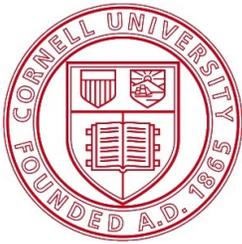
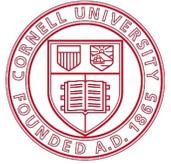


Counting Solution Clusters Using Belief Propagation



Lukas Kroc, Ashish Sabharwal, Bart Selman
Cornell University

Physics of Algorithms
Santa Fe, September 3, 2009



Constraint Satisfaction Problem (CSP)

□ Constraint Satisfaction Problem **P**:

Input: a set V of **variables**

a set of corresponding **domains** of variable values [discrete, finite]

a set of **constraints** on V [constraint \equiv set of allowed value tuples]

Output: a solution, valuation of variables that satisfies all constraints

Well Known CSPs:

□ **k-SAT**: Boolean satisfiability

- Domains: $\{0,1\}$ or $\{\text{true}, \text{false}\}$

- Constraints: disjunctions of variables or their negations (“clauses”) with exactly k variables each

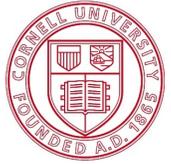
$$F = \underbrace{(x \vee y)}_{\alpha} \wedge \underbrace{(\neg x \vee z)}_{\beta}$$

□ **k-COL**: Graph coloring

- Variables: nodes of a given graph

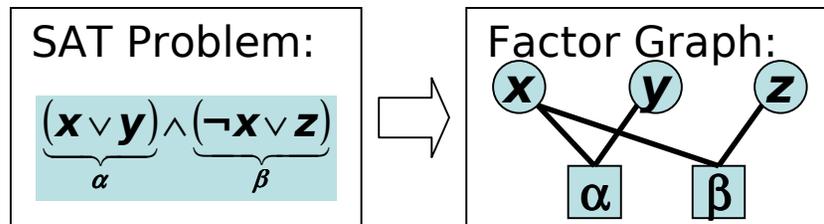
- Domains: colors $1 \dots k$

- Constraints: no two adjacent nodes get the same color.



Encoding CSPs

- One can visualize the connections between variables and constraints in so called **factor graph**:
 - A bipartite undirected graph with two types of nodes:
 - **Variables**: one node per variable
 - **Factors**: one node per constraint

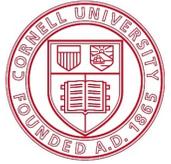


- Each factor node α has an associated **factor function** $f_{\alpha}(\mathbf{x}_{\alpha})$, weighting the variable setting. For CSP, $f_{\alpha}(\mathbf{x}_{\alpha})=1$ iff constraint is satisfied, else =0
 - Weight of the full configuration \mathbf{x} : $F(\mathbf{x}) = \prod_{\alpha} f_{\alpha}(\mathbf{x}_{\alpha})$
 - Summing weights of all configurations defines **partition function**:

