Decimation flows in constraint satisfaction problems

J. Stat. Mech. (2009) P12009

(http://iopscience.iop.org/1742-5468/2009/12/P12009)

Journal of Statistical Mechanics: Theory and Experiment
An IOP and SISSA journal

Decimation flows in constraint satisfaction problems

Saburo Higuchi[1,2] **and Marc Mézard**[1]

[1] Laboratoire de Physique Théorique et Modèles Statistiques, CNRS and Université Paris-Sud, Bâtiment 100, 91405 Orsay Cedex, France
[2] Department of Applied Mathematics and Informatics, Ryukoku University, Otsu, Shiga 520-2194, Japan
E-mail: hig@math.ryukoku.ac.jp and mezard@lptms.u-psud.fr

Received 13 August 2009
Accepted 10 November 2009
Published 21 December 2009

Online at stacks.iop.org/JSTAT/2009/P12009
doi:10.1088/1742-5468/2009/12/P12009

Abstract. We study hard constraint satisfaction problems with a decimation approach based on message passing algorithms. Decimation induces a renormalization flow in the space of problems, and we exploit the fact that this flow transforms some of the constraints into linear constraints over GF(2). In particular, when the flow hits the subspace of linear problems, one can stop the decimation and use Gaussian elimination. We introduce a new decimation algorithm which uses this linear structure and shows a strongly improved performance with respect to the usual decimation methods for some of the hardest locked occupation problems.

Keywords: disordered systems (theory), heuristics, message-passing algorithms

ArXiv ePrint: 0908.1599

J. Stat. Mech. (2009) P12009

Contents

1. Introduction

In recent years, statistical physics ideas and methods have become very useful in a number of problems involving many interacting variables in different branches of science. A particularly fruitful area which is attracting a lot of attention is that of constraint satisfaction problems. These are problems which appear in various areas of computer science and discrete mathematics. They include very fundamental problems like the satisfiability of random Boolean formulae [1, 2] or the coloring of graphs, and can also range all the way to very practical issues like chip testing or scheduling. Using statistical-physics-based approaches, the phase diagram of random ensembles of problems has been computed [3, 4], and new algorithms for finding solutions have been found [5]. There is now a whole field at the intersection of information theory, discrete mathematics and statistical physics [6]–[8].

The algorithms which have been so successful in satisfiability [4] and coloring [9] typically involve two main ingredients. The first one is a message passing method. It starts from a probability distribution on the set of interacting variables, which is typically the uniform measure on the configurations which satisfy all constraints. Then it aims at estimating, through mean-field-based methods, the marginal distribution of each of the variables with respect to this distribution. The second ingredient allows one to make use of the information on marginals provided by the message passing. In the decimation process [4, 10], one identifies, thanks to the messages, a strongly polarized variable and fixes it. The problem is thus reduced and the whole process (message passing and variable fixing) is iterated. The iteration of this process, when successful, eventually fixes all variables and finds a 'satisfying' assignment, i.e. an assignment that satisfies all

doi:10.1088/1742-5468/2009/12/P12009

J. Stat. Mech. (2009) P12009

constraints. Instead of decimation, one can also use a reinforcement procedure [11], but in the present paper we shall keep to decimation methods.

If one could use an algorithm giving the exact values of the marginals, any decimation procedure would be successful: if the marginal probability for the i th variable to take some value v is not zero, one can fix this variable to v and iterate the decimation; it will give one satisfying assignment. Actually fixing the variable at each step with a probability given by its marginal allows one to sample uniformly the space of satisfying assignments. However getting the exact marginals is very hard. The message passing methods give only some approximate estimate of these marginals, and decimation must try to make the best use of these noisy estimates in order to find a satisfying assignment. This paper explores new kinds of decimation procedures, which turn out to be more powerful than the usual one for some classes of hard problems. It is a kind of experimental paper, in that it does not provide any theory of decimation, but searches for efficient new procedures. The recent work of [12, 13] provides a first step of a theoretical approach to the decimation problem, by exploring some features of the ideal decimation procedure, based on the exact marginals.

The decimation can be seen as a flow in the space of problems, where each step of the flow transforms an N-variable problem into a new $(N-1)$-variable problem. The idea of decimation is thus a kind of renormalization procedure where one fixes a single variable at a time. Renormalization group ideas have already been applied to optimization problems, both in low dimensional systems [14] and in random-graph-based problems [15]. In the case of decimation, the choice of a heuristic for deciding what variable to fix at each step defines a flow. The simplest choice is to use the belief propagation (BP) message passing procedure and fix the most polarized variable. This leads to a deterministic flow. An alternative choice, which results in a stochastic flow, consists in choosing at random one variable among the R most polarized variables and fixing it. Belief propagation equations are nothing but the self-consistency equations of the cavity method within a replica symmetric approach; in some particularly difficult problems, when the multiplicity of metastable states is important, it is useful to go to a more sophisticated replica-symmetry-broken cavity approach. This leads to the use of survey propagation (SP) as the message passing, instead of BP. SP-based decimation and reinforcement are currently the best solvers for random satisfiability and coloring close to their SAT–UNSAT transition. Another useful improvement that we have found recently, which is efficient in some categories of hard problems, is performing a special decimation where, instead of fixing one variable, one fixes the relative value of two variables [16]. This requires us to extend the message passing approach in order to obtain correlations between the variables, a technique which has been developed recently by several groups [16]–[21]. The net result is also a decimation flow.

