Belief Propagation for Graph Partitioning*

Parsa Rangriz†
Department of Physics,
Sharif University of Technology,
Tehran, Iran

Sep 2021

Abstract

In graph theory, the minimum cut problem is one of the first basic optimization problems which has deep communication with statistical physics. In fact, it could be shown that computing the ground state energy of the related Ising model is equal to find the solution of what is desired. The problem is known to be NP-complete and for this reason, we use some approximate methods such as belief propagation algorithm to come up with the issue. After using this iterative algorithm for random d-regular graphs and Erdős–Rényi model, we can see the BP algorithm's validation region. Then it would be possible to find out about some phase transitions that will raise, giving us more intuitions about the behavior of the graph partitioning problem.

1 Introduction

Suppose a graph $G = (\mathcal{V}, \mathcal{E})$ with N vertices which is to be cut into two subgraphs with specific value of vertex difference and subject to minimize the number of connections between vertices from different groups. This is called min-cut problem.

As far we know, this a NP-complete problem and we are not able to solve it exactly. But here we will show that this problem is equivalent to finding the ground state of a corresponded Ising model. But how? This would be possible if we set a spin S_i variable to each node $i \in \mathcal{V}$. Then this spin is equal to $+1$ if it is in the first group or -1 otherwise. Thus we define the Hamiltonian as follows

$$H(\{S_i\}) = -J \sum_{(i,j)\in\mathcal{E}} S_i S_j \tag{1}$$

When we seek to find the minimum of the above Hamiltonian, indeed we try to find out the solution of the min-cut problem. But how we should apply the vertex difference? For this, we define the magnetization to be a constant value corresponded to the Hamiltonian. This magnetization is given by

$$m = \frac{1}{N} \sum_{i\in\mathcal{V}} S_i \tag{2}$$

The Hamiltonian that we want to calculate is bounded with the desired magnetization. To come up with this problem, we can introduce a Lagrangian with the corresponding multiplier, which is the magnetic field, h. In fact, the Lagrangian is given by

$$L = H - hm \tag{3}$$

where $L = H_h$ in our problem. So, the Lagrangian is nothing just the Hamiltonian of the field-on Ising model.

Hereafter, we change the Hamiltonian which its magnetic field is on. In this case the above Hamiltonian becomes

$$H(\{S_i\}) = -J \sum_{(i,j)\in\mathcal{E}} S_i S_j - h \sum_{i\in\mathcal{V}} S_i \tag{4}$$

This Hamiltonian have the same solution if h is related to m.

The main aim of this paper is find out the value of bisection width, b, which is means the number of edges between different groups divided by the total number of vertices.

$$b = \frac{1}{2N} E + \frac{1}{2} hm + \frac{1}{4} \alpha \tag{5}$$

where α is the mean degree of the graph.

*Based on Petr Šulc and Lenka Zdeborová 2010 J. Phys. A: Math. Theor. 43 285003

†rangriz@physics.sharif.edu

2 Replica Symmetric Approach

As we discussed above, the problem of graph partitioning is equal to the Ising model with the following Hamiltonian,

$$H(\{S_i\}) = -J \sum_{(i,j)\in\mathcal{E}} S_i S_j - h \sum_{i\in\mathcal{V}} S_i. \tag{6}$$

So what are the relation between the magnetic field, h, and the Boltzmann's distribution? The answer is related to a theorem which we studied in the exponential families. The theorem says that if we want to have the constraint such that the proper value of the statistical function (which here is the Lagrangian, H_h) becomes a special quantity, then the distribution that respects this case is the exponential families. The exponential family is given by

$$P_a(x) = \exp(\langle a, x\rangle - A(a)) \tag{7}$$

where A is the log-partition function.

In physical systems, we are concerned about a special case of this family, namely the Boltzmann's distribution. So the inverse temperature, $(1/T) = \beta$, roles as the a vector, here.