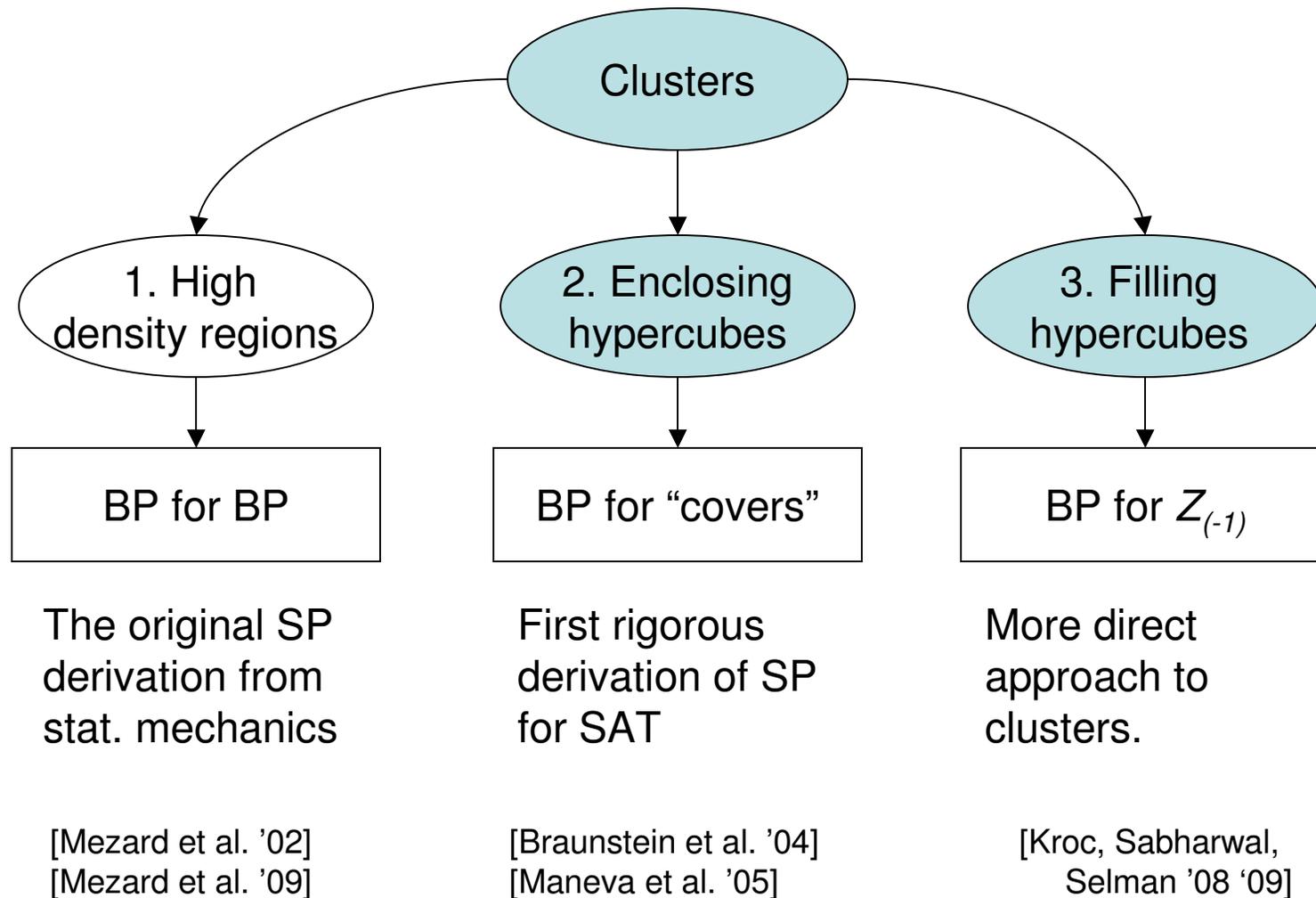
$$Z = \sum_{\mathbf{x}} \prod_{\alpha} f_{\alpha}(\mathbf{x}_{\alpha})$$

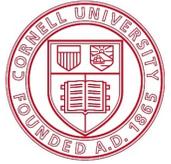
- For CSPs the partition function **computes the number of solutions**

Can we count “clusters” of solutions similarly?



