Approximate Inference for Clusters in Solution Spaces*
Extended Abstract

Lukas Kroc and **Ashish Sabharwal** and **Bart Selman**
Department of Computer Science
Cornell University, Ithaca NY 14853-7501, U.S.A.
{*kroc,sabhar,selman*}*@cs.cornell.edu*

This work proposes new approximate (and exact) inference methods for reasoning about an important and hard-to-compute property of the solution space of combinatorial problems, namely clusters of solutions. Given a constraint satisfaction problem (CSP), we can think of two solutions as being "connected" if they differ in the value of only one variable. Clusters of solutions can thus be defined in a natural manner: two solutions s_1 and s_2 are in the same cluster if and only if there is a path of connected solutions from s_1 to s_2. The main question we seek to address is, *given a CSP, how many solution clusters does it have?*

We propose an approximate method that first reformulates the CSP as a "factor graph" over an extended set of variable domains, approximates the number of clusters using an exponential size expression defined over this factor graph, and then estimates the value of this expression using message passing techniques, specifically an extension of the belief propagation algorithm.

Knowledge about the clustering structure of the solution space of a problem can be a very useful tool in understanding and reasoning about the problem domain. For instance, knowing that a solution lies in a large cluster of solutions suggests robustness: if an application demands a small tweak to the initial solution we provide, it is likely to be easy to accommodate that tweak and still have a valid solution if the original solution was part of a big cluster. Similarly, when an application requires a sample of, say, 10-15 solutions, it is often desirable to provide fairly different 10-15 solutions rather than all solutions that differ only slightly (e.g., in a handful of variable values out of thousands of variables). The number of solution clusters captures, in a sense, how many different *kinds* of solutions the problem has. When planning for contingencies, for example, it can be more useful to know that there are many, very different kinds of logistic plans or schedules, rather than knowing that there are several plans all of which could be insignificant variations of each other.

Yet another motivation for studying solution clusters, which was in fact our original motivation, comes from the recent literature on message passing algorithms, specifically belief propagation (BP) and survey propagation (SP). Message passing algorithms such as BP have been very successful in efficiently computing interesting properties of succinctly represented large, continuous probability spaces. BP has found a range of applications in, e.g., Bayesian inference, coding theory (turbo decoding), and image understanding. Message passing techniques have found a recent focus on computing properties of discrete spaces, in particular, properties of the *space of solutions* of combinatorial problems. For example, marginal probability information about the uniform distribution over solutions (or similar combinatorial objects) has been the key ingredient in the success of BP-like algorithms for propositional satisfiability (SAT) and graph coloring (COL) problems. Most notably, the SP algorithm utilizes this information to solve very large hard random instances of these problems (Mézard, Parisi, and Zecchina 2002; Braunstein et al. 2003). The performance of SP is unmatched by any other known algorithm for these challenging combinatorial problems. However, its origin in the statistical physics community and its roots in complex methodologies have made SP very hard to understand from a traditional computer science perspective.

The design of the SP algorithm has been inspired by the observation that the solution space of large random combinatorial problems often goes through significant and sudden changes in its *clustering structure* as the problems are made more and more constrained. For example, at clause-to-variable density of around 3.92, random 3-SAT instances suddenly go from having almost all solutions contained in one large cluster to having solutions distributed in an exponential number of small clusters. Similarly, another phase shift is believed to occur near density 4.15, when almost all solutions begin to cluster in only a constant number of "dominating" clusters. The BP algorithm is believed to be "confused" by the existence of many clusters, causing convergence difficulties. The SP algorithm was introduced as an attempt to circumvent this issue by explicitly accounting for the existence of clusters. The study of the solution space structure has been the focus of a number of recent papers (e.g., (Krzakala et al. 2007; Achlioptas and Ricci-Tersenghi 2006; Braunstein et al. 2003; Mézard, Parisi, and Zecchina 2002; Hartmann, Mann, and Radenback 2008;

*Preliminary results from this work, particularly for the graph coloring problem, appeared at the NIPS-08 conference, Vancouver, Canada, 2008; a significantly extended version is under preparation for a journal.

Ardelius, Aurell, and Krishnamurthy 2007)), especially by the statistical physics community, which has developed extensive theoretical tools to analyze such spaces under certain structural assumptions and large size limits. We provide a purely combinatorial method for counting the number of clusters and computing marginals, which is applicable even to small size problems and can be approximated very well by message passing techniques.

