Kernel-Size Lower Bounds: The Evidence from Complexity Theory

Andrew Drucker

IAS

Worker 2013, Warsaw

Part 1/3

Note

These slides are taken (with minor revisions) from a 3-part tutorial given at the 2013 Workshop on Kernelization ("Worker") at the University of Warsaw. Thanks to the organizers for the opportunity to present!

Preparation of this teaching material was supported by the National Science Foundation under agreements Princeton University Prime Award No. CCF-0832797 and Sub-contract No. 00001583. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Main works discussed

- [BDFH'07] H. Bodlaender, R. Downey, M. Fellows, and D. Hermelin: On problems without polynomial kernels. ICALP 2008, JCSS 2009. (Preprint '07)
- [FS'08] L. Fortnow and R. Santhanam: Infeasibility of instance compression and succinct PCPs for NP. STOC 2008, JCSS 2011.
- [DvM'10] H. Dell and D. van Melkebeek: Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. STOC 2010.
- [DM'12] H. Dell and D. Marx: Kernelization of packing problems. SODA 2012.
- [D'12] A. Drucker: New limits to classical and quantum instance compression. FOCS 2012.

Breakdown of the slides

- Part 1: introduction to the OR- and AND-conjectures and their use. Covers [BDFH'07], [DvM'10], [DM'12].
- Part 2: Evidence for the OR-conjecture [FS'08]
- Part 3: Evidence for the AND-conjecture (and OR-conjecture for probabilistic reductions) [D'12]

Big picture

- P vs. NP: The central mystery of TCS.
- Can't understand this problem, but would like to use

$$P \neq NP$$

hypothesis to "explain" why many tasks are difficult.

Big picture

- These talks: describe how (an extension of) $\mathsf{P} \neq \mathsf{NP}$ can explain hardness of kernelization tasks.
- Our focus: building the initial bridge between these two domains.
- [Many other papers]: clever reductions between kernelization problems, to show dozens of kernel lower bounds (LBs).

Outline

1. Introduction
2. OR/AND-conjectures and their use
3. Evidence for the conjectures

Outline

Problems and parameters

- Input: Formula ψ.

  Is ψ satisfiable?

- Parameters of interest:
  - total bitlength;
  - # clauses;
  - # variables;
  - can invent many more measures.

Problems and parameters

- Our view in these talks:
  - computational problems can have multiple interesting parameters.
  - won't define parameters formally, but always will be easily measureable.

$$x \longrightarrow k(x)$$

  - Insist: $k(x) \leq |x|$

FPT review

- A parametrized problem P with associated parameter k is Fixed-Parameter Tractable (FPT) if some algorithm solves P in time

$$f(k(x)) \cdot \text{poly}(|x|) \ .$$

Self-reductions and kernelization

- Self-reduction for problem P: a mapping R s.t.

  x a "Yes"-instance of P $\iff$ $R(x)$ a "Yes"-instance of P

- Goal: want $R(x)$ to be smaller than x.
- This talk: only interested in poly-time self-reductions.
- (Will also discuss reductions between param'd problems...)

Kernels

- Let F be a function.
- Poly-time self-reduction R is an $F(k)$-kernelization for P w.r.t. parameter k, if:

$$\forall x : \qquad |R(x)| \leq F(k(x))$$

.

- Output ("kernel") size bounded by function of the parameter alone!

Virtues of kernels

- $F(k)$-kernels for any (decidable) problem yields an FPT algorithm.
- Many natural FPT algorithms have this form.

Virtues of kernels

- If $F(k) \leq \text{poly}(k)$ and problem is in NP, we get an FPT alg. with runtime

$$\underbrace{\text{poly}(|x|)}_{\text{(compress the instance)}} + \underbrace{\exp(\text{poly}(k))}_{\text{(solve reduced instance)}} .$$

- $F(k) \leq \text{poly}(k)$: "Polynomial kernelization"

Virtues of kernels

- Kernelization lets us compress instances to store for the future.
- Also allows us to succinctly describe instances to a second, more powerful computer.

Virtues of kernels

- Many great kernelization algs; won't survey here...
- Which problems fail to have small kernels?

Kernelization limits

- For decidable problems:
  - $F(k)$-kernels implies FPT, so...
  - NOT FPT implies no $F(k)$-kernels for any F!
- E.g., k-Clique is W[1]-complete, so is not FPT or $F(k)$-kernelizable, unless

$$\text{FPT} = \text{W[1]}$$

Kernelization limits

Leaves possibility that all "natural" problems in FPT have poly(k)-kernels!

Kernelization limits

- A few kernel size LBs based on P $\neq$ NP...
- "Dual parameter" technique [Chen, Fernau, Kanj, Xia '05] shows that k-Planar Vertex Cover has no $1.332k$-kernels* unless P = NP.

* (only applies to reductions that don't increase k)

Kernelization limits

- A few kernel size LBs based on P $\neq$ NP...
- Similar results for kernels of restricted form, based on NP-hardness of approximation [Guo, Niedermeier '07].
- These bounds are all $\Theta(k)$.

Kernelization limits

- Lower bound tools were limited, until a paper of [Bodlaender, Downey, Fellows, Hermelin '07].
- Introduced "OR-" and "AND-conjectures,"

  showed that these would rule out poly(k)-kernels for many problems.
- Related, independent work in crypto: [Harnik, Naor '06]

Kernelization limits

- Many follow-up works showed the usefulness, versatility of the OR-conjecture for kernel LBs.
- We'll describe one important example:
  [Dell, Van Melkebeek '10]
  (and follow-up by [Dell, Marx '12])

Kernelization limits

- [Fortnow, Santhanam '08] and [D. '12] showed the OR, AND-conjectures follow from a "standard" assumption in complexity, namely

$$\mathrm{NP} \not\subseteq \mathrm{coNP/poly} \ .$$

- (We'll discuss this assumption...)

Kernelization limits

- We now have strong kernel-size LBs for most problems that resisted kernels. E.g.: unless NP $\subset$ coNP/poly:

  1. k-Path does not have poly(k)-kernels;
  2. Same for k-Treewidth;
  3. N-Clique (param. $N = \#$ vertices), which has a trivial N^2 kernel, does not have have kernels of size $N^{2-\varepsilon}$.

     (For d-uniform hypergraphs, we have the tight threshold N^d.)

Kernelization limits

- Before telling this story...

  Whats the real significance of these negative results?

Possible criticisms

"Kernelizations are assumed to be deterministic. That's too limited."

Agreed.

- In practice, almost all kernelizations we know are deterministic. But for meaningful lower bounds, we need to understand randomized ones as well.
- But—since [D.'12], our kernel LBs also apply to randomized algorithms.

Possible criticisms

"Kernelizations are assumed to map problem instances to instances of the same problem. That's also too limited."

- But all known kernel LBs for NP problems are insensitive to the target problem.
- They apply to "cross-kernelization" as well.

Possible criticisms

"Some applications of kernelization could be achieved under a broader definition. You're just ruling out one path to those goals."

Agreed.

- In particular, self-reductions which output many smaller instances (whose solutions yield a solution to the original instance) could be nearly as useful for FPT algs. [Guo, Fellows]
- We don't understand full power of these "Turing kernels" (yet!)
- Question explored by [Hermelin, Kratsch, Soltys, Wahlstrom, Wu '10].

Possible criticisms

- Kernelization also useful to succinctly transmit hard problems to a powerful helper. $\Rightarrow$ Natural to allow 2-way interaction.
- [Dell, Van Melkebeek '10]: boost our kernel LBs to communication LBs. (More general!)
- OPEN: extend to probabilistic communication.

Possible criticisms

"Ultimately, kernelization is just one approach to fast algorithms. Many of the LBs are for problems which already have good FPT algs."

...but this criticism also applies to kernel upper-bound research!

Many papers give kernels where good FPT results were already known.

The bottom line

Kernelization is a natural, rich algorithmic paradigm.

It's worthwhile and interesting to understand its strengths and limitations.

Outline

1. Introduction
2. OR/AND-conjectures and their use

The seed

- [Bodlaender, Downey, Fellows, Hermelin '07] got this project rolling.
- What core idea lies behind their work?

"Many param'd problems can express an OR of a large number of subproblems, without a blowup in the parameter.

"Those problems should resist small kernels... for a shared reason."

OR(L)

Let $L \subseteq \{0,1\}^*$. Define the problem OR(L) by:

- Input: a list $\langle x^1, \ldots, x^t \rangle$ of binary strings.
- Decide: is some $x^j \in L$?
- Parameter: $k := \max_j |x^j|$.

To ease discussion, let $\mathrm{OR}_=(L)$ be the special case where we require that

$$|x^i| = |x^j| = k \quad \forall i, j \ .$$

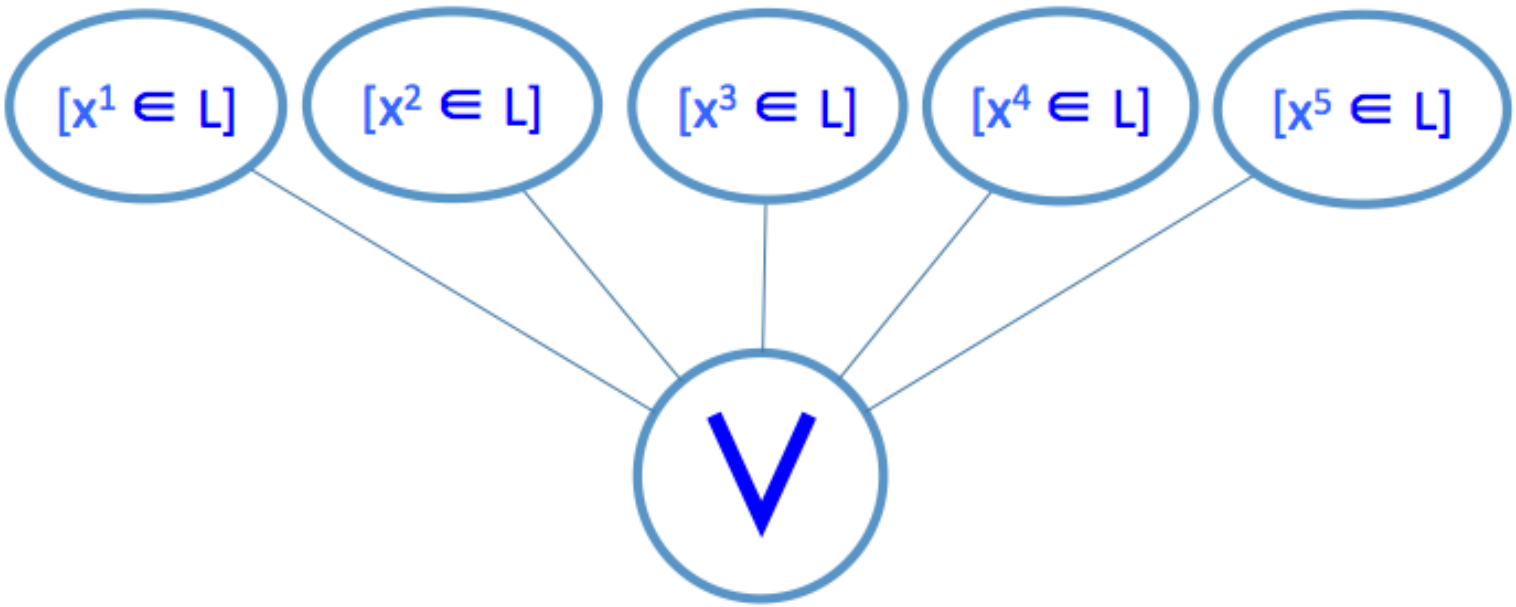
(even this special case resists small kernels.)