In this paper we propose a new use of decimation flows. Instead of trying to decimate the problem all the way until one has fixed all the variables and found a solution, we shall try to set up a decimation process that brings the problem towards a subspace of problems that are not hard to solve. Our approach is based on the remark that there exists a natural class of constraint satisfaction problems for binary variables which can be solved in polynomial time. These are the problems which can be written as systems of linear equations in GF(2). They can be solved by Gaussian elimination, which, in the worst case, takes a time growing like N^3. We shall call these problems 'linear problems' (the reader

J. Stat. Mech. (2009) P12009

should not be confused by the term 'linear': a linear problem does not necessarily have a linear energy function). We shall thus seek a decimation flow which arrives in the subspace of linear problems, and then use Gaussian elimination. This strategy turns out to be rather powerful for a class of constraint satisfaction problems which are particularly hard to solve with usual methods, and in particular with the standard message passing plus decimation strategy. These problems form a subclass of the locked occupation problems (LOPs), introduced recently in [22, 23]. Most of these LOPs are NP-hard [24]. The exceptions are precisely the linear systems, and some of the cases where the variables have degree 2, such as perfect matching. It turns out that, for a whole class of the hard LOPs, the reason why they are hard to solve within message passing/decimation is that they tend to flow towards linear problems, where variables (and even variable pairs) are typically unpolarized. This makes the next decimation steps rather difficult. If one instead interrupts the decimation when the subspace of linear problems is reached by the decimation flow, the resulting algorithm shows much better performance.

The paper is organized as follows. Section 2 introduces the locked occupation problems. Section 3 explains the decimation flows that we study; it describes the space of problems (weighted occupation problems) which are stable under this flow, and it defines the special kinds of constraints that can be encountered in decimation, in particular the linear constraints. In section 4 we explain how to identify the optimal variables (or pairs of variables) on which one performs the decimation steps. Section 5 summarizes the algorithm and shows the results of some numerical experiments. A short conclusion is given in section 6.

2. Locked occupation problems

Locked occupation problems contain $|V| = N$ binary variables taking values $x_i \in \{0, 1\}$ $(i \in V)$. These variables are related by $|F| = M$ constraints, denoted by $a \in F$, each one involving k variables. In order to characterize these constraints, it is convenient to define a variable x_i as 'occupied' when $x_i = 1$, and 'empty' if $x_i = 0$. Each constraint a relates k variables, with indices $i(a, 1), \ldots, i(a, k)$. It imposes some restrictions on the total number of occupied variables among these k variables. These restrictions are fully specified by a $(k+1)$-component 'constraint vector' A with binary entries, $A(r) \in \{0, 1\}$. The constraint a is satisfied if and only if

$$A\left(\sum_{m=1}^{k} x_{i(a,m)}\right) = 1. \tag{1}$$

A factor graph $G = (V, F; E)$ is associated with every instance of a LOP in the usual way [25]. The set of vertices of this bipartite graph G is $V \cup F$ while the set of edges is $E = \{(i, a)|i \in V, a \in F, \exists j \text{ s.t. } i = i(a, j)\}$. The notion of neighborhood is naturally introduced: $\partial a = \{i \in F|(i, a) \in E\}$, $\partial i = \{a \in V|(i, a) \in E\}$. For a collection of variables in $S \subset V$, we shall write $\underline{x}_S = \{x_i|i \in S\}$. We also use the shorthand notation $\underline{x} = \underline{x}_V$. The uniform measure over satisfying configurations can thus be written as

$$P(\underline{x}) = \frac{1}{Z}\prod_{a=1}^{M} \psi_a(\underline{x}_{\partial a}) \tag{2}$$

where $\psi_a(\underline{x}_{\partial a}) = A(\sum_{i\in\partial a} x_i)$. It exists as soon as there is at least one configuration satisfying all constraints.

An occupation problem is *locked* if the following three conditions are met [22, 23, 26]:

- $A(0) = A(k) = 0$.
- $A(r)A(r+1) = 0$ for $r = 0, \ldots, k-1$.
- Each variable appears in at least two constraints.

For example, the problem defined by $k = 3, A = (A(0), A(1), A(2), A(3)) = (0, 1, 0, 0)$ corresponds to positive 1-in-3 satisfiability [27], while the problem with $k = 4, A = (0, 1, 0, 1, 0)$ is a system of odd-parity checks.

3. Decimation flow

In this paper we concentrate on finding a satisfying assignment with large probability, assuming that there is at least one such assignment. To this end, we make use of 'flow' operations which replace an instance I of the constraint satisfaction problem at hand with another instance I' which has fewer variables. We will focus on two basic flow steps, which consist in either fixing a variable to a given value, or fixing the relation between two variables. Our procedure thus generalizes the usual decimation by introducing also the possibility of 'pair fixing'. This strategy follows from the observation in [16] that the correlations between variables play an important role in LOPs. The first case (single-variable decimation) consists in imposing

$$x_i = x \tag{3}$$

where x is 0 or 1. The second one (pair decimation) consists in imposing

$$x_i = x_j + y \bmod 2 \tag{4}$$

where y is 0 or 1. In the next section we will explain with what kinds of approximate methods one can identify a variable x_i and its assigned value x for single-variable decimation, or a pair x_i, x_j and its assigned relative value y for pair decimation. Here we want to study the flow process itself.

First we notice that each step effectively reduces the number of variables by 1. At each decimation we update a table which contains all the variables which have been fixed, and to what value (either a fixed value or a relation to another variable). When the decimation is completed, this table allows us to find the values of all variables. As usual in renormalization group flows, our basic flow steps, when applied to a LOP, produce a problem (instance) which is no longer a LOP. One thus needs to identify a space of problems, containing the LOPs, which is stable under our two basic decimation steps. Such a stable space is provided by the *weighted* occupation problems.