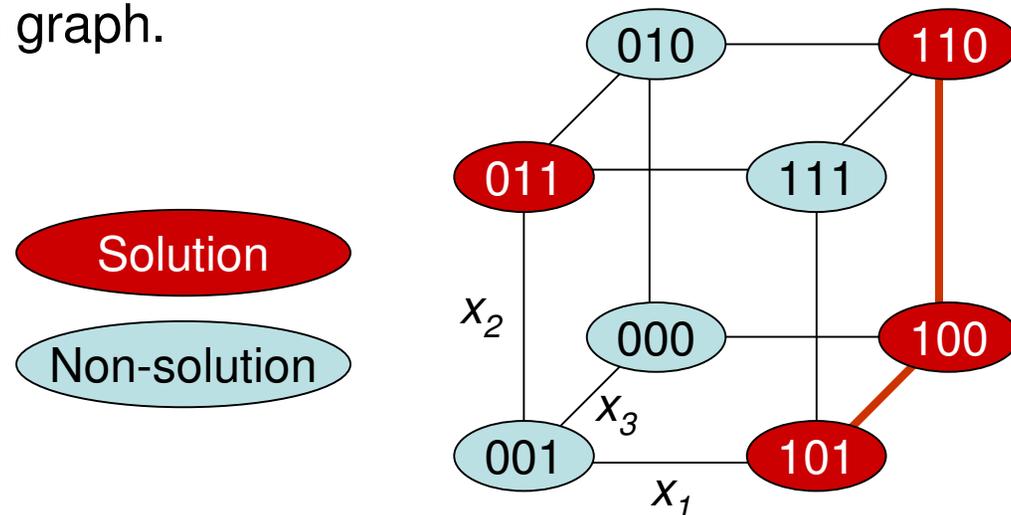
Talking about Clusters



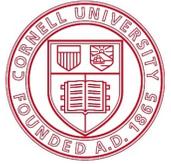


Clusters as Combinatorial Objects

- **Definition:** A **solution graph** is an undirected graph where nodes correspond to solutions and are **neighbors** if they differ in value of only one variable.
- **Definition:** A **solution cluster** is a connected component of a solution graph.



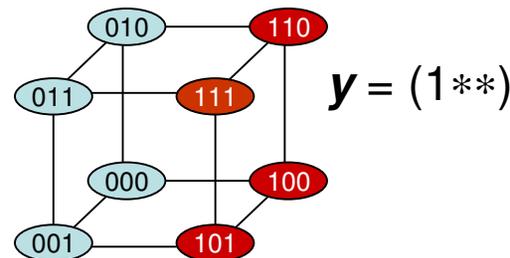
- **Note:** this is not the only possible definition of a cluster



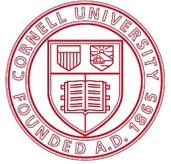
Thinking about Clusters

- Clusters are subsets of solutions, possibly exponential in size
 - not practical to work with
- To compactly **represent clusters**, we trade off expressive power for shorter representation
 - loose some details, but gain representability
- **Approximate by hypercubes “from outside” & “from inside”**
 - **Hypercube:** Cartesian product of non-empty subsets of variable domains