AND(L)

Define the problem $\mathrm{AND}_{=}(L)$ by:

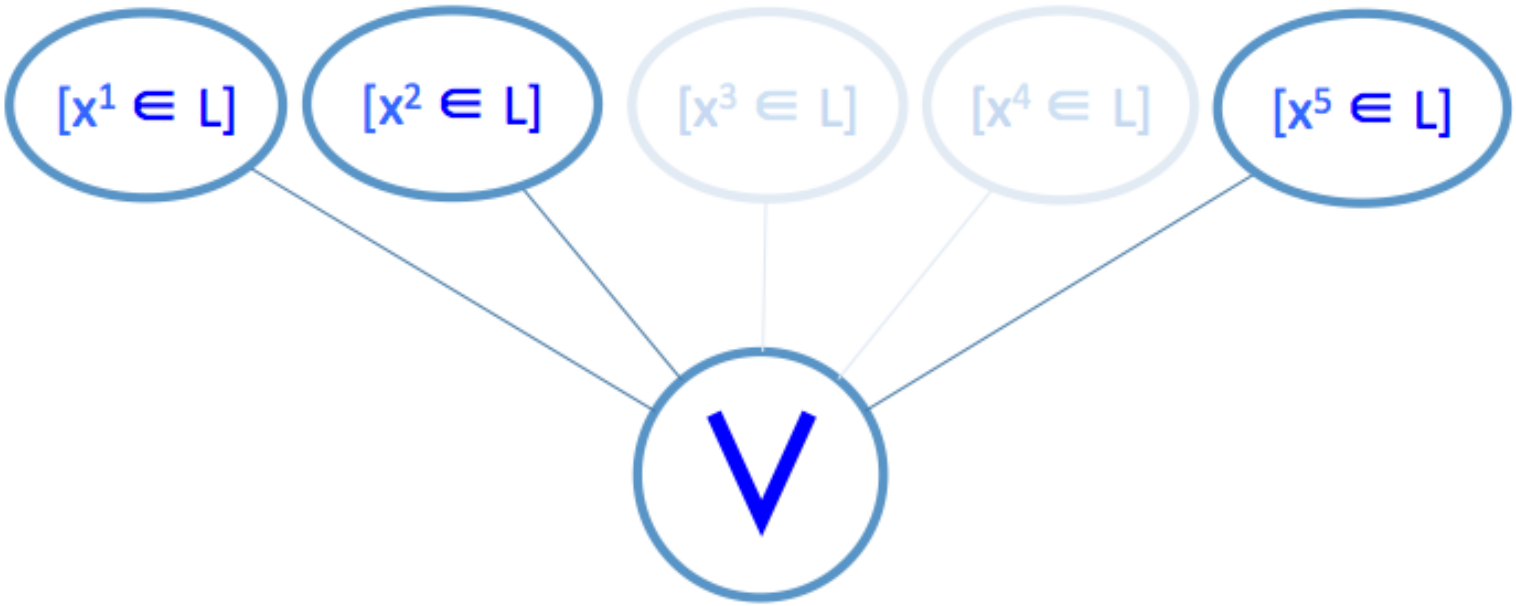
- Input: a list $\langle x^1, \ldots, x^t \rangle$ of binary strings.
- Decide: is every $x^j \in L$?
- Parameter: $k := |x^i| = |x^j|, \qquad j = 1, 2, \ldots, t.$

One approach to kernelization: sparsification



May try to identify instances that are "logically least-likely" to lie in L, remove them.

One approach to kernelization: sparsification



May try to identify instances that are "logically least-likely" to lie in L, remove them.

One approach to kernelization: sparsification

- Reasonable idea, but no nontrivial kernel size bounds known for $\mathrm{OR}_{=}(L)$, $\mathrm{AND}_{=}(L)$...
- No equivalence between the two tasks either!

The conjectures of [BDFH'07]

Let L be any NP-complete language.

- OR-conjecture: The problem $\mathrm{OR}_=(L)$ does not have $\mathrm{poly}(k)$-kernels.

- AND-conjecture: The problem $\mathrm{AND}_=(L)$ does not either.

Slightly "massaged" forms of the conjectures.

[BDFH'07]: equivalent to respective conjectures for $\mathrm{OR}(\mathrm{SAT}), \mathrm{AND}(\mathrm{SAT})$ (but we won't need this)

Consequences

[BDFH'07]: The OR-conjecture $\Longrightarrow$ none of these problems have poly(k)-kernels:

- k-Path, k-Cycle, k-Exact Cycle and k-Short Cheap Tour,
- k-Graph Minor Order Test and
  k-Bounded Treewidth Subgraph Test,
- k-Planar Graph Subgraph Test and
  k-Planar Graph Induced Subgraph Test,
- (k, σ)-
  Short Nondeterministic Turing Machine Computation,
- w-Independent Set, w-Clique and w-Dominating Set.

Dozens more in later works.

Consequences

[BDFH'07]: The AND-conjecture $\Longrightarrow$ none of these problems have poly(k)-kernels:

- k-Cutwidth, k-Modified Cutwidth, and k-Search Number,
- k-Pathwidth, k-Treewidth, and k-Branchwidth,
- k-Gate Matrix Layout and k-Front Size,
- w-3-Coloring and w-3-Domatic Number,

Connections

- OR, AND-conjectures connect to specific parametrized problems through various technical lemmas and reductions.
- Here we explain one of the simplest such connections.[1] Still quite powerful.

[1] Related to definitions in [Harnik-Naor '06], [BDFH'07], [Bodlaender, Jansen, Kratsch '11]

Connections

Claim

Let L be NP-complete, P a parametrized problem, and suppose there is a poly-time reduction R from an instance $\overline{x}$ to $\mathrm{OR}_=(L)$ to an equivalent instance of P, with

$$k(R(\overline{x})) \leq \mathrm{poly}(k(\overline{x})) .$$

Then, if P has some $\mathrm{poly}(k)$-kernelization A, so does $\mathrm{OR}_=(L)$ (and the OR-conjecture fails).

Proof.

To kernelize an instance $\overline{x}$ of $\mathrm{OR}_=(L)$:

$$\overline{x} \to R(\overline{x}) \to A(R(\overline{x})) \to (\text{reduce to } L) .$$

□

Connections

Claim

Let L be NP-complete, P a parametrized problem, and suppose there is a poly-time reduction R from an instance $\overline{x}$ to $\mathrm{AND}_{=}(L)$ to an equivalent instance of P, with

$$k(R(\overline{x})) \;\leq\; \mathrm{poly}(k(\overline{x}))\ .$$

Then, if P has some $\mathrm{poly}(k)$-kernelization A, so does $\mathrm{AND}_{=}(L)$ (and the AND-conjecture fails).

Proof.

To kernelize an instance $\overline{x}$ of $\mathrm{AND}_{=}(L)$:

$$\overline{x} \;\to\; R(\overline{x}) \;\to\; A(R(\overline{x})) \;\to\; (\text{reduce to } L)\ .$$

□

Using the connections

Let's see some (easy) examples.

Define k-Path:

- Input: $\langle G, k \rangle$.
- Decide: does G have a simple path of length k?

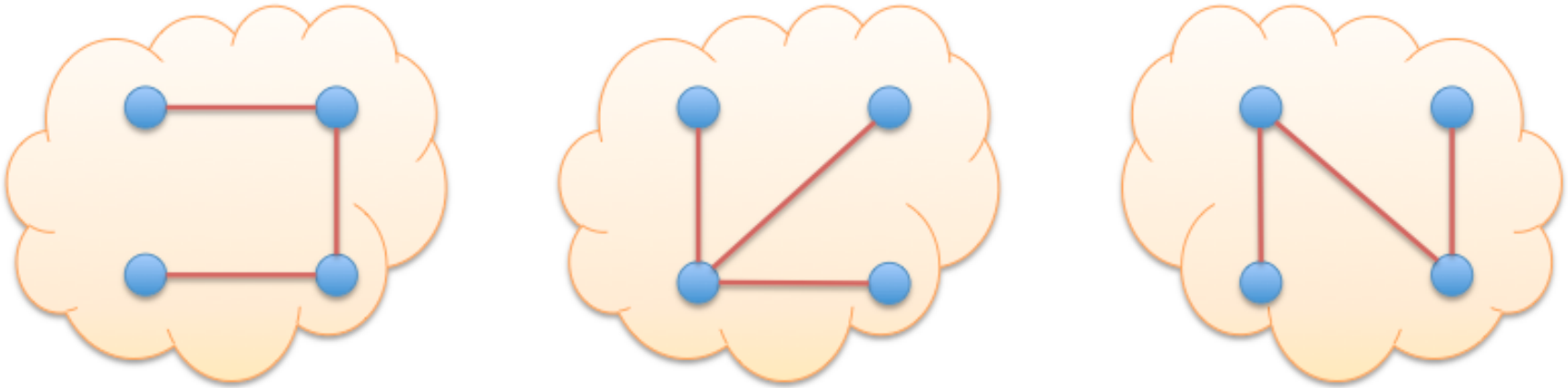
k-Path is FPT, runtime $2^{O(k)}\,\mathrm{poly}(n)$ [Alon, Yuster, Zwick '95].

But no $\mathrm{poly}(k)$-kernel known.

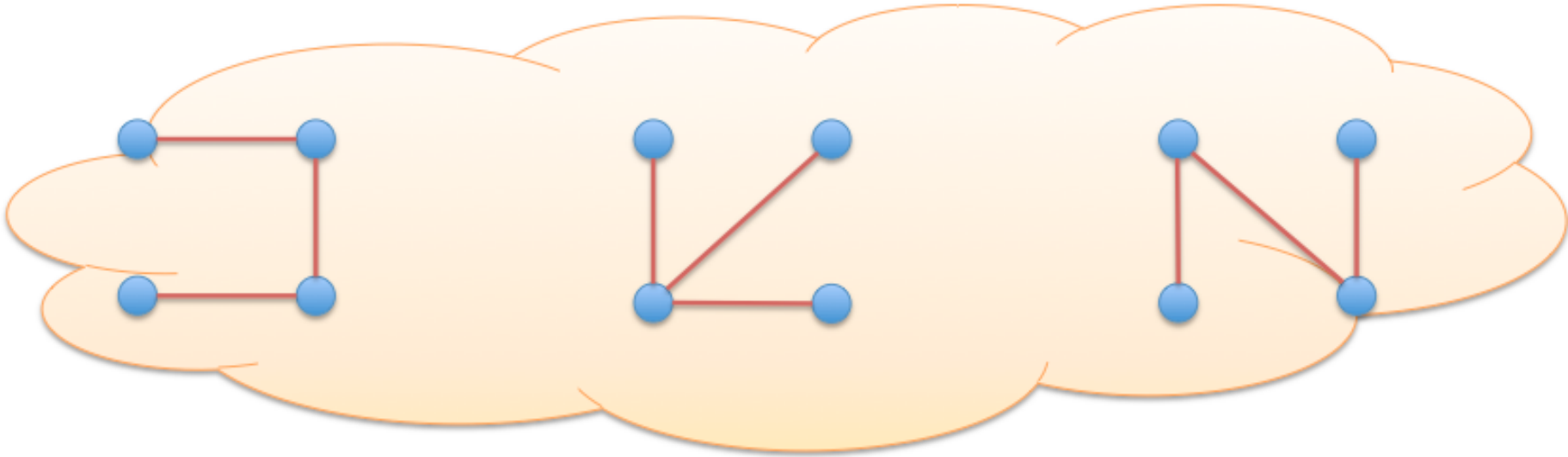
Using the connections

- Apply our Claim to show that k-Path is hard to kernelize.
- How to express an OR of many NP instances of size s, as a single instance of k-Path, with $k \approx s$?
- Which L to choose?

Using the connections



Using the connections



Using the connections

- Take $L =$ HAMILTONIAN PATH.
- Use graph encoding where length-k^2 input encodes a graph on k vertices.
- On the input $\langle G_1, \ldots, G_t \rangle$ to $\mathrm{OR}_{=}(L)$ (where $|\langle G_i \rangle| = k^2$), output

$$\left\langle H := \dot{\bigcup} G_j \ , \ k \right\rangle .$$

- H has a simple k-path $\iff$ some G_i has a Ham path.