3.1. Weighted occupation problems

We generalize the occupation problems as follows. We first allow the constraint degree k and constraint vector A to be dependent on the constraint: the constraint a involves k_a variables and its constraint vector can depend on a and is denoted as A_a. Each constraint a depends furthermore on k_a integer weights $w_{a,i}$, $i \in \partial a$, and on a shift s_a. In a weighted

occupation problem, the constraint a is satisfied if and only if the variables $x_i, i \in \partial a$ are such that

$$A_a \left(\sum_{i \in \partial a} w_{a,i} x_i + s_a \right) = 1. \tag{5}$$

Let W be the set of weighted occupation problems. We shall now show that W is stable under our two basic decimation steps, and make explicit the two flow steps in W which correspond to the two decimation procedures.

3.2. Flow equations

Let us first study a single-variable decimation, fixing x_i to a value $x \in \{0, 1\}$. The variable x_i disappears from the problem. The constraints a with $a \in \partial i$ are modified: their degree is decreased by 1, as they no longer involve variable i, and their constraint vector is shifted by a constant:

$$s'_a = s_a + w_{a,i} x \qquad \text{for } a \in \partial i. \tag{6}$$

We now turn to a pair decimation operation. Suppose that we fix $x_i = x_j + y \bmod 2$. Explicitly, this amounts to an operation on integers:

$$x_i = x_j + y(1 - 2x_j). \tag{7}$$

Then the variable x_i disappears from the problem, the number of variables is reduced by 1, and all the edges (i, a) for $a \in \partial i$ disappear from the factor graph. The constraints $a \in \partial i$ are modified, but their modification depends on whether $j \in \partial a$ or not.

If a constraint $a \in \partial i$ does not involve j, then j is added to the neighborhood of a. The modification of a is given by

$$E' = E \cup (j, a); \qquad (\partial a)' = (\partial a) \cup (j); \qquad w'_{a,j} = w_{a,i}(1 - 2y); \qquad s'_a = s_a + w_{a,i} y. \tag{8}$$

Notice that the degree k_a of a is unchanged in this case: through decimation, a loses one neighbor (i), and gains another one (j).

If a constraint a involves both i and j, then

$$w'_{a,j} = w_{a,j} + (1 - 2y) w_{a,i}; \qquad s'_a = s_a + w_{a,i} y. \tag{9}$$

In this procedure, the degree of a is decreased by 1 (if $w'_{a,j} \neq 0$), or by 2 (if $w'_{a,j} = 0$, in which case both i and j disappear from ∂a).

Clearly, W is stable through our decimation operations. It is important to notice that, at each step:

- The number of variables decreases.
- The degree k_a of each constraint is non-increasing.
- The function A_a is unchanged.
- The sum $\sum_{i \in \partial a} |w_{a,i}|$ is non-increasing for each constraint.

The flow generated through these operations can be seen as a kind of renormalization group flow. If we start from a LOP where each constraint a has degree $k_a = k$, this flow stays within a finite subspace of W, where $k_a \leq k$ and $\sum_{i \in \partial a} |w_{a,i}| + s_a \leq k$. These properties allow us to use this type of flow as an effective algorithm.

doi:10.1088/1742-5468/2009/12/P12009

J. Stat. Mech. (2009) P12009

3.3. Special constraints

When applying the flow process, one may encounter constraints which present some peculiarities which should be noticed.

Irrelevant variables. First of all it may happen that a variable is irrelevant for some constraint. With some appropriate labeling of the variables, consider the constraint $A(\sum_{i=1}^{k} w_i x_i + s) = 1$. Variable 1 is irrelevant if and only if

$$\forall \{x_2, \ldots, x_k\} \in \{0,1\}^{k-1}: \qquad A\left(\sum_{i=2}^{k} w_i x_i + s\right) = A\left(w_1 + \sum_{i=2}^{k} w_i x_i + s\right). \tag{10}$$

When a variable is irrelevant for a constraint, it can just be eliminated from this constraint, reducing k by 1, without any other change in the weights or threshold. We shall suppose here that, whenever a constraint is changed by the flow, one checks for possible irrelevant variables and eliminates them.

Linear constraints. Let us consider a constraint $A(\sum_{i=1}^{k} w_i x_i + s) = 1$ which has no irrelevant variable. This constraint is a linear constraint if and only if the set $\mathcal{B}$ of configurations of the k variables $x_1, \ldots, x_k$ which satisfy this constraint can be characterized by an affine subspace in GF(2). This is the case whenever there exists a number $b \in \{0,1\}$ such that $\mathcal{B}$ consists of all solutions of the equation $\sum_{i=1}^{k} x_i = b (\mathrm{mod}\, 2)$. A typical case where this happens is when $w_i = 1$ and the vector A is either given by $\forall\ r: A(r) = \frac{1}{2}(1 - (-1)^r)$ or by $\forall\ r: A(r) = \frac{1}{2}(1 + (-1)^r)$. But it may happen that a constraint is linear in a slightly less obvious way. For instance consider the constraint on $k = 2$ variables characterized by $w_1 = -2, w_2 = 3$, $s = 2$ and the vector $A = (110001)$. This is a linear constraint (which can be rewritten as $x_1 + x_2 = 1 (\mathrm{mod}\, 2)$). As there are only two linear constraints on GF(2) (with $b = 0$ and 1), one can identify whether a constraint is linear by comparing its truth table to each of these two linear constraints. This takes of order $k 2^{k+1}$ operations and can be done for the relatively small values of k that we study here.

