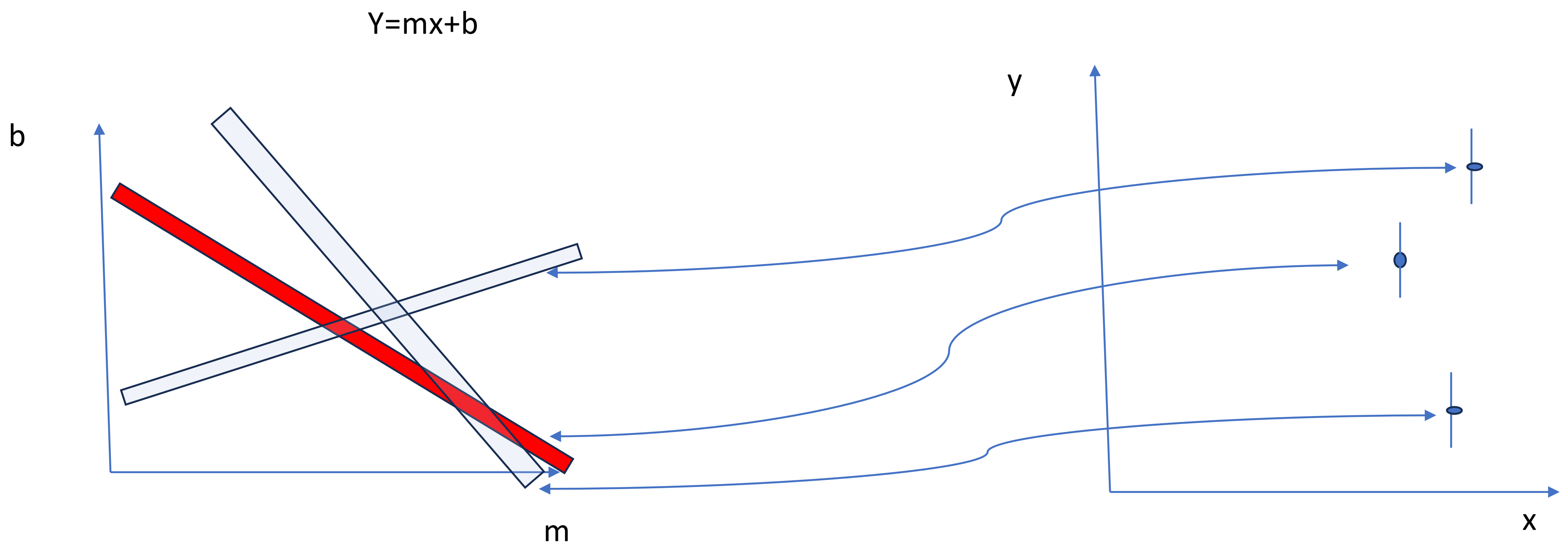
A (vertical) line segment of "pencils of lines"



The intersection model of (affine) line fitting (regression error) within tolerance "fat" line arrangements!

This is an example of an infeasible triplet of points – can you see why?

A (vertical) line segment of “pencils of lines”



The intersection model of (affine) line fitting (regression error) within tolerance “fat” line arrangements!

This is an example of an infeasible triplet of points – can you see why?

(this size of tolerance! If we increase that tolerance then it will become feasible)

But what does this have to do with AI, Machine Learning, Signal and Image Processing???

Image 1 (x,y,1) <-> Image 2 (x’,y’,1)

(x’,y’,w)^T=H(x,y,1)^T
H – 3x3 homography matrix

(image stitching)

Other “models” R,t (point cloud aignment)
F (3x3 matrix relating geometry of stereo vision)
x^T Fx=0
Etc.

(Permission to use photo via Daniel Barath)

Fundamental Matrix estimation

(Common pinhole camera assumption)

$x^T Fx'=0$ for matching pair of points $x=(x_1,x_2,1)$, $x'=(x'_1,x'_2,1)$
Note F is a 3x3 matrix BUT it is scale independent (hence 1 less degree of freedom) and det(F)=0 (hence another reduction in degrees of freedom).
If you “know” F (can extract from matching pairs) then you can derive the relative camera poses (up to a projective transformation). You can then usually use other information to recover Euclidean geometry – up to a scale. It has 7 parameters, effectively, but a nonlinear problem if posed as such.