By supposing that these spins are based on the Boltzmann distribution,

$$p(\{S_i\}) = \frac{1}{Z} e^{-\beta H(\{S_i\})}, \tag{8}$$

where Z is the partition function and $\beta = \frac{1}{T}$ is the inverse temperature. Thus we have a pairwise graphical model,

$$p(\{S_i\}) = \frac{1}{Z} \prod_{(i,j)\in\mathcal{E}} e^{\beta J S_i S_j} \prod_{i\in\mathcal{V}} e^{\beta h S_i}. \tag{9}$$

Also, the graph partitioning problem corresponds to $\beta \to \infty$. As we know in statistical physics, if we could calculate the partition function, it would be possible to obtain any thermodynamic variables such as the magnetization, free energy and so on. By this, our aim is to compute

$$Z = \sum_{\{S_i\}} \frac{1}{Z} \prod_{(i,j)\in\mathcal{E}} e^{\beta J S_i S_j} \prod_{i\in\mathcal{V}} e^{\beta h S_i}. \tag{10}$$

2.1 The Sum-Product Algorithm

As far we know, it takes too long to do so naively by summing each term of the above equation. For this reason, there are some algorithms such as the sum-product [2] (which is known as the cavity method in the physics literature) that help us to use some methods for the calculation of each marginal distribution. Thus for any node and edge, the corresponding distribution is given by

$$p_i(S_i) = e^{\beta h S_i} \prod_{j\in\partial i} m_{j\to i}(S_i) \tag{11}$$

$$p_{ij}(S_i, S_j) = e^{\beta h S_i} e^{\beta h S_j} e^{\beta J S_i S_j} \prod_{k\in\partial i\setminus j} m_{k\to i}(S_i) \prod_{l\in\partial j\setminus i} m_{l\to j}(S_j) \tag{12}$$

where ∂i is the set of S_i's neighbors and the messages are defined as follows,

$$m_{i\to j}(S_i) = \frac{1}{Z_{i\to j}} e^{\beta h S_i} \prod_{k\in\partial i\setminus j} \left(\sum_{S_k} e^{\beta J S_i S_k} m_{k\to i}(S_k) \right) \tag{13}$$

and $Z_{i\to j}$ are some normalization constants ensuring $m_{i\to j}(+1) + m_{i\to j}(-1) = 1$.

2.2 The Bethe Method

For calculating the partition function, it is equal to compute the free energy, $F = -\frac{1}{\beta} \ln Z$, so we seek to obtain the internal energy E and the entropy S of the Ising model because,

$$F = E - \frac{1}{\beta} S \tag{14}$$

From [3], we have

$$E = -\sum_{(i,j)\in\mathcal{E}} \sum_{S_i,S_j} p_{ij}(S_i, S_j) J S_i S_j + \sum_{i\in\mathcal{V}} (|\partial i| - 1) \sum_{S_i} p_i(S_i) h S_i \tag{15}$$

$$S = -\sum_{(i,j)\in\mathcal{E}} \sum_{S_i,S_j} p_{ij}(S_i,S_j)\ln p_{ij}(S_i,S_j) + \sum_{i\in\mathcal{V}}(|\partial i|-1)\sum_{S_i} p_i(S_i)\ln p_i(S_i) \tag{16}$$

The Bethe free energy is given by

$$-\beta F = \ln Z = \sum_{i\in\mathcal{V}} \ln Z_i - \sum_{(i,j)\in\mathcal{E}} \ln Z_{ij} \tag{17}$$

where

$$Z_i = \sum_{S_i} e^{\beta h S_i} \prod_{k\in\partial i}\left(\sum_{S_k} e^{\beta J S_i S_k} m_{k\to i}(S_k)\right) \tag{18}$$

$$Z_{ij} = \sum_{S_i,S_j} e^{\beta J S_i S_j} m_{i\to j}(S_i) m_{j\to i}(S_j) \tag{19}$$

2.3 Zero Temperature Limit

To understand the configuration of the local minimum energy, we need to go to the zero temperature in the case that the local minimum energy is zero. To do this we use the Legendre transformation by changing the role of m (magnetization) with h(magnetic field).

$$H_h = H_m - hm \tag{20}$$

This hm is a constant which gives us the zero minimum energy for the zero-temperature limit. Then the question is, how to calculate the energy shift? The answer is by using the self-consistency equation which gives us the iteratively answer for h from the cavity fields, $(h_{i\to j})$. Indeed, these fields are trying to find out the energy shift.