Variable-polarizing constraints. Consider the constraint $A(\sum_{i=1}^{k} w_i x_i + s) = 1$. This constraint is said to be polarizing for the variable x_1 if and only if there exists $\tau \in \{0,1\}$ such that

$$\forall \{x_2, \ldots, x_k\} \in \{0,1\}^{k-1}: \qquad A\left(\sum_{i=2}^{k} w_i x_i + s\right) = \tau = 1 - A\left(w_1 + \sum_{i=2}^{k} w_i x_i + s\right). \tag{11}$$

If $\tau = 1$, the constraint imposes that x_1 should be equal to 1. If $\tau = 0$, it imposes that x_1 should be equal to 0.

Pair-polarizing constraints. Consider the constraint $A(\sum_{i=1}^{k} w_i x_i + s) = 1$. This constraint is said to be polarizing for the pair x_1, x_2 if and only if there exists $\tau \in \{0,1\}$ such that

$$\forall \{x_3, \ldots, x_k\} \in \{0,1\}^{k-2}: \begin{cases} A\left(\sum_{i=3}^{k} w_i x_i + s\right) = A\left(w_1 + w_2 + \sum_{i=3}^{k} w_i x_i + s\right) = \tau \\ A\left(w_1 + \sum_{i=3}^{k} w_i x_i + s\right) = A\left(w_2 + \sum_{i=3}^{k} w_i x_i + s\right) = 1 - \tau. \end{cases} \tag{12}$$

If $\tau = 1$, the constraint imposes that $x_1 + x_2 = 0 \pmod 2$. If $\tau = 0$, it imposes that $x_1 + x_2 = 0 \pmod 2$.

4. How to find polarized variables and correlated pairs

In this section we explain how to choose a variable, or a pair of variables, to which the flow operations are applied.

One first natural class is found when the flow itself generates variable- or pair-polarizing constraints as defined in section 3. As soon as the flow generates a constraint, it can be tested with respect to these two criteria, and if the constraint is found to be variable- or pair-polarizing, one performs the corresponding variable or pair decimation. Notice that the variable decimation process induced in such a way is equivalent to the well known 'warning propagation' procedure [4, 12], which is frequently used with BP and SP in order to infer the consequences of fixing a variable.

Next we consider the subset of all the linear constraints defined in section 2. This subset obviously constitutes a necessary condition for the constraint satisfaction problem at hand. That condition can in principle imply relations of the form (3) or (4). If the standard Gaussian elimination on GF(2) is applied on the subset of linear constraints, all the variables that can be solved in the form (3) are found. Some of the two-variable relations (4) can also be found even if x_i and x_j are not neighbors. The set of pairs found in this way depends on the detailed order of operations in the Gaussian elimination. If more than one such variable or pair is found, one performs variable or pair decimation. This step does more than the standard warning propagation. It can find some distant but completely correlated pairs in a single step. How often this occurs depends on the nature of instances.

When the above procedures do not impose any strict relations on variables or pairs, one must estimate the single-variable marginals $P_i(x) = P(x_i = x)$ and the relative pair marginals $P_{ij}(y) = P(x_i + x_j = y)$. The standard methods for finding polarized variables are BP [28] and SP [4]. In fact, application of the single-variable decimation defined above to a variable which has an extreme one-variable marginal according to BP or SP is equivalent to BP-or SP-guided decimation [12]. In this case, one stays in a subspace where all the weights on the edges are equal to 1.

To find a highly correlated pair, one should resort to some of the recently proposed methods which aim at estimating correlations with message passing [16]–[21]. The application of the second flow operation within the susceptibility propagation method [16, 19] is equivalent to pair decimation [16].

The degree of polarization of a variable i is conveniently measured by the entropy of the marginal, $S_i = -\sum_x P_i(x) \log P_i(x)$. Similarly, we define the relative pair entropy as $S_{ij} = -\sum_y P_{ij}(y) \log P_{ij}(y)$. The basic step of the algorithm is thus finding a strongly polarized (low entropy) variable or pair, as in [16]. On top of this primary goal, one can also decide to favor the decimation steps which lead to a larger fraction of linear constraints. In the most general case within our setting, one just decides what operation to do (variable or pair fixing) by looking for variables (resp. pairs) with small S_i (resp. S_{ij}) such that the corresponding fixing induces some linear constraints. Many kinds of penalty functions can be used, putting more or less weight on the variables, the pairs, or the induction of linear constraints. In the next section we shall show that the result

doi:10.1088/1742-5468/2009/12/P12009

J. Stat. Mech. (2009) P12009

obtained with some of the simplest schemes is already much better than those from all existing methods for some classes of hard LOPs.

5. Numerical experiments

5.1. The algorithm

In our experiments we have used a very simple version of the general class of algorithms defined above. In this version, when we generate a new constraint, we check whether it is a variable-polarizing constraint, and if this is the case we implement the corresponding variable decimation step. If the constraint has degree 2, we also check whether it is a pair-polarizing constraint, and if this is the case we implement the corresponding pair decimation step. Then we check for the linear constraints and run Gaussian elimination on them. (Notice that these steps can also be seen as a simple 'on-line' exploitation of the linear structure of the problem.) In all other situations we run belief propagation, estimate the corresponding single-variable entropies S_i, and fix the variable i with the smallest entropy. When all the constraints are linear, we stop the decimation process and apply a Gaussian elimination procedure to the corresponding linear problem. When the decimation leads to a problem with a small enough number of variables, we perform an exhaustive search. The structure of the program is thus the following:

`Input:` factor graph with N nodes, constraint vector A, convergence criterion ϵ, initial messages.