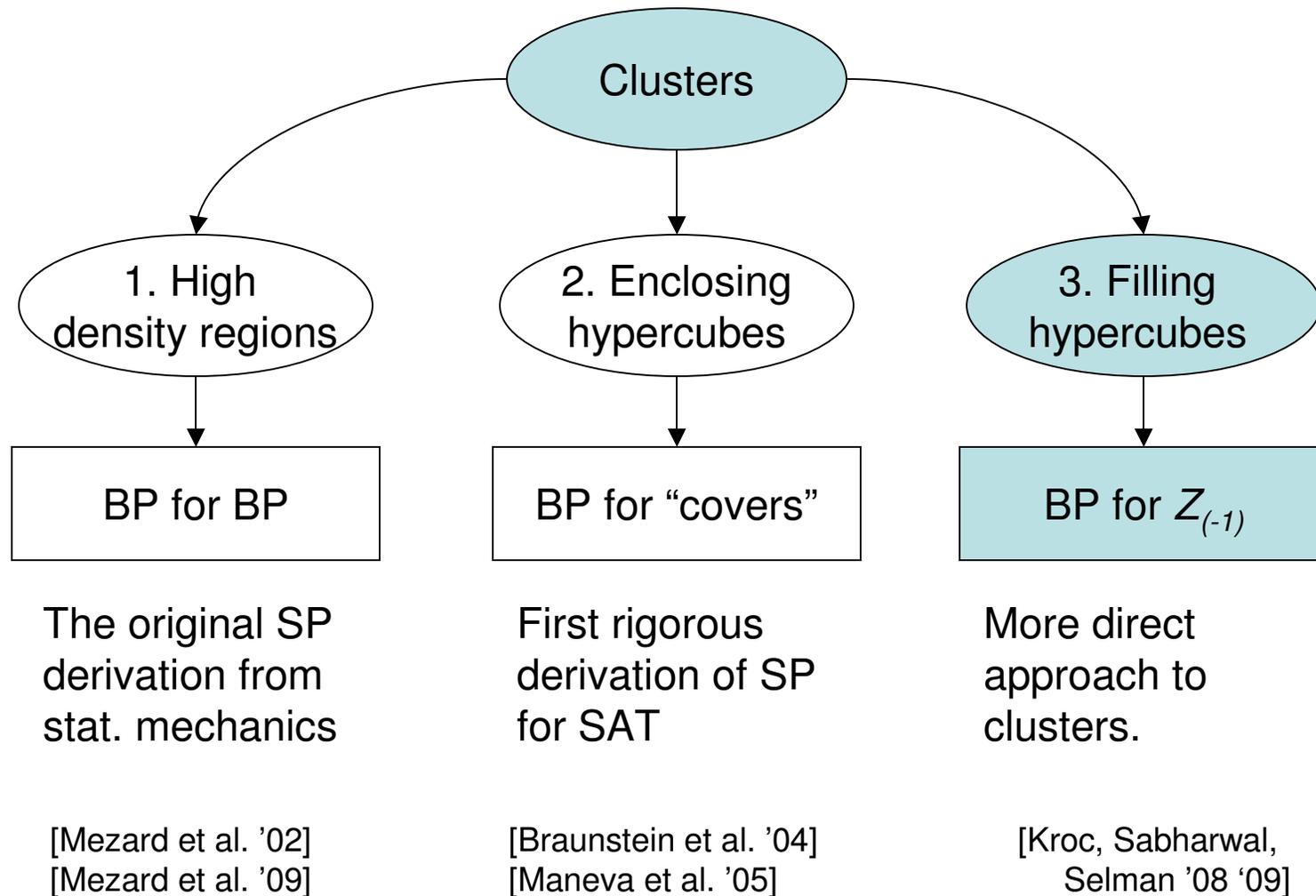
- E.g. with $* = \{0,1\}$,
 $\mathbf{y} = (1**)$ is a
2-dimensional hypercube
in 3-dim space

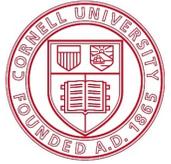


- **From outside:** The (unique) minimal hypercube enclosing the whole cluster.
- **From inside:** A (non-unique) maximal hypercube fitting inside the cluster.



Talking about Clusters





Factor Graph for Clusters

- To reason about clusters, we seek a factor graph representation
 - Because we can do approximate inference on factor graphs
 - Need to count clusters with an expression similar to Z for solutions:

$$Z = \sum_{\mathbf{x} \in \{0,1\}^n} \underbrace{\prod_{\alpha} f_{\alpha}(\mathbf{x}_{\alpha})}_{=F(\mathbf{x}) = 1 \text{ iff } \mathbf{x} \text{ is a solution}}$$

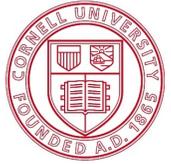
- Indeed, we derive the following for **approximating number of clusters**:

$$Z_{(-1)} = \sum_{\mathbf{y} \in \{0,1,*\}^n} (-1)^{\#\ast(\mathbf{y})} \prod_{\alpha} f'_{\alpha}(\mathbf{y}_{\alpha})$$

$$f'_{\alpha}(\mathbf{y}_{\alpha}) = \prod_{\mathbf{x}_{\alpha} \in \mathbf{y}_{\alpha}} f_{\alpha}(\mathbf{x}_{\alpha})$$

Checks whether all points in \mathbf{y}_{α} are good

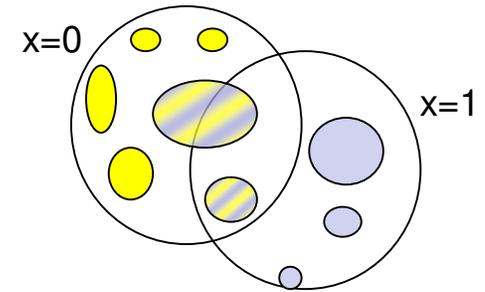
- Syntactically very similar to standard Z , which computes exactly number of solutions
- Exactly counts clusters under certain conditions, as discussed later
- Analogous expression can be derived for any discrete variable domain