Then from the obtained field, h, we can now compute the energy of H_m which is the main problem in our problem in the following way:

$$H_m = H_h + hm \tag{21}$$

One may calculate H_h by using the zero-limit temperature equations. Then the problem is solved. In order to write the zero temperature limit, $\beta\to\infty$, the above equations lead us to define a new parameter, namely cavity fields $h_{i\to j}$,

$$e^{2\beta h_{i\to j}} = \frac{m_{i\to j}(+1)}{m_{i\to j}(-1)} \tag{22}$$

Therefore, with respect to the above equations,

$$e^{2\beta h_{i\to j}} = e^{2\beta h}\frac{\prod_{k\in\partial i\setminus j}\left(\sum_{S_k} e^{\beta J S_k} m_{k\to i}(S_k)\right)}{\prod_{k\in\partial i\setminus j}\left(\sum_{S_k} e^{-\beta J S_k} m_{k\to i}(S_k)\right)} \tag{23}$$

If we apply the logarithm function to both sides of the above equation, then

$$\begin{aligned}
2\beta h_{i\to j} &= 2\beta h + \sum_{k\in\partial i\setminus j}\left(\ln\sum_{S_k} e^{\beta J S_k} m_{k\to i}(S_k)\right) - \sum_{k\in\partial i\setminus j}\left(\ln\sum_{S_k} e^{-\beta J S_k} m_{k\to i}(S_k)\right)\\
&= 2\beta h + \sum_{k\in\partial i\setminus j}\ln\left(\frac{e^{\beta J} m_{k\to i}(+1) + e^{-\beta J} m_{k\to i}(-1)}{e^{-\beta J} m_{k\to i}(+1) + e^{\beta J} m_{k\to i}(-1)}\right)\\
&= 2\beta h + \sum_{k\in\partial i\setminus j}\ln\left(\frac{m_{k\to i}(+1)}{m_{k\to i}(-1)}\right) + \sum_{k\in\partial i\setminus j}\ln\left(\frac{1+e^{-2\beta J}\frac{m_{k\to i}(-1)}{m_{k\to i}(+1)}}{1+e^{-2\beta J}\frac{m_{k\to i}(-1)}{m_{k\to i}(+1)}}\right)\\
&= 2\beta h + \sum_{k\in\partial i\setminus j} 2\beta h_{k\to i} + \sum_{k\in\partial i\setminus j}\ln\left(\frac{1+e^{-2\beta(J+h_{k\to i})}}{1+e^{-2\beta(J-h_{k\to i})}}\right)
\end{aligned} \tag{24}$$

In the limit of zero temperature ($\beta\to\infty$), the self-consistent equations for messages become

$$h_{i\to j} = h + \sum_{k\in\partial i\setminus j}\left(\max((J+h_{k\to i}),0) - \max(h_{k\to i},J)\right) \tag{25}$$

The internal energy is defined as

$$E = \frac{\sum H e^{-\beta H}}{\sum e^{-\beta H}} = -\frac{\partial}{\partial\beta}\ln Z \tag{26}$$

where,

$$E^0 = \lim_{\beta \to \infty} E \tag{27}$$

First of all, we should know the total form of the internal energy,

$$E = -\frac{\partial}{\partial \beta} \left(\sum_{i \in \mathcal{V}} \ln Z_i - \sum_{(ij) \in \mathcal{E}} \ln Z_{ij} \right) = \sum_{i \in \mathcal{V}} E_i - \sum_{(ij) \in \mathcal{E}} E_{ij} \tag{28}$$

where,

$$E_i = -\frac{\partial}{\partial \beta} \ln Z_i, \quad E_{ij} = -\frac{\partial}{\partial \beta} \ln Z_{ij} \tag{29}$$

As same as what we have done above, for $\beta \to \infty$, the above equations become,

$$E_i = h + J|\partial i| + 2 \sum_{k \in \partial i} \max(0, h_{k \to i}) - 2 \max(h + \sum_{k \in \partial i} (J + h_{k \to i}, 0), \sum_{k \in \partial i} \max(h_{k \to i}, J)) \tag{30}$$

$$E_{ij} = J + 2 \max(0, h_{i \to j}) + 2 \max(0, h_{j \to i}) - 2 \max(J + h_{i \to j} + h_{j \to i}, J, h_{i \to j}, h_{j \to i}) \tag{31}$$

and,

$$E = \sum_{i \in \mathcal{V}} E_i^0 - \sum_{(ij) \in \mathcal{E}} E_{ij}^0 \tag{32}$$