`Output:` a satisfying assignment (or FAIL-NOT-FOUND).

- Repeat until the instance either has less than D variables, or is equivalent to a linear system:
  - * Initialize the messages.
  - * Iterate the BP update equations until all messages have converged within precision ϵ.
  - * Using the BP messages: for all the variables i of the instance, compute the local marginal $P_i(x_i)$ and the entropy estimate S_i
  - * Find the variable i with the smallest entropy S_i, fix it to the value x_i which is the most probable according to $P_i(x)$. Fix the number of changes, n_c, at $n_c = 1$.
  - * While $n_c > 0$:
    - + Initialize $n_c = 0$.
    - + For all the constraints a in the instance:
      - For all the variables i involved in the constraint a, check whether i is irrelevant, using the criterion (10). If i is irrelevant, eliminate it from the constraint a.
      - If the constraint a contains only irrelevant variables: if it is automatically satisfied, eliminate this constraint; if it is never satisfied: return FAIL-NOT-FOUND.
      - For all the variables i involved in the constraint a, check whether a is polarizing for i, using the criterion (11). If this is the case, fix the variable i to the value forced by a, update the constraint vector using (6), and increase n_c by 1.

J. Stat. Mech. (2009) P12009

+ For all the constraints a in the instance with degree $|\partial a| = 2$: check whether a is a pair-polarizing constraint, using the criterion (12). If this is the case, perform the corresponding pair decimation steps (7)–(9), and increase n_c by 1.
+ For all the constraints a in the instance: if a is a linear constraint described by an alternating vector $A = (010101\ldots)$ or $A = (101010\ldots)$, include it in the set $\mathcal{S}$.
+ If $\mathcal{S}$ is non-empty, run Gaussian elimination on the subproblem involving all the constraints in $\mathcal{S}$, and all the variables i connected to these constraints:
  - If the linear problem has no solution return FAIL-NOT-FOUND
  - Else, if the linear space of solutions has dimension d, find a set of d linearly independent variables and express all the other variables in terms of the variables in this set. Increase n_c by the number of variables fixed in this Gaussian elimination.
+ If the linear problem admits a solution, return a satisfying assignment.
+ Else return FAIL-NOT-FOUND.

* When the number of variables is equal to or smaller than D: perform an exhaustive search for satisfying assignments. If found:
+ Then return the satisfying assignment.
+ Else return FAIL-NOT-FOUND.

It is not possible to state strictly the complexity of the algorithm, as it involves some steps, like the BP iteration, which do not always converge. So the main results on running time will be obtained from numerical experiments. However one can make a few general comments on the scaling with size of the number of operations. The number of variables is strictly decreasing at each iteration of the main loop. Therefore the main loop will be performed at most N times. It contains three basic ingredients: the first one is the iteration of BP. In the cases which we have considered one finds a fixed point of BP (with accuracy ϵ) in a number of operations which is either constant or grows logarithmically with the size of the system. The second one involves the search for various polarizing constraints; it implies a number of operations which grows like the number of constraints multiplied by their degree, i.e. it scales linearly with the size of the system. The last ingredient is the Gaussian elimination for the constraints in $\mathcal{S}$; this involves at most $O(N^3)$ operations. Altogether, one can thus expect that the algorithm will run in a time that is $\leq O(N^3)$.

5.2. Class of problems

We have studied various LOPs defined from random factor graphs, uniformly chosen among the graphs with the following degrees. All function nodes have degree k and the variables have random degrees chosen from the truncated Poisson degree distribution

$$q(\ell) = \begin{cases} 0 & (\ell = 0, 1) \\ \dfrac{\mathrm{e}^{-c} c^{\ell}}{\ell!(1 - (1 + c)\mathrm{e}^{-c})} & (\ell \geq 2), \end{cases} \tag{13}$$

for which the average degree is

$$\bar{\ell} = \sum_{\ell=0}^{\infty} \ell q(\ell) = \frac{c(1-\mathrm{e}^{-c})}{(1-(1+c)\mathrm{e}^{-c})}. \tag{14}$$

The number of function nodes M and the number of variables N are related by $Mk = N\bar{\ell}$. We are interested in the thermodynamic limit where N and M go to infinity at fixed $k, \bar{\ell}$.

The phase diagram of these problems has been studied in [22, 23]. Like most random constraint satisfaction problems, they exhibit two main phase transitions when the average degree $\bar{\ell}$ increases (this increases the density of constraints). For $\bar{\ell} < \ell_{\mathrm{d}}$ the problem is typically easy; for $\ell_{\mathrm{d}} < \bar{\ell} < \ell_{\mathrm{s}}$ there exists with probability 1 (in the large N limit) a solution, but the space of solutions is made up of isolated configurations which are very far away from each other. For $\bar{\ell} > \ell_{\mathrm{s}}$ the problem has no solution with probability 1. In problems like 1-or-3-in-5 SAT (described by the vector $A = 010100$) or 1-or-4-in-5 SAT (described by the vector $A = 010010$), the usual BP + decimation or BP + reinforcement algorithms are unable to find solutions when they are applied to instances in the intermediate phase $\ell_{\mathrm{d}} < \bar{\ell} < \ell_{\mathrm{s}}$. We have thus focused on this region.