Counting Solution Clusters

Divide-and-Conquer Recursively:

- Arbitrarily pick a variable, say x , of formula F
- Count how many clusters contain solutions with $x=0$
(ok if the cluster has solutions with both $x=0$ and $x=1$)
- Add number of clusters that contain solutions with $x=1$
- Subtract number of clusters that contain both solutions with $x=0$ and solutions with $x=1$

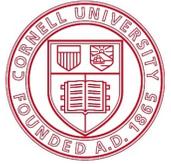


$$\#clusters = \#clusters(F)|_{x=0} + \#clusters(F)|_{x=1} - \#clusters(F)|_{x=0 \& x=1}$$

(Inclusion - exclusion formula)

Key issues:

- how can we compute $\#clusters(F)|_{x=0}$?
($\#clusters|_{x=1}$ would be similar)
- how do we compute $\#clusters(F)|_{x=0 \& x=1}$? (not a problem for SAT)

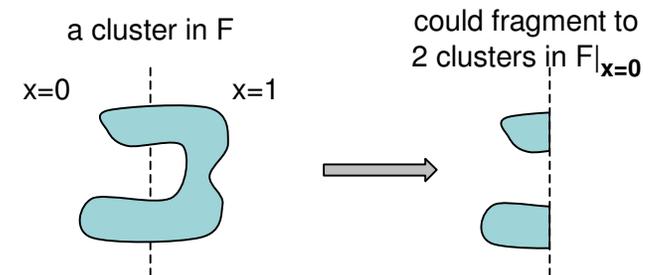


Computing $\#clusters(F)|_{x=0}$: **Fragmentation**

- Algorithmically, easiest way is to
 - “fix” x to 0 in the formula F , compute $\#clusters$ in new formula $(F|_{x=0})$

- So, use as approximation: $\#clusters(F)|_{x=0} \approx \#clusters(F|_{x=0})$

- Risk?
 - Potential *over-counting*: a cluster of F may **break/fragment** into several smaller, disconnected clusters when x is fixed to 0



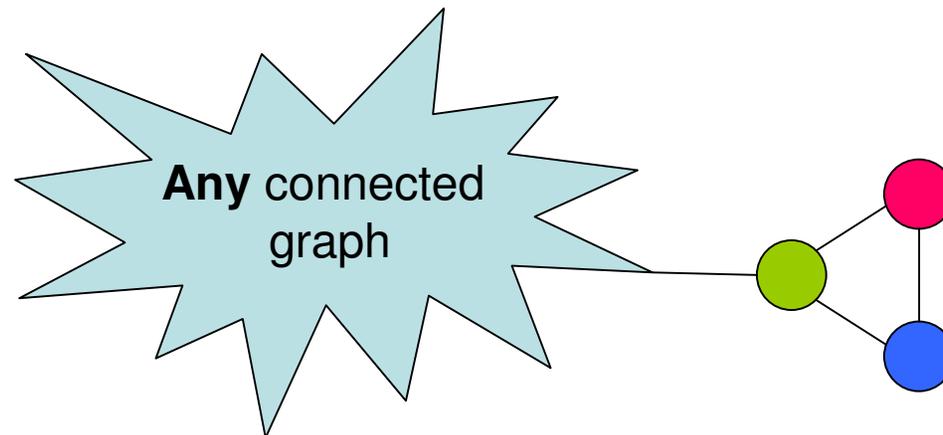
- Interestingly: Clusters often do not fragment!
- In particular, **provably** no fragmentation in **2-SAT** and **3-COL*** instances! (any instance, i.e., worst-case).
- Also, **empirically** holds for almost all clusters in random **3-SAT**, **logistics**, **circuits**, ...



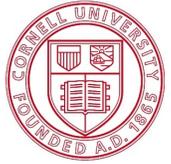
Theoretical Results: Exactness of $Z_{(-1)}$

On what kind of solution spaces does $Z_{(-1)}$ count clusters exactly?