If you have camera calibration information (or can guess reasonable camera calibration parameters then you work in terms of the Essential matrix – which has the same constraint eqn: $x^T Ex'=0$ where again, the matrix E is singular (it has two equal singular values and one zero singular values). Effectively 5 degrees of freedom. But again nonlinear.

In short, there are in fact a multitude of versions of camera geometry formulations depending on whether one wants to use a “linearized” version (and hence suffer issues that full constraints are not enforced and suffer heteroscedastic noise issues, and also needs ”more points” to uniquely determine) or use a nonlinear version (which requires less points, may be statistically much better, but requires complex polynomial solvers).

Fundamental Matrix estimation

(Common pinhole camera assumption)

Here we only focus/illustrate the linearized version(s)

a) Because they are in fact the popular
b) Because they are simpler to understand/analyse (at least from more points of view)
c) Because they more naturally fit a progression of “hyperplane fitting” problems from 1 D location estimation, 1D subspace in 2D regression, affine line fitting in 2D, plane fitting in 3D….and other linearized (or already linear) model fitting
d) With my focus on understand the hypergraph classes associated with computer vision problems – they are *probably* the easiest to understand.

The essence is to simply expand the constraint equation: giving a bilinear constraint; and treating each blinear term as a separate “data variable”.
e.g. $x_1x'_1\ f_{1,1}$+…………+1 $f_{3,3}$ = 0. 9 terms, four linear (one in each of x_1,x_2,x'_1,x'_2), one constant (in terms of the x’s) and four homogeneous bilinear ($x_1,x'_1,x_2x'_1,x_1x'_2,x_2x'_2$).

If ALL of the matches were correct AND accurate (AND not degenerate), every set of 9 matches would supply an independent constraint on the 9 entries of F and we would have a correct F (similar story for E)
by solving linear equations.

Fundamental Matrix estimation

(Common pinhole camera assumption)

BUT even of the matches were correct, they certainly are not infinitely accurate (small noise) and moreover we haven't enforced det(F)=0 nor used the fact that the scale of F is indeterminate. Both can cause problems (complex interactions with small noise) – and that even assuming the matches are correct!

There have been many variants proposed but a common one is the 8-point algorithm.
Fix ONE of the entries of F to be 1 (which we can do, since the scale is arbitrary - SO LONG as we are not unlucky to fix an entry that *should be* zero.(!)
And solve and then project onto rank 2.

It "can go wrong" for multiple reasons – (picking the wrong F entry to fix, degenerate matching points, inaccurate matches, totally erroneous matches) and the latter is where I have taken most interest – outliers!
Various fixes have been proposed (including the "obvious": try multiple times varying different F entries to fix, detecting degenerate configurations, least squares refinement – particularly after using RANSAC (the robust fitting method that introduced the MaxCon criterion for robust fitting!) to remover the biggest issue (typically) – outliers!

The main message for our purposes though is that the above framework more or less treats F estimation as hyperplane fitting in 9D.

Fundamental Matrix estimation

(Common pinhole camera assumption – 8-point linearized method assumed)

Every 8D hyperplane is given by 8 points in general position (ignore for now degeneracy).
But you will always “get an answer” – just as fitting a line to two points in 2D is not a really sensible thing to do when the data has noise.

If you know something about the likely noise size (tolerance to your data), then it is more sensible to fit to 9 or more points and outliers are unlikely (statistically) to fit within tolerance to many other selections of 8 points.

That is why I am interested in up to 9-uniform hypergraphs (and mostly linear models).

Robust Fitting (MaxCon) – Fitting within tolerance
(Given a model – line, plane, hyperplane,… and data points, what is the model that fist more points than any other model? Alternatively, what is the largest subset of points within tolerance of a model)

Problem	(Hyper)Graph class (every instance is a member of this class)
1D location estimation (equal tolerances/unequal tolerances)	Unit interval/Interval
1D subspace (in 2D) regression	Interval
1D line regression in 2D (affine line fitting in 2D)	3-uniform hypergraph (some subclass of – WHICH?) Intersection of ”fat” lines in 2D – but what class is that?
(2D) plane regression in 3D	4-uniform hypergraph (some subclass of – WHICH?) Intersection of fat planes in 3D
……… (7D) hyperplane regression in 8D (linear model/approximation of some projective geometry in computer vision)	9-uniform hypergraph (some subclass of – WHICH?) Intersection of fat hyperplanes in 3D

Note: the above are just *SOME* examples of the models. Some use geometric distance instead of regression distance – some are fitting “element of a group” (rigid body registration), some are in projective space and some “strange ”distance”).