Using the connections

Define k-Treewidth:

- Input: $\langle G, k \rangle$.
- Decide: does G have treewidth* $\leq k$?

*(tw = a monotone measure of graph "fatness")

Treewidth an important graph complexity measure for FPT alg. design.

k-Treewidth FPT, but treewidth is NP-hard to compute exactly.

Using the connections

- Given graphs $G_1, \ldots, G_t$, if

$$H := \dot{\bigcup} G_j$$

then

$$\mathrm{tw}(H) := \max_i(\mathrm{tw}(G_i)) ,$$

so

$$\mathrm{tw}(H) \leq k \iff \bigwedge_i [\mathrm{tw}(G_i) \leq k] .$$

- Basis of proof that (AND-Conjecture) $\implies$ k-Treewidth has no poly(k)-kernels. We take

$$L := \{\langle G \rangle : \mathrm{tw}(G) \leq |V(G)|/2 \} .$$

Using the connections

- In k-Treewidth, k-Path examples, choice of NP-complete language L was "obvious," closely related to param'd problem.
- Time has shown: sometimes "best" choice of L is not obvious, makes reductions easier.

(We'll see an example...)

Tight poly kernel LBs

- We've seen a strong framework for ruling out poly(k) kernels (modulo the AND-/OR-conjectures).
- Shortly after [BDFH'07], Fortnow and Santhanam showed OR-conjecture is true (for deterministic algorithms) if NP $\not\subseteq$ coNP/poly. (We'll come back to this...)
- Dell and Van Melkebeek built on [BDFH'07,FS'08] ideas to give tight kernel LBs for problems that do have poly(k)-kernels.
- How??

Tight poly kernel LBs

- First step: studied [FS'08] carefully!
- [FS'08] implicitly shows something much stronger than the OR-conjecture.
- Important to know this...

Restricting t

- Recall $\mathrm{OR}_{=}(L)$: given $x^1, \ldots, x^t$ each of length k, compute

$$\bigvee_{i \in [t]} [x^i \in L] \, .$$

- Let $t(k)$ be a function, and let $\mathrm{OR}_{=}(L)^{t(\cdot)}$ be the same problem where we further restrict $t = t(k)$.

- Focus on "reasonable" $t(\cdot)$: easily computable, and satisfy

$$t(k) \leq \mathrm{poly}(k) \, .$$

Stronger bounds

Theorem

[FS'08, implicit]: Assume $\mathrm{NP} \not\subseteq \mathrm{coNP/poly}$. *If* L *is* NP*-complete and* $t(k) \leq \mathrm{poly}(k)$, *no poly-time reduction* R *from* $\mathrm{OR}_{=}(L)^{t(\cdot)}$ *to any other problem can achieve output size*

$$|R(\overline{x})| \;\leq\; O(t \log t)\ ,$$

where $t = t(k)$, $k = k(\overline{x})$.

- E.g., take $t(k) := k^{100}$.
- Then input $(x^1, \ldots, x^{t(k)})$ to $\mathrm{OR}_{=}(L)^{t(\cdot)}$, of size $k \cdot k^{100}$, cannot be reduced to a kernel of size $< k^{100}$!
- Here, $\mathrm{OR}_{=}(L)^{t(\cdot)}$ trivially has a k^{101}-kernel, and by [FS'08] it is nearly optimal!

Stronger bounds

- Corollary: the k-Path problem on k^{100}-vertex graphs does not have kernels of size k^{100}.

- So [DvM'10] not really the first ones to prove good fixed-poly kernel LBs.
  Their achievement:

  1. Express $\mathrm{OR}_{=}(L)$ instances very efficiently within a parametrized problem instance, minimizing parameter blowup;

  2. Find a way to "boost" the [FS'08] bound and get truly tight results.

N-Clique

Define the N-Clique problem as:

- Input: $\langle G, s \rangle$.

  (G a graph on N vertices; $s \leq N$)

- Decide: Does G have a clique of size s?

- Parameter: N.

- Natural input size: up to N^2.

  Can we compress?

N-Clique

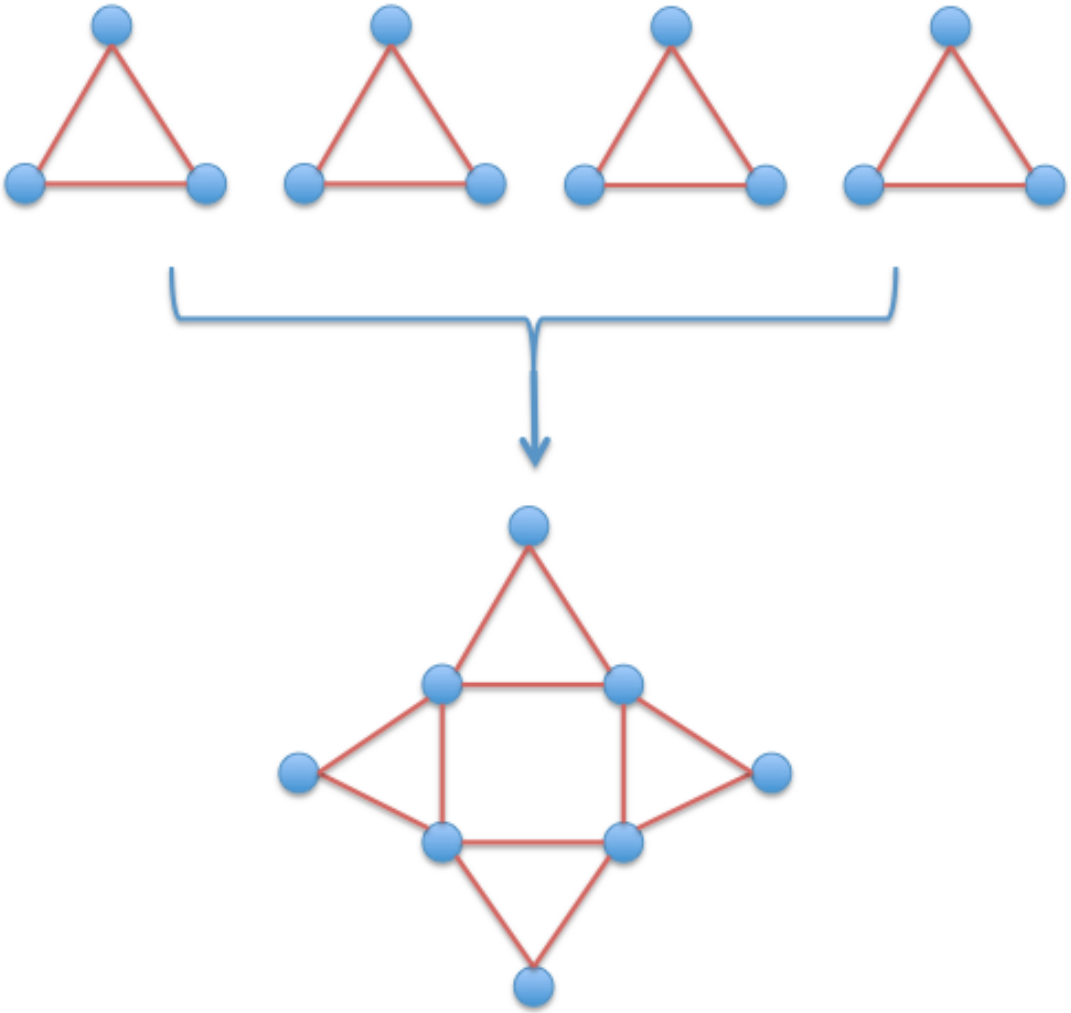
- For kernel LBs, goal is to efficiently express an OR of NP instances within an N-Clique instance.
- Can easily express an OR of NP instances by $\dot{\bigcup}$ ...
- Problem: blows up the parameter N linearly!

  Wasteful, since most potential edges are not used...

N-Clique

- Idea: "Pack" many CLIQUE instances into one graph, in a way that creates no large "unwanted" cliques.
- Main effort: Find special "host" graph to contain these instances.

Edge-disjoint clique packing: example



The packing lemma

Lemma (Packing Lemma for graphs, DvM '10)

For any $s, t > 0$ there is a graph G^* on

$$s \cdot (s + t^{.5+o(1)})$$

vertices. $E(G^*)$ is union of t edge-disjoint cliques $K_1, \ldots, K_t$ of size s, and has no other ("unwanted") s-cliques.
G^* can be constructed in time $\text{poly}(s + t)$.

- With this lemma we can "embed" t instances of an appropriate problem into $K_1, \ldots, K_t$ respectively.

(Details...)

The packing lemma

Lemma

For any $s, t > 0$ there is a graph G^* on

$$s \cdot (s + t^{.5+o(1)})$$

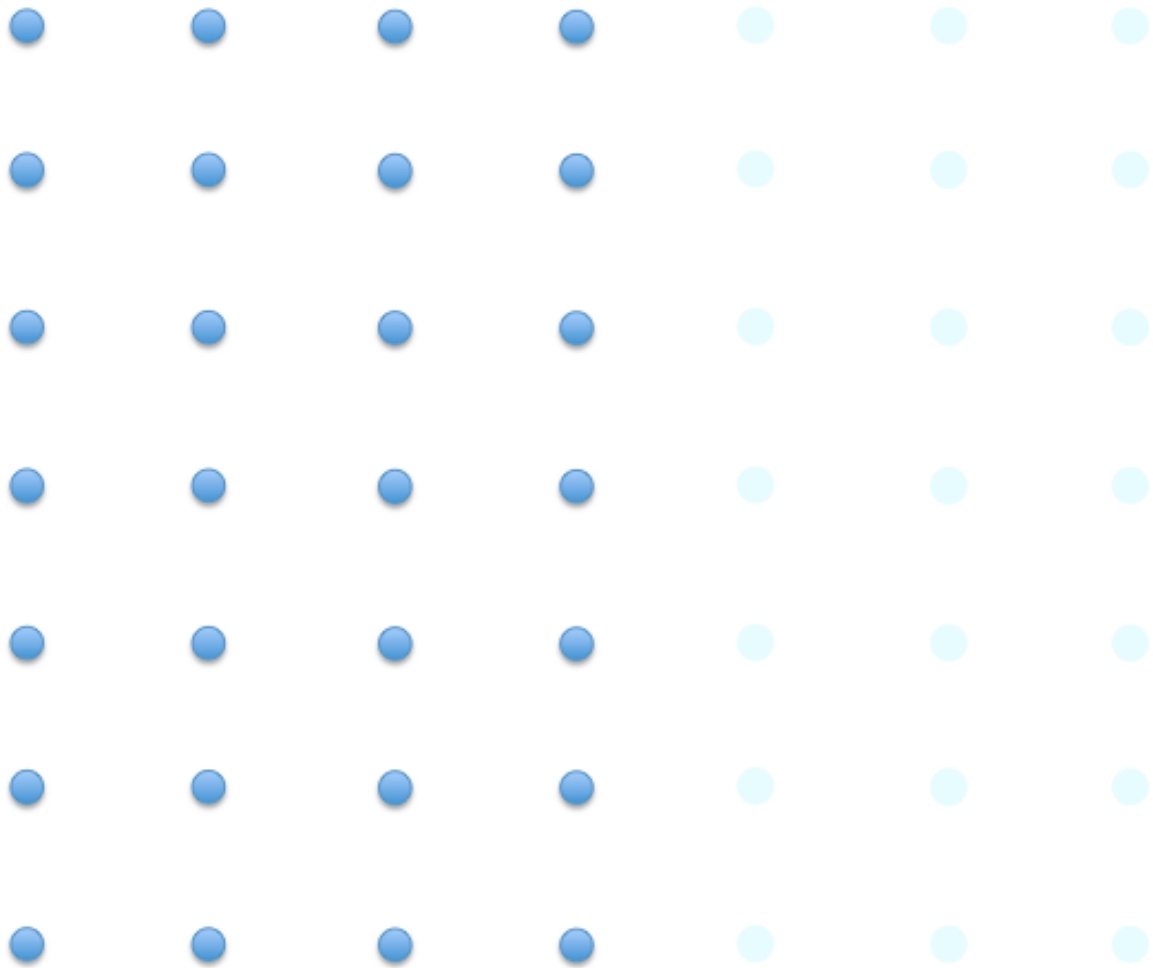
vertices. $E(G^*)$ is union of t edge-disjoint cliques $K_1, \ldots, K_t$ of size s, and has no other ("unwanted") s-cliques.