The problem of computing the number of clusters is hard, both theoretically and empirically. Doing it efficiently necessarily demands abstraction and approximation techniques. For constraint satisfaction problems (or a search problems) defined on a finite set of variables with discrete and finite domains, the problem of counting the number of solutions is #P-complete, while deciding whether there exists a solution is NP-complete. Many practical tools have been proposed in the past few years for the problem of solution counting, or model counting as it is sometimes called. These range from sampling based methods to exhaustive search to knowledge compilation to the use of special constraints to obtain probabilistically guaranteed lower or upper bounds. For cluster counting, however, not much is known yet.

The decision problem of deciding whether there exists a solution cluster is also "only" NP-complete (as there exists a solution cluster if and only if there exists a solution to begin with). Nonetheless, counting the number of clusters appears to be harder, at least in practice, than counting the number of solutions.[1] One intuitive explanation is the following. Suppose we are given a candidate solution s. Verifying whether s is in fact a valid solution is a polynomial time task, i.e., in the class P. On the other hand, a similar picture with solution clusters is much more complex. To begin with, it is not even clear how to succinctly represent a cluster. Suppose we are working with the binary domain and n variables, i.e., the search space is $\{0,1\}^n$. We can *approximate* a solution cluster with a string in $\{0,1,*\}^n$, with the semantic meaning that a 0 or 1 value indicates that the corresponding variable takes only this value in all solutions in the cluster, while a $*$ value indicates that it takes both 0 and 1 values in the cluster. Interestingly, we can show that even verifying that such a string in $\{0,1,*\}^n$ represents a legal cluster of a given CSP is both NP-hard and coNP-hard, and thus much harder than verifying the validity of a candidate solution (unless P = NP).

Given this computational difficulty, we seek to develop approximate inference methods for counting the number of clusters. Our approach can naturally be written in terms of a graphical model for any discrete constraint satisfaction problem (CSP). This yields an inclusion-exclusion based partition function style expression, $\mathbf{Z}_{(-1)}$, for counting the number of clusters, to which we apply the so-called variational method to obtain BP-like equations for estimating this quantity. This yields one of the first scalable methods for estimating the number of clusters of solutions of combinatorial problems using a BP-like algorithm. Our technique applies to discrete constraint satisfaction problems (CSPs) in general, and we evaluate it on random 3-SAT, random 3-COL, and a number of structured instances from the SAT Competition (Le Berre and Simon (Organizers) 2005). While the naïve method, based on enumeration of solutions and pairwise distances, scales to, for example, COL problems with 50 or so nodes and a recently proposed local search based method provides estimates up to a few hundred node graphs (Hartmann, Mann, and Radenback 2008), our approach—being based on BP—easily provides fast estimates for graphs with $100,000$ nodes. We validate the accuracy of our approach by also providing a non-trivial exact counting method for clusters, utilizing advanced knowledge compilation techniques for CSPs, namely binary decision diagrams (BDDs) (Bryant 1986) and the decomposable negation normal form (DNNF) (Darwiche 2001).

Our approach works with the factor graph representation of a CSP. An arbitrary CSP can be expressed in the form of a **factor graph**, a bipartite graph with two kinds of nodes. The *variable nodes*, denoted $\vec{x} = (x_1, \ldots, x_n)$, represent the variables in the CSP with their discrete domain *Dom*. The *factor nodes*, denoted $\alpha, \ldots$, with associated factor functions, denoted $f_\alpha, \ldots$, represent the constraints of the CSP. Each factor function is a Boolean-valued function with arguments $\vec{x}_\alpha$ (denoting a subset of the variables in $\vec{x}$) and range $\{0,1\}$, and evaluates to 1 if and only if the associated constraint is satisfied. For efficiency of valuating the factor functions, we assume they only have a constant number of arguments, independent of n. The factor graph has an edge between a variable node x_i and a factor node α if and only if the variable x_i appears in the constraint represented by f_α; we denote this event by $i \in \alpha$.