- **Theorem:** $Z_{(-1)}$ is exact for any 2-SAT problem.
- **Theorem:** $Z_{(-1)}$ is exact for a 3-COL problem on G , if every connected component of G has at least one triangle.

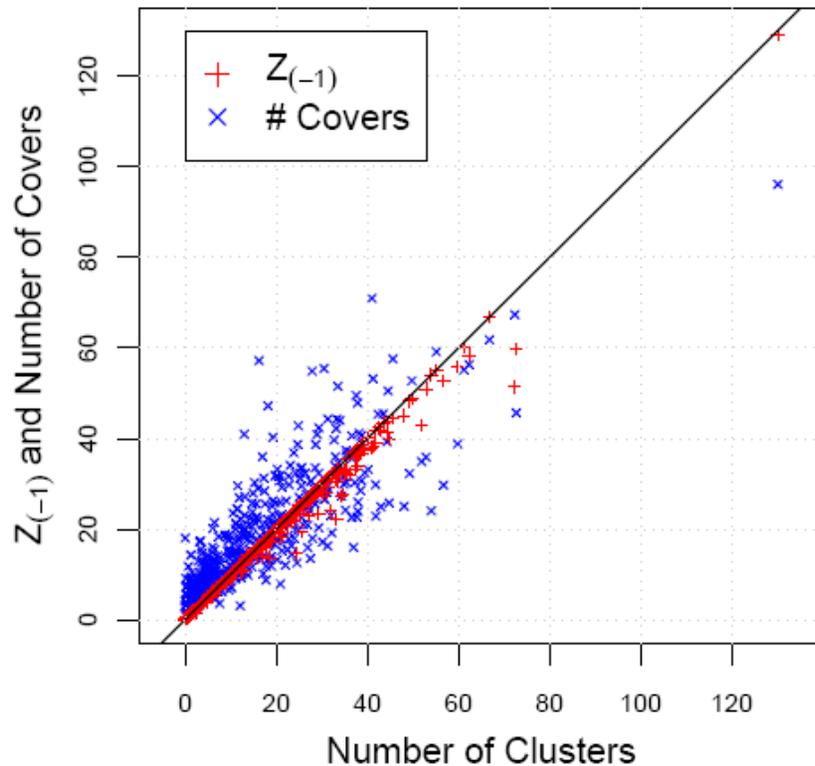


- **Theorem:** $Z_{(-1)}$ is exact if the solution space decomposes into “recursively-monotone subspaces”.