- Cliques K_i, K_j can intersect in at most one vertex.
- Suggests we consider them as lines in a (finite) plane...

The packing lemma

- Fix any prime $p > s$.
- We'll build a graph G with sp vertices, and see how large we can take t...
- Vertex set: $\mathsf{LP} = \mathbb{F}_p \times \{0, 1, \ldots, s-1\}$. ("left-plane")

The "left-plane"



Lines and line-cliques

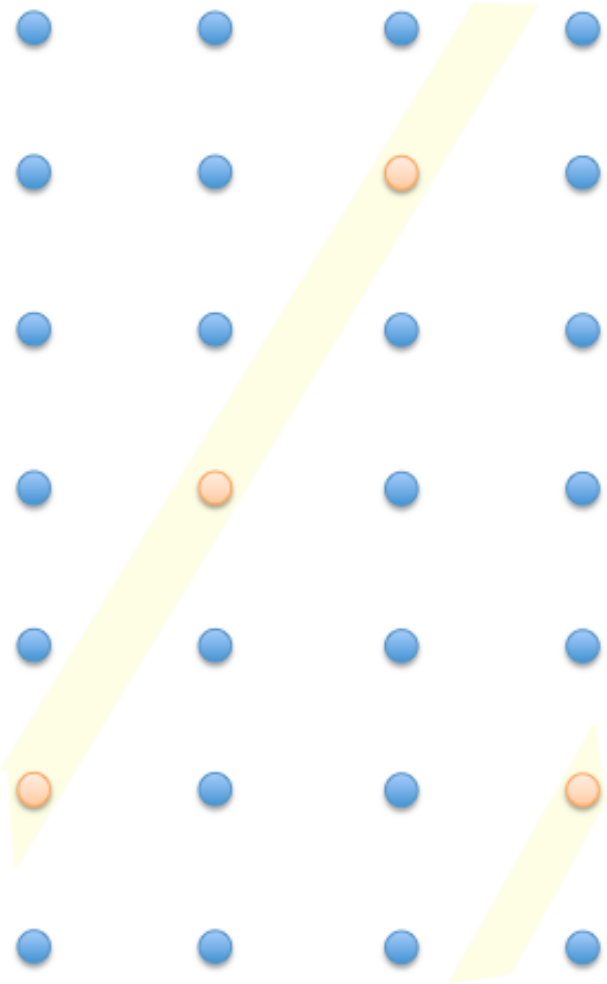
- line in LP:

$$\ell_{[a,b]} \;=\; \{(x,y) \in \mathrm{LP} :\; y = ax + b\} \qquad (a, b \in \mathbb{F}_p, a \neq 0)$$

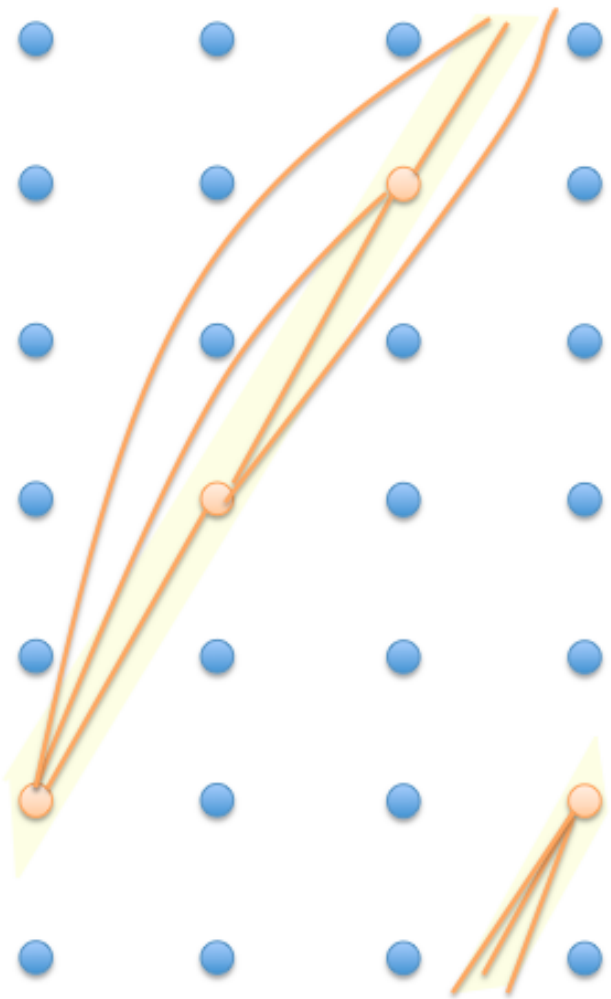
- For each line ℓ define

$$K_\ell \;=\; \left\{ \Big((x,y)\;,\; (x',y') \Big) \;:\; (x,y),(x',y') \in \ell \right\}$$

Lines and line-cliques



Lines and line-cliques



The packing lemma

- $K_\ell, K_{\ell'}$ are edge-disjoint as needed.
  (two points contained in unique line)
- Each K_ℓ is a clique of size s!
  (so sp cliques placed in total.)
- Problem: many other s-cliques...

The packing lemma

- Inspired idea: restrict the slope of lines we use.
- Choose "special" $A \subset \mathbb{F}_p^*$, and take

$$E(G^*) = \bigcup_{\text{slope}(\ell) \in A} K_\ell \ .$$

- $A =$ large set without length-3 arithmetic progressions (3-APs).

The packing lemma

- Key claim: The only s-cliques in G^* are the K_ℓ's we included.
  1. Why true?
  2. What does it get us?

What does key claim give?

- In our construction, we packed $p \cdot |A|$ cliques of size s into G^*.

Theorem (Salem, Spencer '42)

There is a 3-AP-free set $A \subset \mathbb{F}_p^*$ with

$$|A| \geq p^{1-o(1)},$$

constructible in time $\text{poly}(p)$.

- So to pack t cliques, we may take $p \leq s + t^{.5+o(1)}$.
- Number of verts. $N = sp \leq s(s + t^{.5+o(1)})$, as needed!

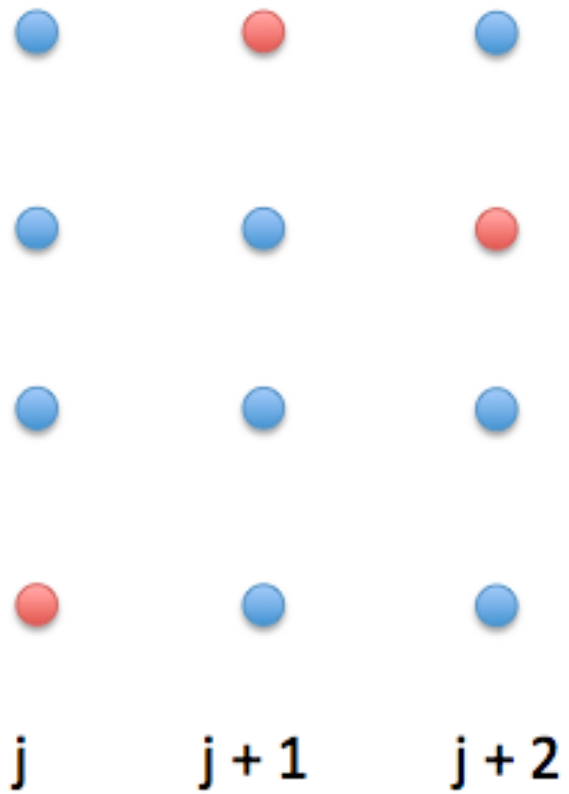
Why is key claim true?

- First, no "vertical" edges $((x, y), (x, y'))$ in G^*.
- Thus any s-clique must have one element from each "column"

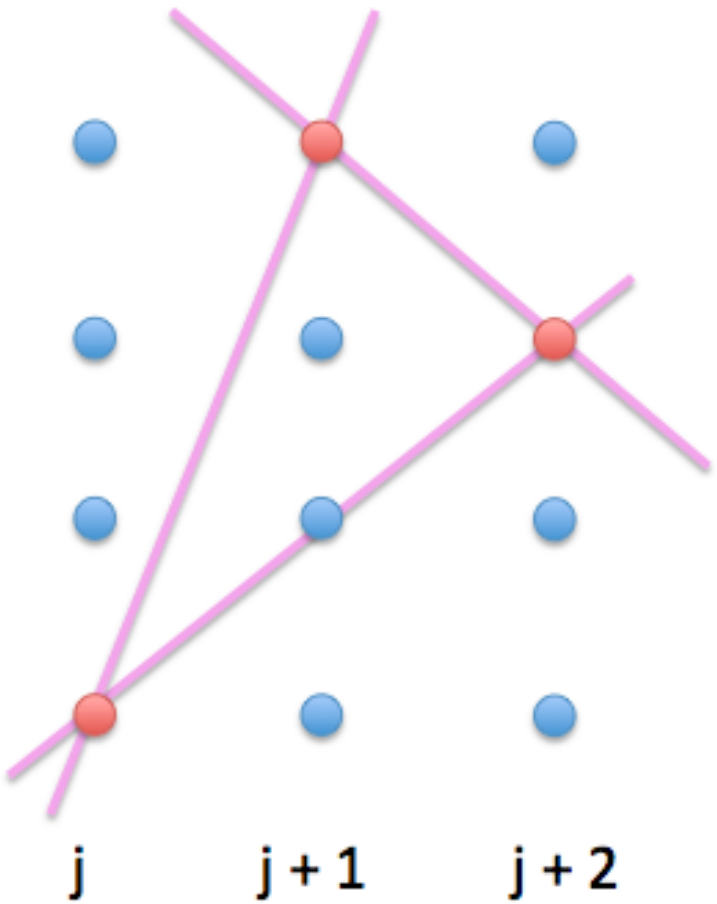
$$\mathrm{col}_j = \mathbb{F}_p \times [j] .$$

Why is key claim true?