The factor representation weighs all variable assignments $\vec{x}$ according to the product of values of all factors. We denote this product by $F(\vec{x}) := \prod_\alpha f_\alpha(\vec{x}_\alpha)$. In our case, the weight of an assignment $\vec{x}$ evaluates to 1 if and only if all of the factors have value of 1, otherwise it evaluates to 0. The assignments with weight 1 correspond exactly to satisfying assignments, or solutions, of the CSP. The number of satisfying assignments can thus be expressed as the weighted sum across all possible value assignments to $\vec{x}$, where the weight is 1 if the value assignment corresponds to a solution of F and 0 otherwise. We denote this quantity by Z, the so called **partition function** at zero temperature:

$$Z := \sum_{\vec{x} \in Dom^n} F(\vec{x}) = \sum_{\vec{x} \in Dom^n} \prod_\alpha f_\alpha(\vec{x}_\alpha) \qquad (1)$$

Yedidia et al. (Yedidia, Freeman, and Weiss 2005) showed that there is a fairly mechanical variational method derivation that yields BP equations for estimating Z. Under certain assumptions, we derive a partition function style quantity, $Z_{(-1)}$, which has negations and relies on an inclusion-exclusion argument to count the number of clusters. While

[1] It is easy to see that counting the number of solution clusters is at least as hard as counting the number of solutions. One way to reduce the solution counting problem to cluster counting is as follows. Given a CSP F, construct a new CSP F' which has essentially two copies of F on disjoint sets of variables along with an equality constraint between each pair of corresponding variables. Then, solutions of F are in one-to-one correspondance with solutions to F'. Moreover, every two solutions of F' differ in the value of at least two variables, implying that every solution cluster in F' is of size one, which in turn implies that the number of clusters of F' equals the number of solutions of F', which equals the number of solutions of F.

the Z expression involves a summation where every variable takes all possible values in its domain, the $Z_{(-1)}$ expression generalizes this so that every variable takes all possible non-empty *subsets of values* from its domain. Without going into much detail, we include here our expression for $Z_{(-1)}$ for completeness, for the case of CSPs with binary domains $\{0,1\}$ for which this "extended" subset-based domain is nothing but $\{\{0\},\{1\},\{0,1\}\}$, which we will denote by $\{0,1,*\}$ for simplicity. We define the $Z_{(-1)}$ expression (for this domain) as follows:

$$Z_{(-1)} := \sum_{\vec{y}\in\{0,1,*\}^n} (-1)^{\#*(\vec{y})} \prod_{\alpha} f_{\alpha}^{ext}(\vec{y}_{\alpha}) \qquad (2)$$

where the extended function f^{ext} is defined to evaluate to 1 on $\vec{y} \in \{0,1,*\}$ if and only if f evaluates 1 on every instantiation $\vec{x}$ of $\vec{y}$ obtained by replacing each of the $*$'s with a 0 or a 1. Also, $\#*$ denotes the number of $*$ values in $\vec{y}$. When $\vec{y}$ has k $*$'s, then it represents a hypercube in $\{0,1\}^n$ with 2^k points.

We empirically demonstrate that $Z_{(-1)}$ is very accurate in estimating the number of clusters of problems from a variety of domains, both random and structured. See Figure 1 for an example of random instances, where an exact cluster count would correspond to all points falling on the diagonal. Each point in this plot represents a random 3-SAT instance. The x-coordinate is the true number of clusters while the y-coordinate is the value of $Z_{(-1)}$ on the formula.

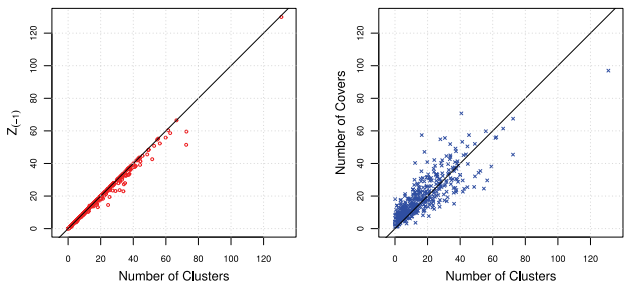

Figure 1: Number of clusters vs. $Z_{(-1)}$ (left) or number of covers (right, previous approach) in random 3-SAT for 90 variables and 360 clauses..

For structured formulas from the SAT Competitions, we did a similar study on small instances whose true cluster count could be computing using our BDD and DNNF based method. In this case as well, the value of $Z_{(-1)}$ often precisely matched the true number of clusters. For example, the driverlog1_ks99i instance, which has 856,152 solutions, has precisely 338,100 clusters and $Z_{(-1)}$ computes this value exactly. Similarly, the rovers1_v01a instance has 83,200,608 solutions and 46 clusters, and $Z_{(-1)}$ again computes this exactly. Another non-trivial problem from circuit synthesis, circuit-4-2-8-16-renum has 4,193,280 and 80,640 clusters, which $Z_{(-1)}$ again computes exactly.