5.3. Results

The LOP 1-or-3-in-5 satisfiability is a candidate of choice for exploiting the idea of flow towards a linear system, because if a variable connected to a given factor is fixed to 0, the corresponding factor is transformed into a 1-or-3-in-4 constraint, which is a linear node. Figure 1 shows the result of experiments for this problem, with the algorithm of section 5.1 used with $D = 16$.

It can be observed that the algorithm works well in the lower half $\bar{\ell} \precsim 4.0$ of the clustered phase. By inspecting the decimation process in this lower half closely, we find that all the instances flow to a linear problem, consisting only of 1-or-3-in-4 constraints, in approximately M steps. This means that, in the early stage, a variable in the neighborhood of each constraint is fixed to 0. Then the Gaussian elimination finds some solutions. (Notice that at this stage the problem is linear and all the one-variable marginals are unpolarized [6]; therefore, if instead of using Gaussian elimination one were to keep on with BP, the next decimation steps would be taken totally at random.) On the other hand, on the upper half $\bar{\ell} \succsim 4.0$, the algorithm also flows to a linear problem but it turns out that this problem has no solution. When $\bar{\ell} < \ell_{\mathrm{s}}$, we know *a priori* that the instances all have satisfiable solutions with probability 1. This means that the algorithm has made a fatal error in guessing in the early stage.

While the new algorithm is still not able to reach the satisfiability threshold, it widely increases the set of instances for which one can find a solution, with respect to existing algorithms. The amount of improvement can be seen from figure 2, which shows the size dependence of the typical time needed to solve an instance of a 1-or-3-in-5 problem, with average degree $\bar{\ell} = 3.6$, with various algorithms. In order to compare to some standard approaches, we have converted each instance into an instance of satisfiability, as in [23]. The satisfiability formula obtained can be studied with two main strategies. The first one consists in incomplete local search algorithms, which have been shown to be rather powerful for random satisfiability formulae [29]–[31]. In this category we have used the

J. Stat. Mech. (2009) P12009

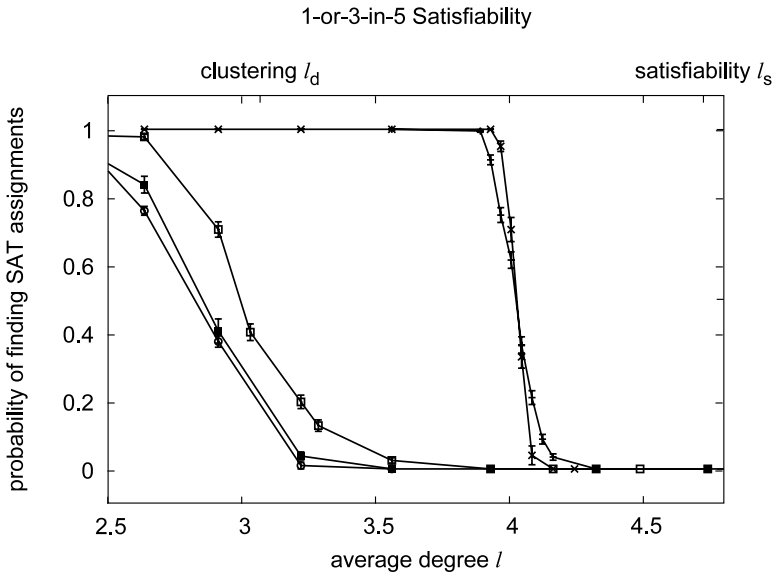

Figure 1. Probability of finding a solution for 1-or-3-in-5 satisfiability, plotted versus the average degree $\bar{\ell}$. The clustering threshold $\ell_{\rm d}$ and the satisfiability threshold $\ell_{\rm s}$ are shown by vertical lines. The results from our new algorithm are the two curves on the right, obtained with $M = 500$ (400 samples), and 2000 (200 samples). The data show a threshold behavior around $\bar{\ell} = 4$: the transition becomes sharper when M increases. The three curves on the left, displaying a smoother crossover around $\bar{\ell} = 3$, are obtained with the single-variable decimation procedure, without Gaussian elimination, for $M = 500$, 200 and 100 (from left to right), both averaged over 400 samples. With this standard BP decimation it is very hard to find a solution for instances in the clustered region $\bar{\ell} > \ell_{\rm d} = 3.07$. In both algorithms we use three restarts.

standard Walksat procedure [29]. The second strategy is that of complete solvers using the DPLL approach [32, 33], for which we have written a simple code. The performances of walksat, of DPLL, and of our decimation + Gaussian elimination procedures shown in figure 2 clearly show that this problem is hard for standard approaches, but our following algorithm can handle much larger instances. Notice that we have not tried to fully optimize any of the three codes, as we are mainly interested in the qualitative improvement that can be obtained by Gaussian elimination. We have observed that this improvement also occurs in the LOPs 1-or-3-in-6 and 2-or-4-in-6.