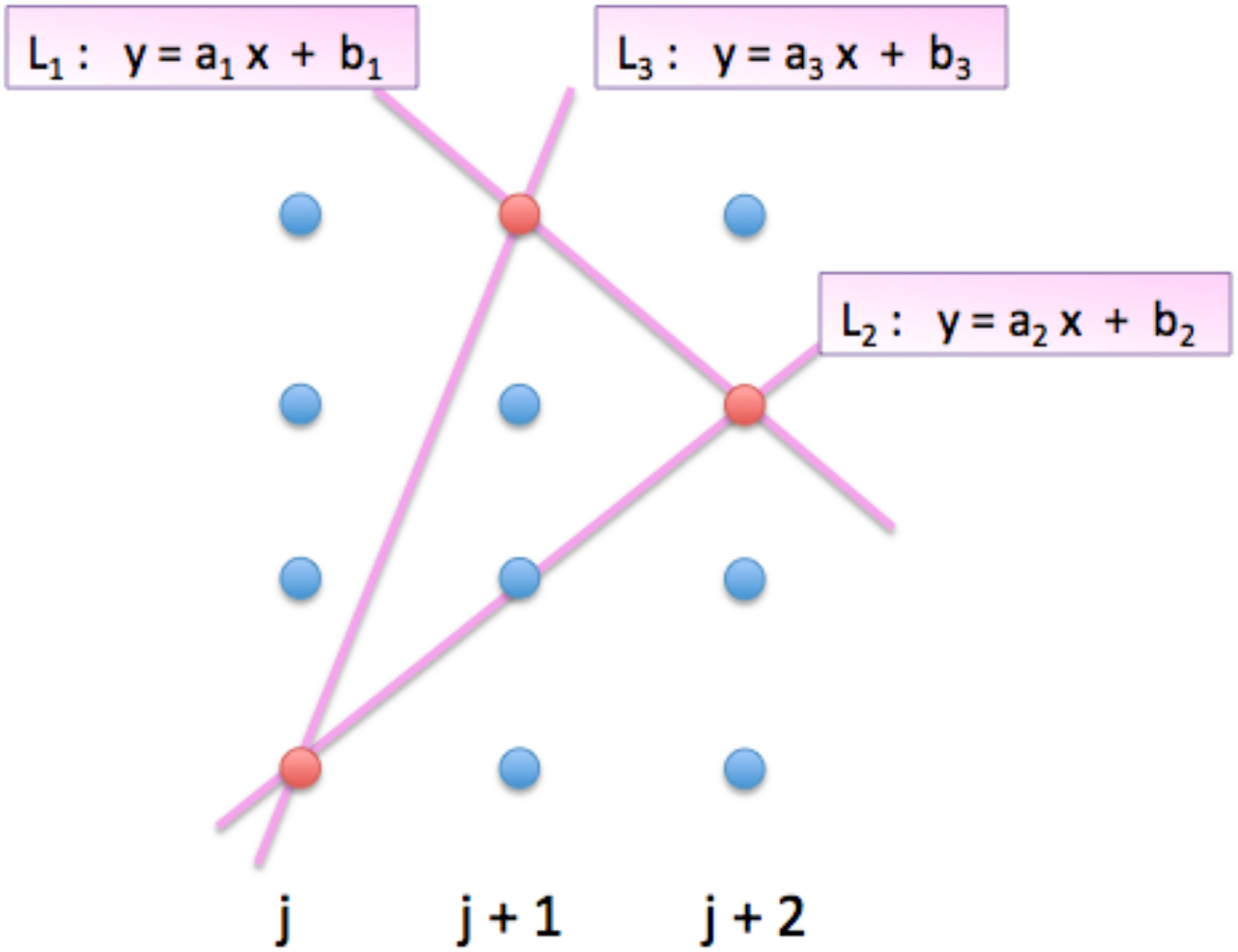
- If K_{bad} is an s-clique in G^* not equal to some K_ℓ, then $\exists$ some three adjacent columns like so:



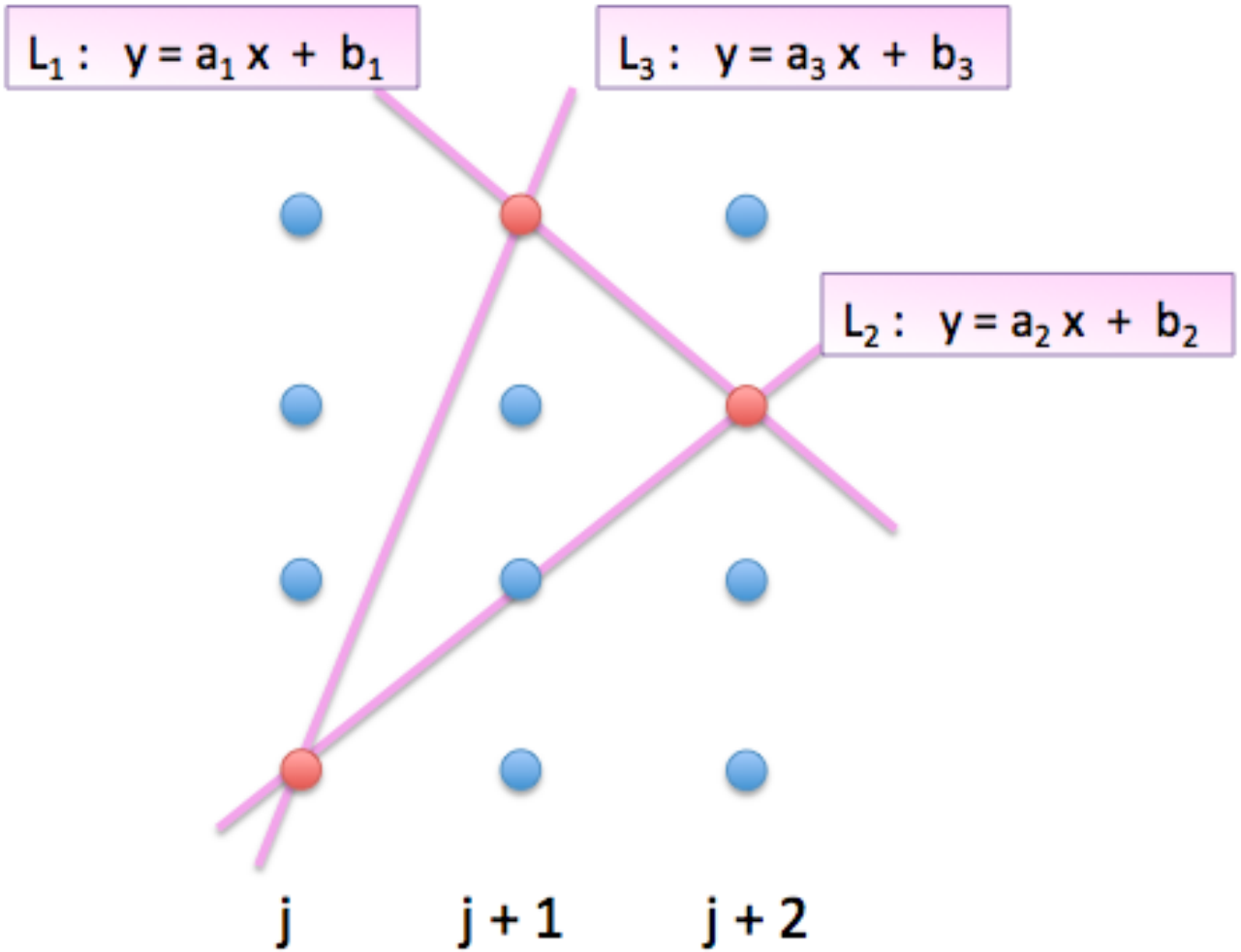
Why is key claim true?



Why is key claim true?

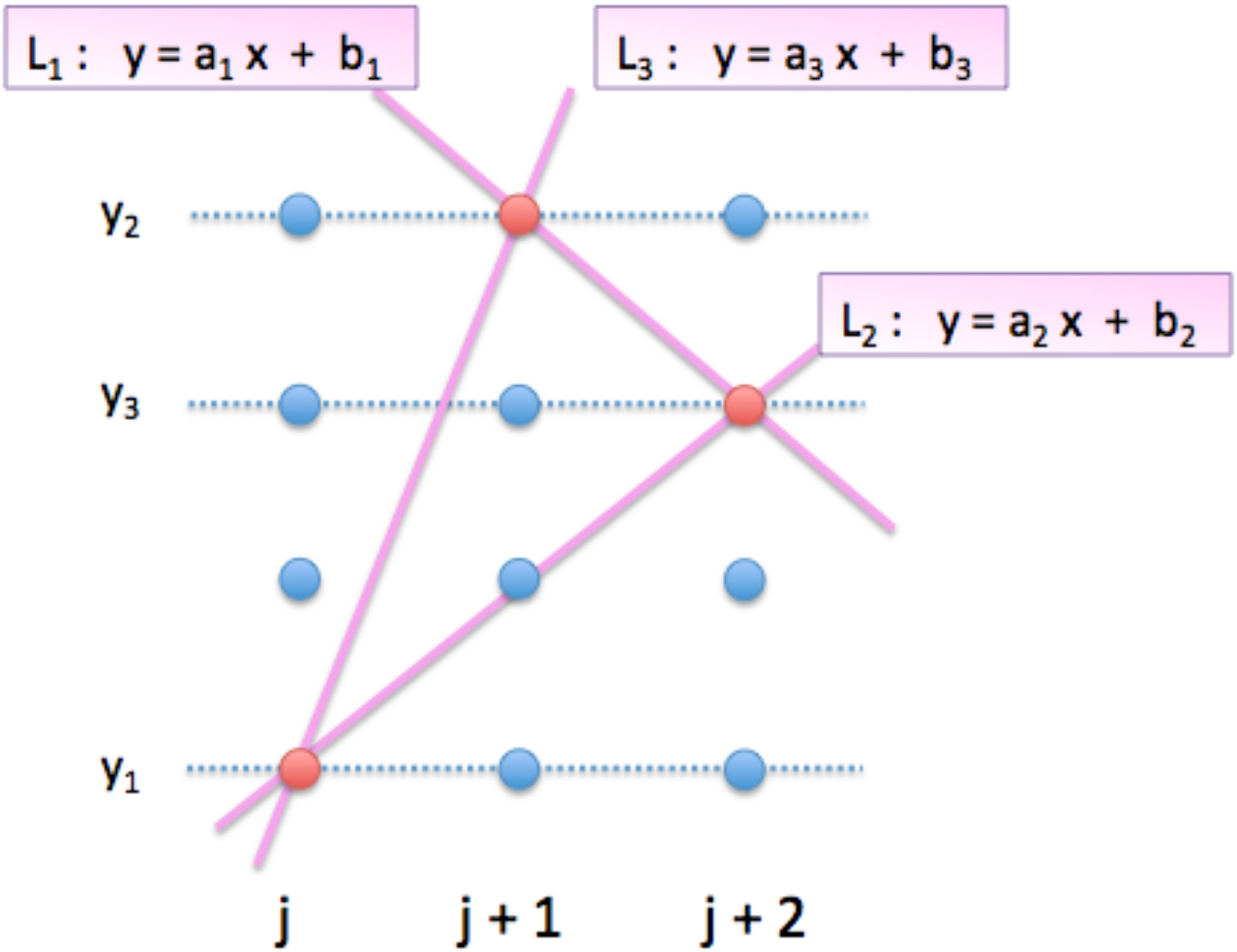


Why is key claim true?