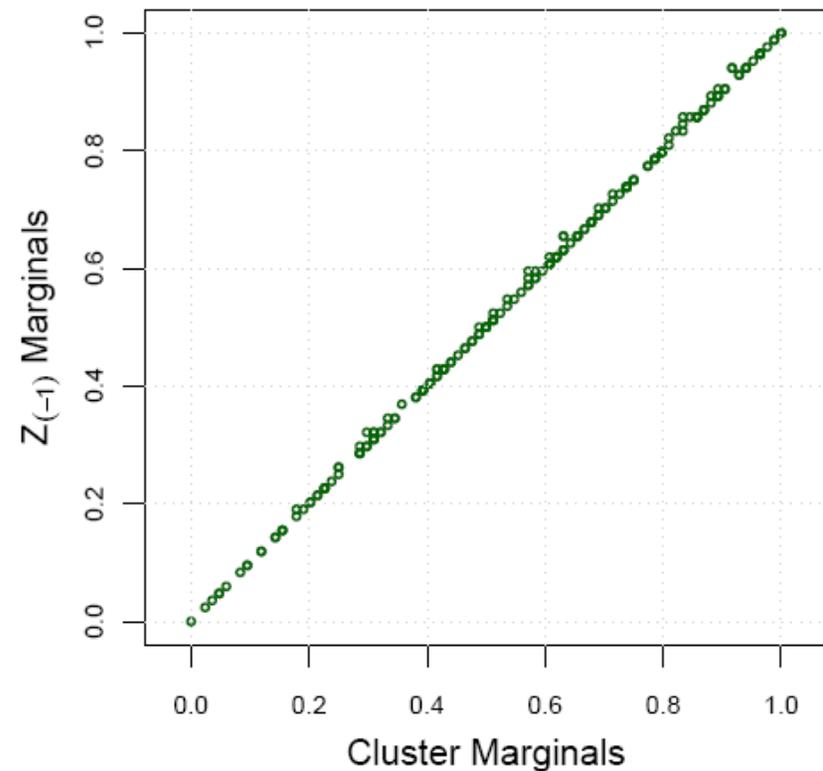


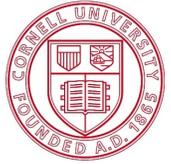
Empirical Results: $Z_{(-1)}$ for SAT

Random 3-SAT, $n=90$, $\alpha=4.0$
One point per instance



Random 3-SAT, $n=200$, $\alpha=4.0$
One point per variable
One instance





Empirical Results: $Z_{(-1)}$ for SAT

- $Z_{(-1)}$ is remarkably accurate even for many **structured formulas** (formulas encoding some real-world problem):

Instance Name	# solutions	# clusters	$Z_{(-1)}$
v32r250p1	52081218	6	6
v32r500p5	1543304664	6	6
driverlog1_ks99i	856152	338100	338100
rovers1_ks99i	17850294	15	15
rovers1_v01a	83200608	46	46
rovers1_v01i	266000	15	15
rovers2_ks99i	531360	8	8
rovers2_v01a	52107696	316	308
rovers2_v01i	21504	8	8
rovers4_ks99i	13794198600	11	11
rovers4_v01a	2592794880	22	22
rovers4_v01i	28447200	11	11



BP for Estimating $Z_{(-1)}$

- Recall that the number of clusters is very well approximated by

$$Z_{(-1)} = \sum_{\mathbf{y} \in \{0,1,*\}^n} (-1)^{\#\ast(\mathbf{y})} \prod_{\alpha} f'_{\alpha}(\mathbf{y}_{\alpha})$$

- This expression is in a form that is very similar to the standard partition function of the original problem, which we can approximate with BP.
- $Z_{(-1)}$ can also be approximated with “BP”: the factor graph remains the same, only the semantics is generalized:
 - Variables: $\mathbf{y} \in \{0, 1, *\}^n$
 - Factors: $f'_{\alpha}(\mathbf{y}_{\alpha}) = \prod_{\mathbf{x}_{\alpha} \in \mathbf{y}_{\alpha}} f_{\alpha}(\mathbf{x}_{\alpha})$
- And we need to adapt the BP equations to cope with (-1).



BP Adaptation for (-1)

- Standard BP equations can be derived as **stationary point conditions** for continuous constrained optimization problem [Yedidia et al. '05]
 - Let $p(\mathbf{x})$ be the uniform distribution over solutions of a problem
 - Let $b(\mathbf{x})$ be a unknown parameterized distribution from a certain family
 - The goal is to **minimize** $D_{\text{KL}}(\mathbf{b}||\mathbf{p})$ over parameters of $b(\cdot)$
 - Use $b(\cdot)$ to approximate answers about $p(\cdot)$

- The BP adaptation for $Z_{(-1)}$ follows exactly the same path, and generalizes where necessary.

One can derive a message passing algorithm for inference in factor graphs with (-1)

- We call this adaptation **BP**₍₋₁₎



The Resulting $BP_{(-1)}$

- The $BP_{(-1)}$ iterative equations:

The black part is BP

$$n_{i \rightarrow \alpha}(y_i) \propto \prod_{\beta \ni i \setminus \alpha} m_{\beta \rightarrow i}(y_i)$$

$$m_{\alpha \rightarrow i}(y_i) \propto \sum_{\mathbf{y}_{\alpha \setminus i} \in \{0,1,*\}^{|\alpha|-1}} f'_{\alpha}(\mathbf{y}_{\alpha}) \prod_{j \in \alpha \setminus i} (-1)^{\delta(y_j=*)} n_{j \rightarrow \alpha}(y_j)$$

Relation to SP:

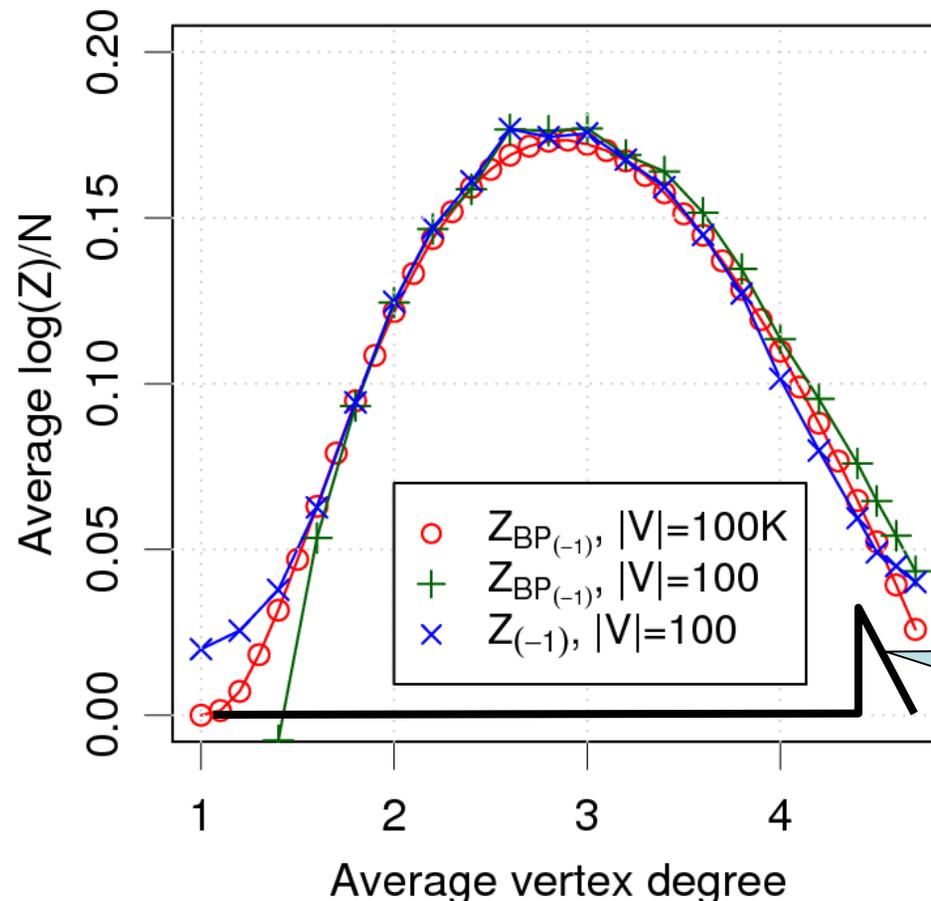
- **For SAT: $BP_{(-1)}$ is equivalent to SP**
 - The instantiation of the $BP_{(-1)}$ equations can be rewritten as SP equations
- **For COL: $BP_{(-1)}$ is different from SP**
 - $BP_{(-1)}$ estimates the total number of clusters
 - SP estimates the number of clusters with most frequent size



BP₍₋₁₎: Results for COL

Experiment: rescaling number of clusters and $Z_{(-1)}$

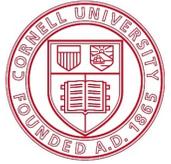
1. for **3-colorable** graphs with various average degrees (x-axis)
2. count $\log(Z_{(-1)})/N$ and $\log(Z_{BP(-1)})/N$ (y-axis)



The rescaling assumes that
 $\#clusters = \exp(N \Sigma(c))$

$\Sigma(c)$ is so called **complexity**
and is instrumental in various
physics-inspired approaches
to cluster counting (will see
later)

Sketch of SP results:
Nonzero between
4.42 and 4.69



Summary

- Truly combinatorial framework for cluster counting: $Z_{(-1)}$
 - Applicable to structured problems (contrast with original SP clusters)
 - With theoretical exactness results

- Algorithm for approximate inference over clusters: $BP_{(-1)}$
 - Direct derivation of SP for SAT
 - Allows derivation of new algorithms for other combinatorial problems