On the other hand, the improvement is much less impressive for the following problems: 1-in-3, 1-in-4, 1-in-5, 1-in-6, 2-in-4, 2-in-5, 2-in-6, 3-in-6,1-or-4-in-5, 1-or-5-in-6. For instance, figure 3 shows the performance for 2-in-5. The reason for this relatively poor performance is that these problems typically do not flow to XORSAT. In this case the pair decimation algorithm that we have studied in [16] has a probability of success similar to that of the new algorithm using Gaussian elimination. The present situation concerning the locked occupation problems is rather paradoxical. Nearly all of these problems are NP-hard, with the noticeable exception of the systems which can be written as linear systems, for which Gaussian elimination provides a polynomial algorithm. Message passing decimation, when it works, is very fast as it typically requires a time of

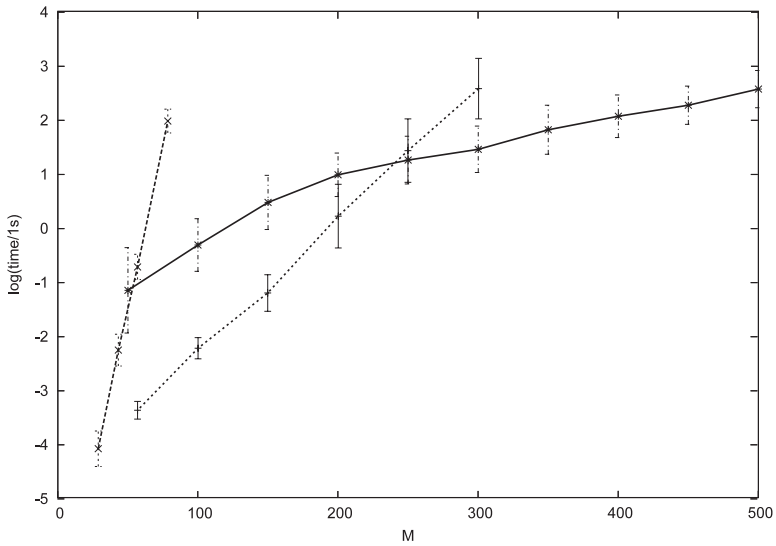

Figure 2. Average of the log of the computer time (in seconds) used to solve an instance of the 1-or-3-in-5 problem, with average degree $\bar{\ell} = 3.6$, versus size of the instance. The full curve is for the decimation flow with Gaussian elimination. The two dashed curves are for the Walksat algorithm run for the corresponding satisfiability formula (left) and the DPL-type algorithm (right); they display a clear exponential behavior of the typical time versus size. The averages are computed with 100 instances.

order N^2, which can even be decreased to $O(N/logN)$ by simultaneously fixing groups of variables at each decimation step. However, it typically fails on the subspace of linear systems. The reason is that the messages, when they are not constraining a variable, are typically unbiased: they give no clue as to what variable to fix, and to which value. A trace of this difficulty remains in the problems which are 'close' to linear systems. This is the case for instance for 1-or-3-in-5 satisfiability. This is not a linear problem, but 1-or-3-in-4 is linear. It has been found in previous studies that 1-or-3-in-5 is very hard for message passing [16, 23], again because the messages lead to marginals which have too small biases to be really useful (taking into account that marginals based on message passing are inherently imperfect). It is also very hard for all other methods. The present approach aims at exploiting the nearness of such a problem to a linear system by using a RG decimation flow until it hits the manifold of linear problems. For 1-or-3-in-5 and for 2-or-4-in-6 this is by far the best method known so far. The other problems for which our strategy is less useful are those which are in some sense too far from the subspace of linear systems: we do not succeed in hitting this manifold with the decimation flow. In these cases the improvement with respect to standard decimation is thus only due to the 'on-line' use of linear constraints when one identifies pair-polarizing or linear constraints and performs the subsequent decimation. The corresponding performances are already much better than the standard single-variable decimation one, but do not quite reach the performance of the more general pair decimation process that we used in [16].

J. Stat. Mech. (2009) P12009

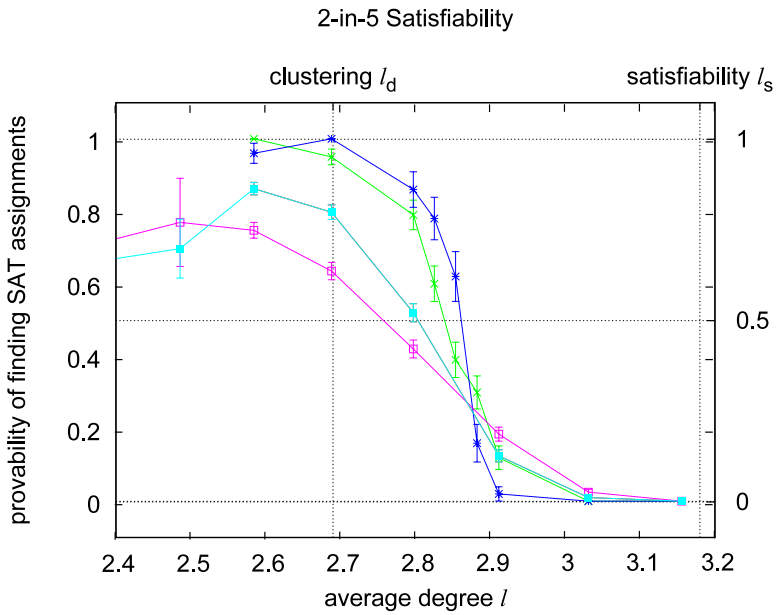

Figure 3. Probability of finding a solution for 2-in-5 satisfiability, plotted versus the average degree $\bar{\ell}$. From top to bottom at $\ell = 2.8$ the results from our new algorithm with $M = 2000$ (50 samples), 1000 (100 samples) and 500 (200 samples), and those of standard BP decimation with $M = 1000$, 500 and 200 (400 samples) are shown. For this problem the improvement obtained with the new algorithm is still present, but less impressive than in the case of 1-or-3-in-5 satisfiability, because the algorithm very rarely uses Gaussian elimination.