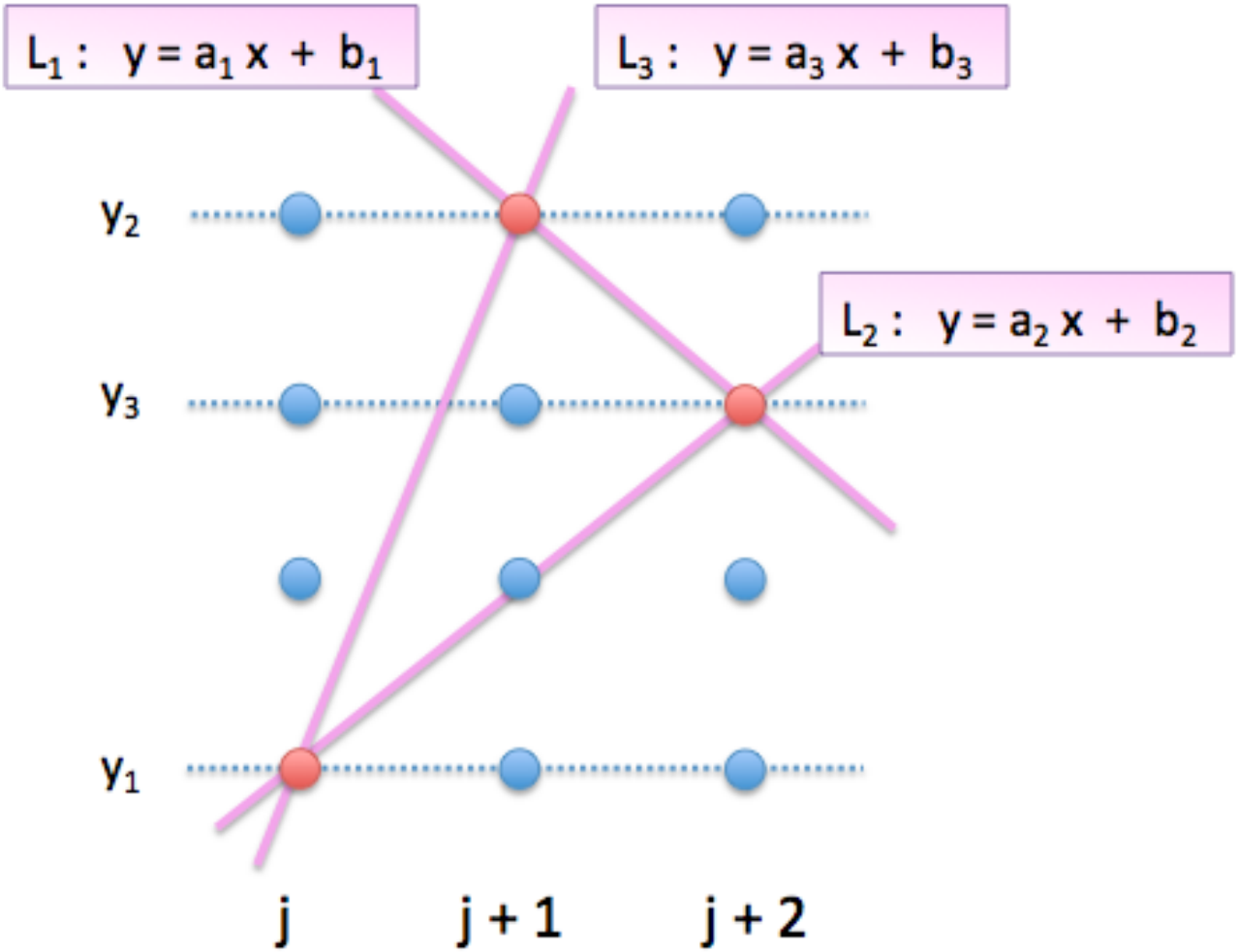
Claim: $a_1 + a_3 = 2a_2$

Why is key claim true?



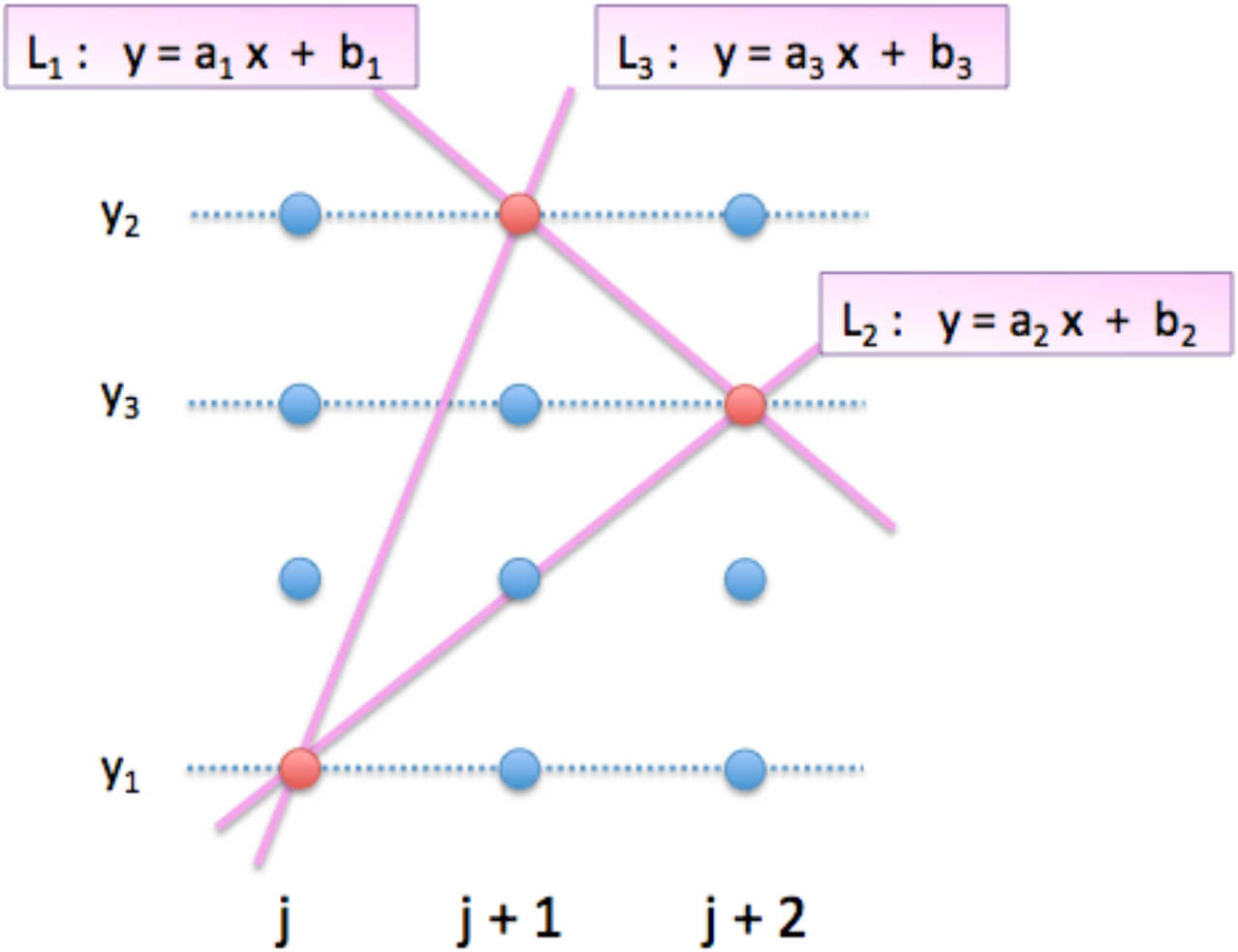
Claim: $a_1 + a_3 = 2a_2$

Why is key claim true?



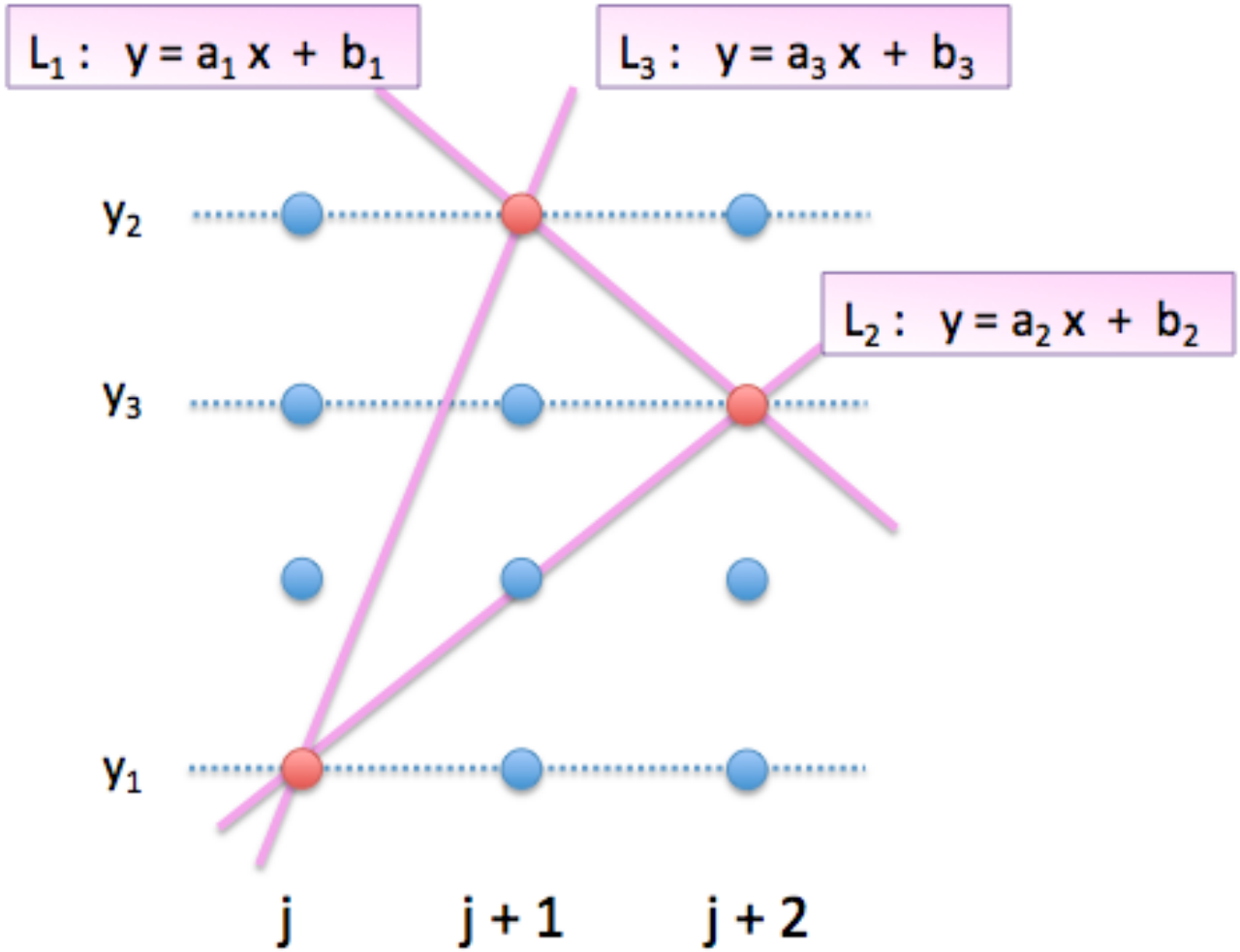
$y_2 - y_1 = a_3$

Why is key claim true?



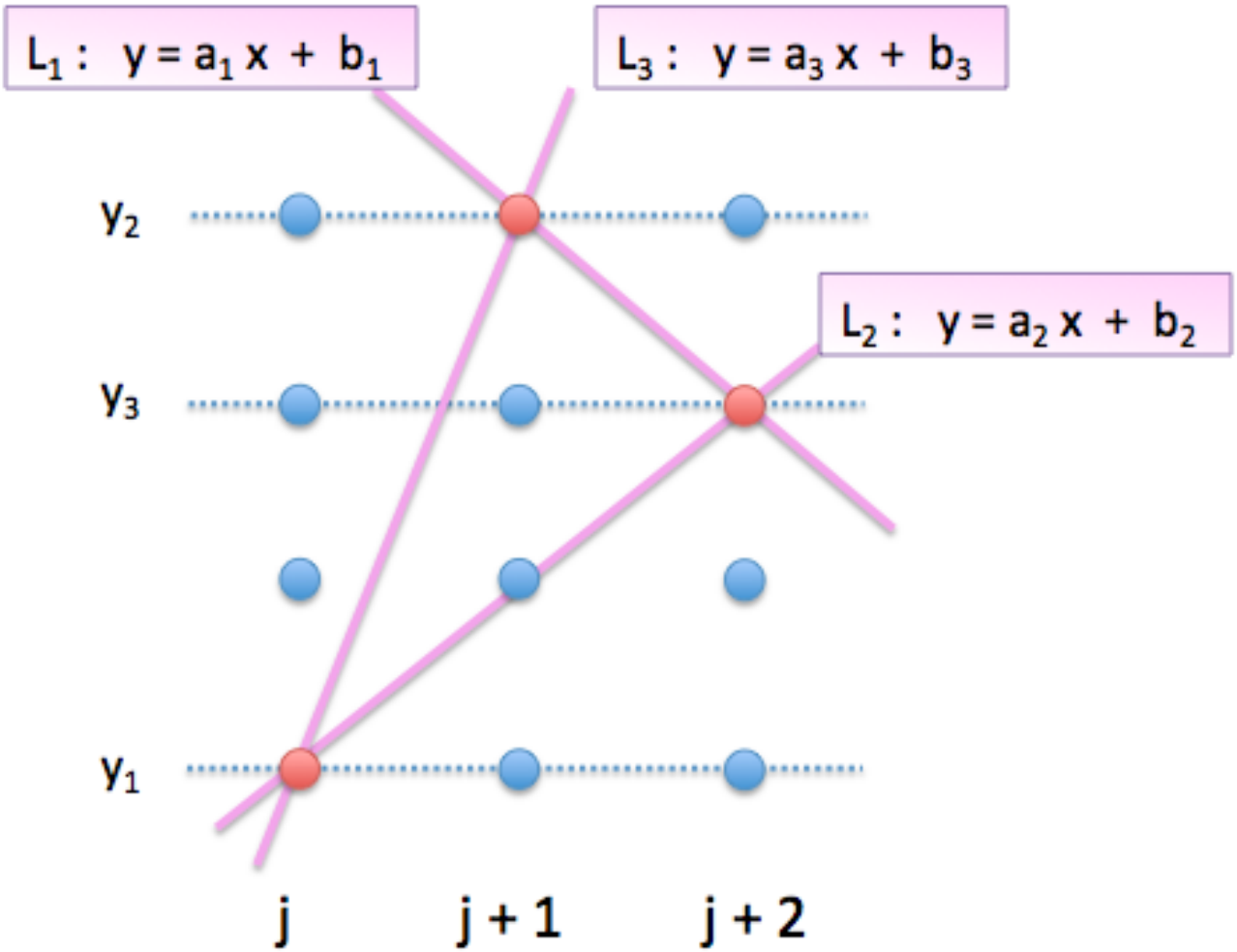
$y_2 - y_1 = a_3$ $\qquad$ $y_3 - y_2 = a_1$

Why is key claim true?