We also provide some formal results about $Z_{(-1)}$, including the following:

Theorem 1. *$Z_{(-1)}$ is exact on the 2-SAT problem.*

Theorem 2. *$Z_{(-1)}$ is exact on the 3-COL problem restricted to graphs where every solution cluster has at least one vertex that takes at most two colors in that cluster.*

The required graph property in Theorem 2 holds, for example, when every connected component of the graph has a triangle, and therefore our result applies to all such graphs.

Finally, we use the variational method alluded to earlier (with negations, in this case) to obtain BP equations for estimating $Z_{(-1)}$, denoted $\text{BP}_{(-1)}$. It turns out that for 3-SAT, $\text{BP}_{(-1)}$ is identical to the SP equations. For 3-COL, however, $\text{BP}_{(-1)}$ works with all subsets of the 3-valued color domain, resulting in a richer set of equations than the SP equations previously proposed for 3-COL (Braunstein et al. 2003). We empirically demonstrate that $\text{BP}_{(-1)}$ is much better than SP in computing the number of clusters over a wide range of graph density parameters. We also show that marginal values obtained using $Z_{(-1)}$ and $\text{BP}_{(-1)}$ are much closer to the true cluster marginals than the marginals obtained by SP or the previously introduced notion of *covers* (Braunstein and Zecchina 2004; Maneva, Mossel, and Wainwright 2007; Kroc, Sabharwal, and Selman 2007). Overall, our experiments show that $Z_{(-1)}$ itself is an extremely accurate estimate of the number of clusters, and so is its approximation, $Z_{\text{BP}_{(-1)}}$, obtained from our $\text{BP}_{(-1)}$ equations.

Main Intuition Behind the Approach

For clarity, we consider binary domains here. Let F be a Boolean formula, possibly but not necessarily in the conjunctive normal form (CNF). Let the variables of F be $x_1, x_2, \ldots, x_n$. The idea is to use the *inclusion-exclusion principle* for counting: split clusters into two categories, count clusters that belong to the first category, add to it the number of clusters the belong to the second category, and then compensate for the potential overcount by subtracting away clusters that belong to both categories.

To illustrate the approach, let us first consider how one could count the number of solutions in a recursive manner, but repeatedly splitting the space into half. This will lead us to the traditional *partition function* expression (at zero temperature), Z, for F. We could compute $\#solutions(F)$ by selecting any variable x_1, recursively counting the solutions of F with $x_1 = v_1$ for $v_1 \in \{0,1\}$, and adding the two sub-counts (see left pane of Fig.). Each of these sub-counts, denoted $\#solutions(F)|_{x_1=v_1}$ is nothing but $\#solutions(F|_{x_1=v_1})$, i.e., we can determine the sub-count by first simplifying F by setting x_1 to v_1 and then computing the count of the simplified formula. We can in turn estimate $\#solutions(F|_{x_1=v_1})$ by selecting another variable x_2 and recursively repeating the same decomposition procedure to obtain 4 terms of the form $\#solutions(F|_{x_1=v_1,x_2=v_2})$. Repeating this n times, we get precisely the summation of 2^n terms, as in the Z expression.

Let's apply a similar technique to counting solution clusters. To compute $\#clusters(F)$, we could first count all clusters in which x_1 takes the value v_1, for $v_1 \in \{0,1\}$, and then compensate for the potential overcount by subtracting

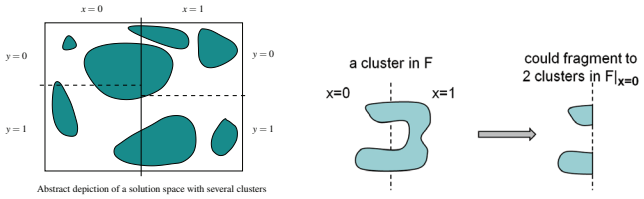

Figure 2: Left: recursive counting by partitioning the solution space. Right: fragmentation of solution clusters.