6. Conclusions

The random LOPs are typically hard to solve in their clustered phase. An exception is given by linear problems, where all the constraints can be written as linear equations over GF(2), which can be solved in polynomial time by Gaussian elimination. In this paper we have shown how one can exploit this peculiarity within the decimation approach. The idea is that, instead of trying to follow the decimation flow all the way until all variables are fixed, one stops the flow whenever it hits the subspace of linear problems, and then uses Gaussian elimination. This idea is also complemented by the use of Gaussian elimination 'on-line' through the detection of polarizing constraints and linear constraints. This strategy has been found to greatly improve the range of problems that can be solved efficiently in the case of 1-or-3-in-5, 1-or-3-in-6 and 2-or-4-in-6 LOPs. In other problems the improvement is mainly due to the on-line elimination and is less impressive, but still present.

The present approach was the simplest implementation of this idea, and one could think of several directions in which to develop it. We have done a few experiments in which we tried to accelerate the flow towards the subspace of linear problems by favoring the decimation steps which lead to more linear constraints. We have seen that this approach typically improves the efficiency, but it is not evident that it improves the threshold in the $N \to +\infty$ limit. More work along this direction needs to be done in order to find the optimal protocol and its threshold. Another important long-term project would be

doi:10.1088/1742-5468/2009/12/P12009

J. Stat. Mech. (2009) P12009

to develop analytic studies of decimation, starting from the studies of ideal decimation in [12, 13].

Acknowledgment

SH was supported by a Ryukoku University Research Fellowship (2008).

References

[1] Cook S, 1971 *Proc. 3rd Annual ACM Symp. on Theory of Computing* (New York: ACM) pp 151–8
[2] Cook S and Mitchell D, 1997 *Satisfiability Problem: Theory and Applications: DIMACS Workshop (March, 1996)* American Mathematical Society p 1
[3] Mézard M, Parisi G and Zecchina R, 2002 *Science* **297** 812
[4] Mézard M and Zecchina R, 2002 *Phys. Rev.* E **66** 56126
[5] Braunstein A, Mézard M and Zecchina R, 2005 *Random Struct. Algorithms* **27** 201
[6] Mézard M and Montanari A, 2009 *Information, Physics and Computation* (Oxford: Oxford University Press)
[7] Hartmann A K and Rieger H (ed), 2004 *New Optimization Algorithms in Physics* (Berlin: Wiley–VCH)
[8] Percus A, Istrate G and Moore C (ed), 2006 *Computational Complexity and Statistical Physics* (Oxford: Oxford University Press)
[9] Braunstein A, Mulet R, Pagnani A, Weigt M and Zecchina R, 2003 *Phys. Rev.* E **68** 036702
[10] Krzakala F, Montanari A, Ricci-Tersenghi F, Semerjian G and Zdeborová L, 2007 *Proc. Nat. Acad. Sci.* **104** 10318
[11] Chavas J, Furtlehner C, Mézard M and Zecchina R, 2005 *J. Stat. Mech.* P11016
[12] Montanari A, Ricci-Tersenghi F and Semerjian G, 2007 arXiv:0709.1667
[13] Ricci-Tersenghi F and Semerjian G, 2009 *J. Stat. Mech.* P09001
[14] Houdayer J and Martin O C, 1999 *Phys. Rev. Lett.* **83** 1030
[15] Coppersmith S, 2007 *Europhys. Lett.* **77** 30006
[16] Higuchi S and Mézard M, 2009 arXiv:0903.1621
[17] Montanari A and Rizzo T, 2005 *J. Stat. Mech.* P10011
[18] Rizzo T, Wemmenhove B and Kappen H, 2007 *Phys. Rev.* E **76** 011102
[19] Mézard M and Mora T, 2009 *J. Physiol. Paris* **103** 107–13
[20] Tanaka K, 2003 *IEICE Trans. Inform. Syst.* **E86-D** 1228
[21] Welling M and Teh Y W, 2004 *Neural Comput.* **16** 197
[22] Zdeborová L and Mézard M, 2008 *Phys. Rev. Lett.* **101** 078702
[23] Zdeborová L and Mézard M, 2008 *J. Stat. Mech.* P12004
[24] Schaefer T, 1978 *Proc. 10th STOC* (New York: ACM) pp 216–26
[25] Kschischang F, Frey B and Loeliger H, 2001 *IEEE Trans. Inform. Theory* **47** 498
[26] Zdeborová L, 2008 *PhD Thesis* Université Paris-Sud arXiv:0806.4112
[27] Raymond J, Sportiello A and Zdeborová L, 2007 *Phys. Rev.* E **76** 11101
[28] Yedidia J, Freeman W and Weiss Y, 2003 *Understanding Belief Propagation and its Generalizations* (Amsterdam: Elsevier Science & Technology Books) pp 239–6
[29] Selman B, Kautz H A and Cohen B, 1996 *Proc. 2nd DIMACS Challenge on Cliques, Coloring, and Satisfiability (Providence RI)* ed M Trick and D S Johnson citeseer.ist.psu.edu/article/selman96local.html
[30] Seitz S, Alava M and Orponen P, 2005 *J. Stat. Mech.* P06006
[31] Alava M, Ardelius J, Aurell E, Kaski P, Krishnamurthy S, Orponen P and Seitz S, 2008 *Proc. Nat. Acad. Sci.* **105** 15253
[32] Davis M and Putnam H, 1960 *J. Assoc. Comput. Mach.* **7** 201
[33] Davis M, Logemann G and Loveland D, 1962 *Commun. ACM* **5** 394