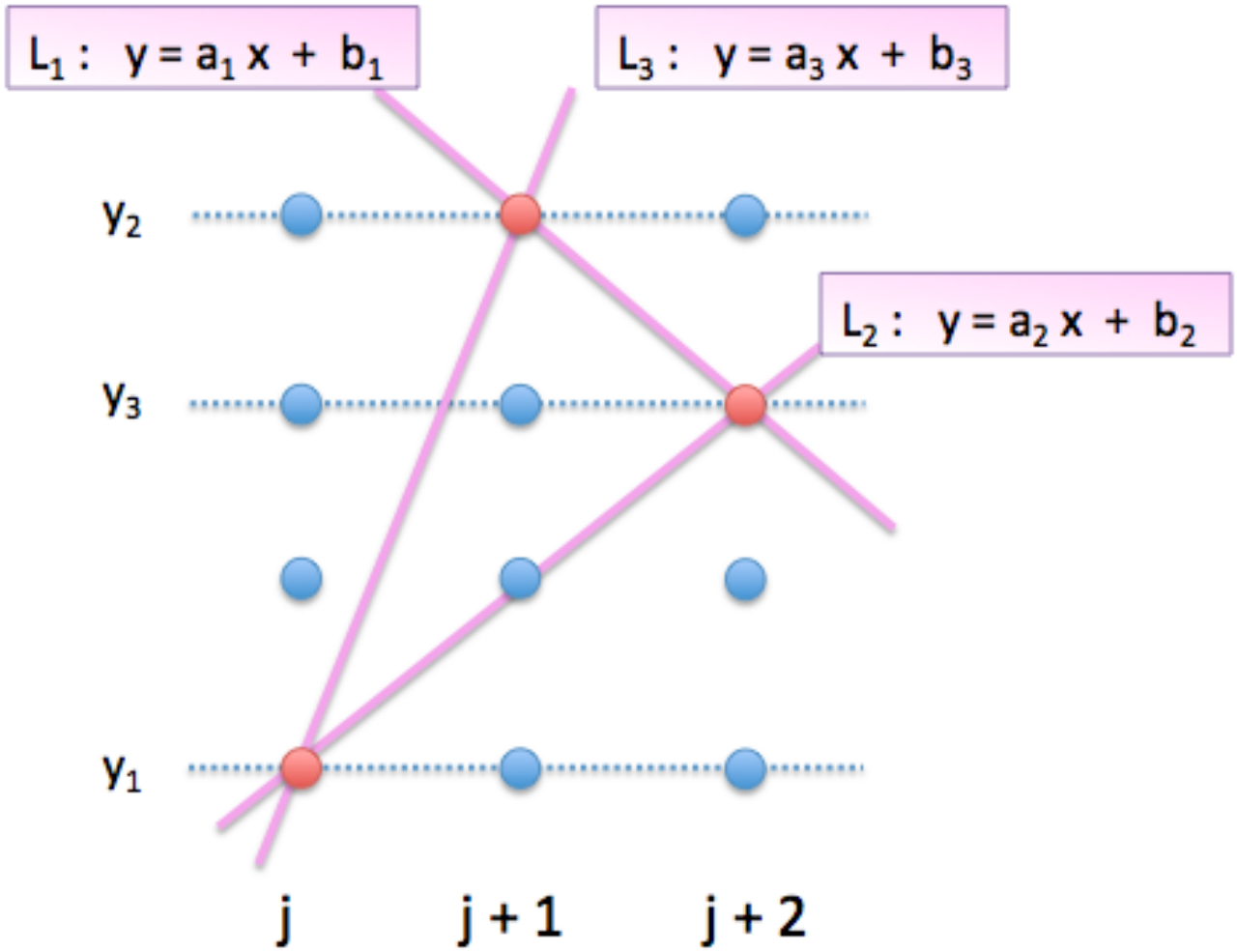
$y_2 - y_1 = a_3$ $y_3 - y_2 = a_1$ $y_3 - y_1 = 2a_2$

Why is key claim true?



Conclude: $a_1 + a_3 = 2a_2$

Why is key claim true?



This contradicts (A is 3-AP-free). So K_{bad} can't exist!

The upshot

Theorem (DvM'10)

Fix any $t(k) \leq \text{poly}(k)$. There is an NP-complete language L, and a poly-time reduction from $\text{OR}_=(L)^{t(k)}$ to N-Clique, whose output instance $\langle G, s \rangle$ satisfies

$$N = |V(G)| \leq O(k^2 + k \cdot t^{.5+o(1)}) .$$

- Now suppose N-Clique had a kernelization R' with output size bound $N^{2-\varepsilon}$.
- Let $t(k) := k^C$, for some $C \gg 1/\varepsilon$. Apply R to the output of DvM reduction. Maps $\text{OR}_=(L)^{t(k)}$ instance to an N-Clique instance of size

$$O\left(\left[k^2 + k \cdot k^{.5C+o(1)}\right]^{2-\varepsilon}\right) = o\left(k^{C-1}\right) .$$

The upshot

- But L is NP-complete, so [FS'08] tells us: can't compress $\mathrm{OR}_{=}(L)^{k^C}$ instances even to size $O(k^C \log k)$

  (of any target problem)...

- unless $\mathrm{NP} \subset \mathrm{coNP/poly}$.

- Similar proof: N-Clique on d-hypergraphs does not have $N^{d-\varepsilon}$-kernels. Same for N-Vertex Cover, others.

Simplification

- [Dell, Marx '12]: simpler proofs of these and related results.
- Basic idea: to efficiently compress $\mathrm{OR}_{=}(L)$ instances into N-Clique, choose L as an NP-complete language with "special structure"
  (making instances easier to combine in a shared graph)

The "fussy clique" problem

Define $L =$ FUSSY-CLIQUE:

- Input: A graph G on $2s^2$ vertices, presented as

$$V(G) = X_1 \cup \ldots \cup X_s \cup Y_1 \cup \ldots \cup Y_s \, .$$

  Require:

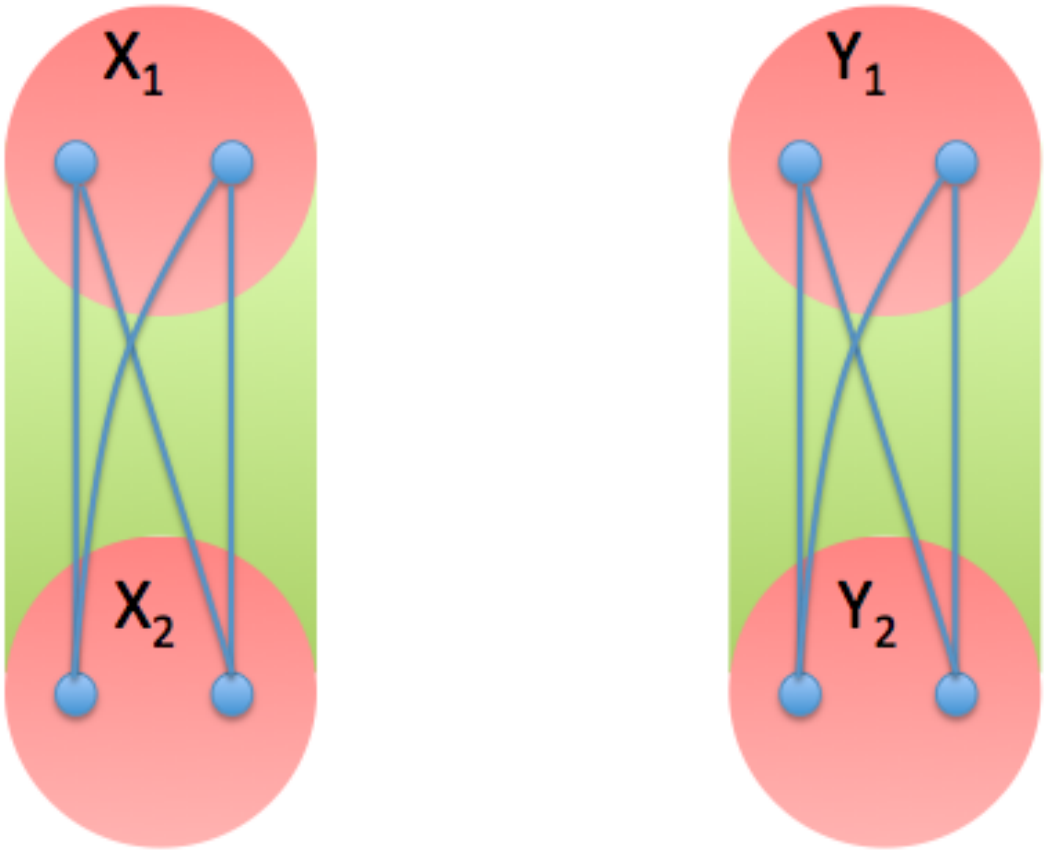
  1. each X_i (Y_i) is an independent set of size s;
  2. each pair (X_i, X_j) is a complete bipartite graph $(i \neq j)$. Same for (Y_i, Y_j).

- Decide: does G have a clique of size $2s$?