away the number of clusters in which x_1 takes both values, 0 and 1. Thus, $\#clusters(F) = \#clusters(F)|_{x_1=0} + \#clusters(F)|_{x_1=1} - \#clusters(F)|_{x_!=0 \& x_1=1}$. The tricky part is to compute $\#clusters(F)|_{x_1=v_1}$ for any v_1, as this no longer necessarily equals the number of clusters in the simplified formula $F|_{x_1=v_1}$. The reason is possible *cluster fragmentation*—a single cluster of F could fragment into two or more clusters if we restrict the value of x_1 to v_1, as depicted in the right pane of Fig. . Thus, one way to prove that $Z_{(-1)}$ is exact on a given domain (e.g., 2-SAT) is to show that there is no cluster fragmentation in that domain, and this indeed forms the main combinatorial argument of our proof of exactness. The second problem that may arise but only in domains with three or more values (e.g., 3-COL) is what we call *simultaneous valuation*—even if a variable takes three different values, say $\{R,G,B\}$, within a cluster, there may not be any valuation of the remaining $n-1$ variables for which it can take simultaneously take each subset of values, say, $\{R,G\}$ in the cluster. Thus, when computing the number of clusters in the simplified formula $F|_{x_1 \in \{R,G\}}$, which is one term we need for applying inclusion-exclusion, we will incorrectly get a zero count from this cluster even when x_1 takes both values R and G in this cluster. Therefore, one way to prove exactness of $Z_{(-1)}$ on domains like 3-COL involves showing that we can circumvent the simultaneous valuation issue in that domain by, e.g., requiring that the input graph has a triangle in each of its connected components.

Acknowledgments

This work was supported by the Intelligent Information Systems Institute, Cornell University (Air Force Office of Scientific Research AFOSR, grant FA9550-04-1-0151), the Defence Advanced Research Projects Agency (DARPA, REAL Program, grant FA8750-04-2-0216), and the National Science Foundation (NSF IIS award, grant 0514429; NSF Expeditions in Computing award for Computational Sustainability, grant 0832782). The authors thank Yahoo! for generously providing access to their M45 compute cloud.

The first author is currently affiliated with Los Alamos National Lab, Los Alamos, NM, USA.

References

Achlioptas, D., and Ricci-Tersenghi, F. 2006. On the solution-space geometry of random constraint satisfaction problems. In *38th STOC*, 130–139.

Ardelius, J.; Aurell, E.; and Krishnamurthy, S. 2007. Clustering of solutions in hard satisfiability problems. *J. Statistical Mechanics* P10012.

Braunstein, A., and Zecchina, R. 2004. Survey propagation as local equilibrium equations. *J. Statistical Mechanics* P06007.

Braunstein, A.; Mulet, R.; Pagnani, A.; Weigt, M.; and Zecchina, R. 2003. Polynomial iterative algorithms for coloring and analyzing random graphs. *Physical Review E* 68:036702.

Bryant, R. E. 1986. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers* 35(8):677–691.

Darwiche, A. 2001. Decomposable negation normal form. *J. ACM* 48(4):608–647.

Hartmann, A.; Mann, A.; and Radenback, W. 2008. Clusters and solution landscapes for vertex-cover and SAT problems. In *Workshop on Physics of Distributed Systems*.

Kroc, L.; Sabharwal, A.; and Selman, B. 2007. Survey propagation revisited. In *23rd UAI*, 217–226.

Kroc, L.; Sabharwal, A.; and Selman, B. 2008. Counting solution clusters in graph coloring problems using belief propagation. In *22nd NIPS*, 873–880.

Krzakala, F.; Montanari, A.; Ricci-Tersenghi, F.; Semerjian, G.; and Zdeborova, L. 2007. Gibbs states and the set of solutions of random constraint satisfaction problems. *PNAS* 104(25):10318–10323.

Le Berre, D., and Simon (Organizers), L. 2005. SAT 2005 competition.

Maneva, E.; Mossel, E.; and Wainwright, M. J. 2007. A new look at survey propagation and its generalizations. *J. ACM* 54(4):17.

Mézard, M.; Parisi, G.; and Zecchina, R. 2002. Analytic and algorithmic solution of random satisfiability problems. *Science* 297(5582):812–815.

Yedidia, J. S.; Freeman, W. T.; and Weiss, Y. 2005. Constructing free-energy approximations and generalized belief propagat ion algorithms. *IEEE Transactions on Information Theory* 51(7):2282–2312.