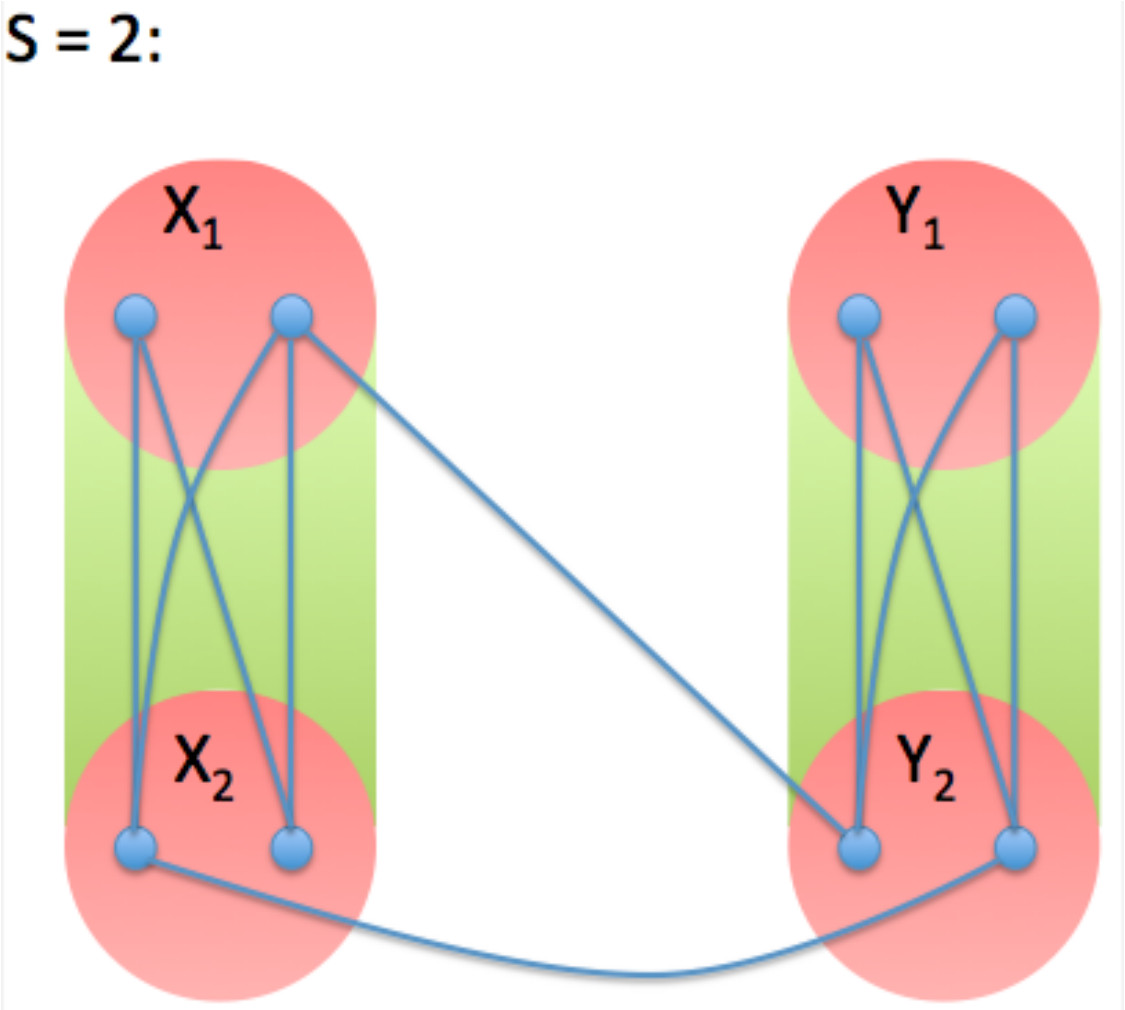
- NP-complete ([Dell, Marx '12], essentially)

Structure of "fussy" graphs

S = 2:



Structure of "fussy" graphs



The "fussy clique" problem

- Easy to compress $OR_{=}$(FUSSY-CLIQUE) instance into an N-Clique instance.

  Given: t FUSSY-CLIQUE instances

  $$\{G_{p,q}\}_{p,q \leq \sqrt{t}} \ .$$

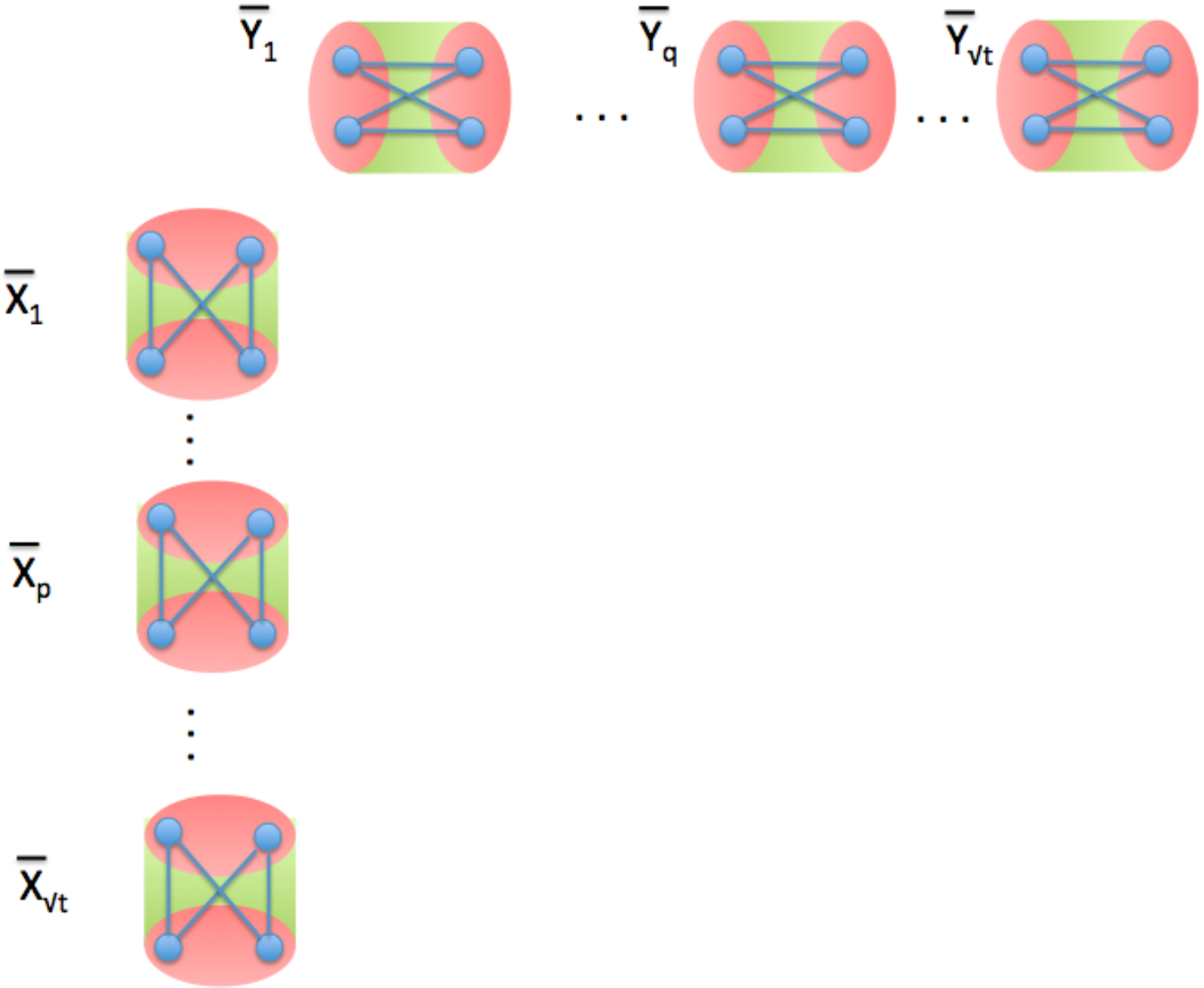
- Create graph G^* on $\sqrt{t} \times 2s^2$ vertices, in s-vertex parts

  $$X_{p,i} \ , \ Y_{p,i} \qquad i \leq s, \ p \leq \sqrt{t} \ .$$

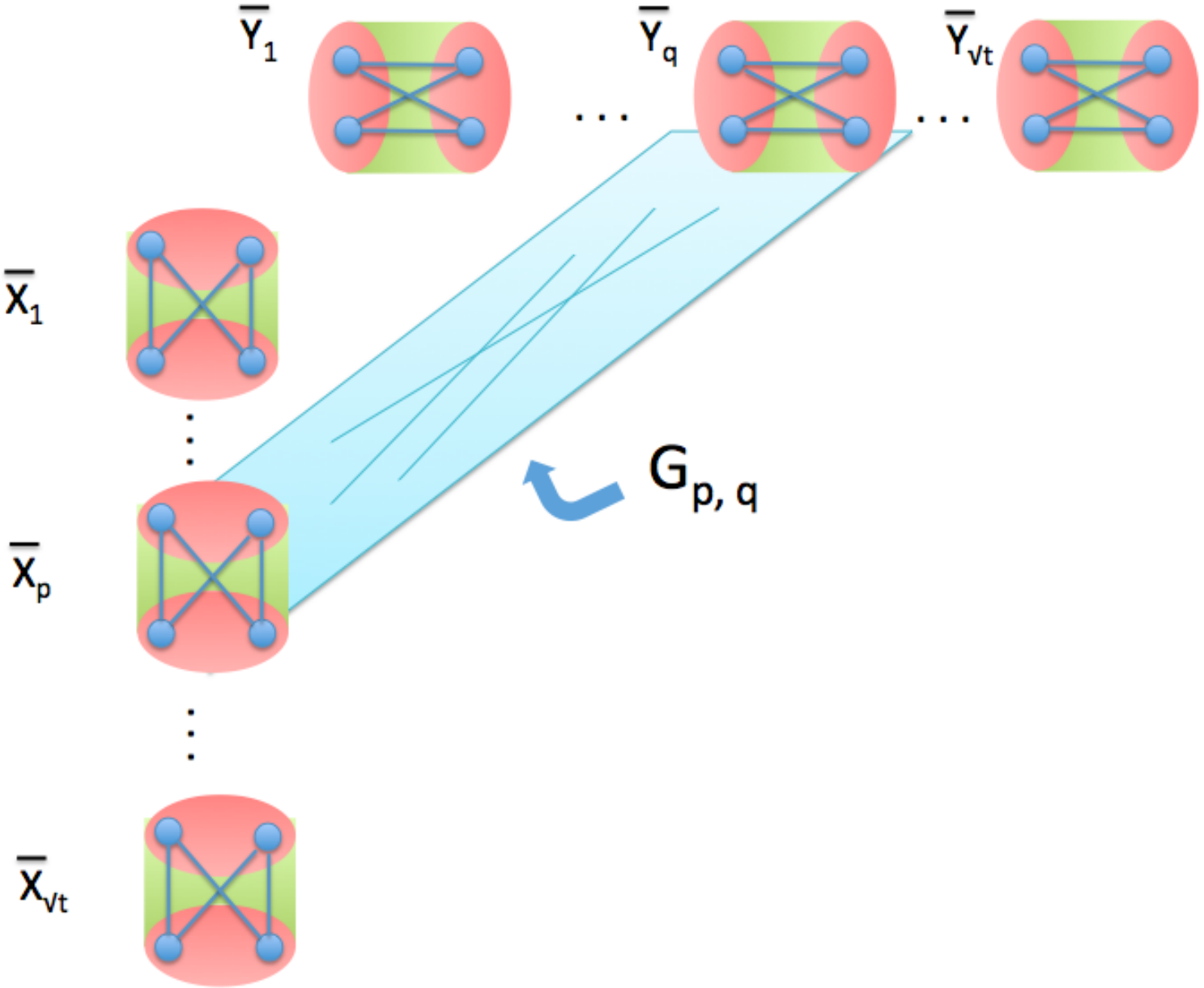
- For $p, q \leq \sqrt{t}$, place copy of $G_{p,q}$ on vertex-set

  $$\overline{X}_p \ := \ X_{p,1}, \ldots, X_{p,s} \quad \cup \quad \overline{Y}_q \ := \ Y_{q,1}, \ldots, Y_{q,s} \ .$$

The "fussy clique" problem



The "fussy clique" problem



- Analysis: First: if some $G_{p,q}$ has an $2s$-clique, so does G^*.
- Now suppose G^* has an $2s$-clique C. Can intersect only one $\overline{X}_p$, and one $\overline{Y}_q$.
- Every $G_{p,q'}$ using $\overline{X}_p$ adds the same edges within $\overline{X}_p$. Similarly for $\overline{Y}_q$.
- Thus, C must be a clique in $G_{p,q}$!
- Have reduced $OR_{=}(\text{FUSSY-CLIQUE})^t$ to an N-Clique instance $\langle G^*, 2s \rangle$ with $N = |V(G^*)| \leq O(s^2\sqrt{t})$.
- Here $|\langle G_{p,q} \rangle| \approx s^4$, so reduction is good enough to infer the same kernel-size lower bounds we got from [DvM'10].