Now, we want to compute the value of magnetization m. For this purpose, it requires that use the definition of magnetization,

$$m = \frac{1}{N} \frac{\sum S e^{-\beta H(S)}}{\sum e^{-\beta H(S)}} = -\frac{1}{N} \frac{\partial}{\partial \beta h} \ln Z \tag{33}$$

For $\beta \to \infty$, one may obtain,

$$m_i = \begin{cases} +1 & h + \sum_{k \in \partial i} (\max(J + h_{k \to i}, 0) - \max(h_{k \to i}, 1)) > 0 \\ -1 & \text{otherwise} \end{cases} \tag{34}$$

and hence,

$$m = \frac{1}{N} \sum_i m_i \tag{35}$$

Will be written...

3 Erdős–Rényi Model

Erdős–Rényi model is a random graph with N vertices and mean degree α in which every edge is present with probability $\frac{\alpha}{N-1}$. The size of the largest component is $gN + \mathcal{O}(N)$, where g satisfies the following equation:

$$g = 1 - e^{-\alpha g} \tag{36}$$

In order to divide the graph into two parts of size $\frac{N(1+m)}{2}$ and $\frac{N(1-m)}{2}$ such that the number of edges between the two is zero, the size of the largest component g must be at maximum $\frac{1+m}{2}$ then

$$\alpha_s = -\frac{2}{1+m} \log \frac{1-m}{2} \tag{37}$$

This yields that for $\alpha > \alpha_s$ an extensive number of edges needs to be cut in the minimal bipartition. (Figure 1)

Now it is time to use the above equations and techniques in Erdős–Rényi model. For this purpose, we should write a program. We sketch our program in the algorithm 1. One can find out after running the algorithm 1, the results show that the bisection width b has a tiny distance with an exact solution. These plots are illustrated in the Figure 2.

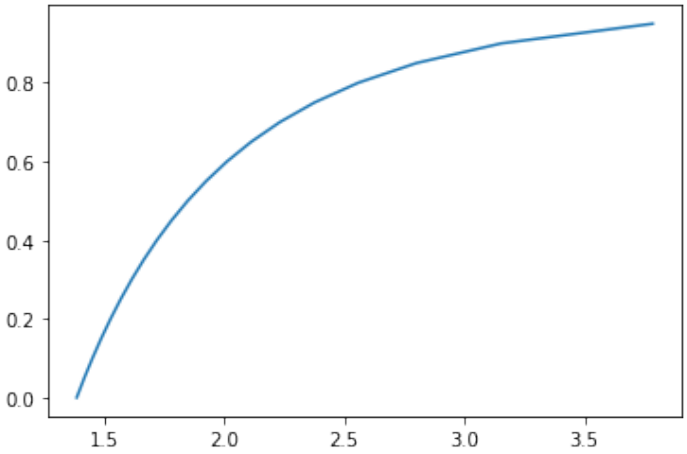


Figure 1: One phase transitions in partitioning of Erdős–Rényi random graphs. The satisfiability threshold α_s above which the giant component have to be cut to bipartition the graph

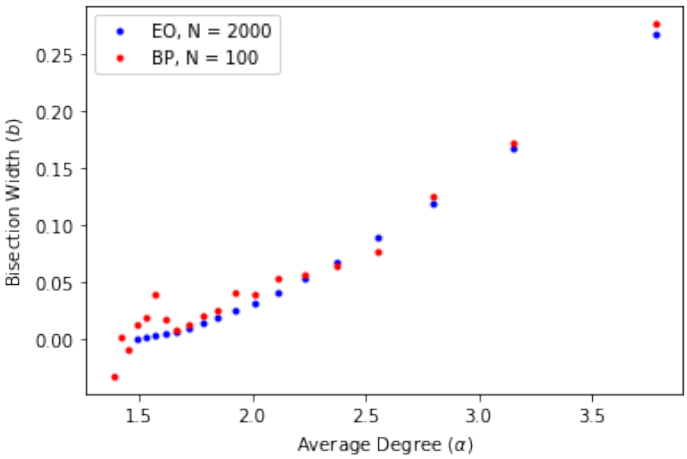


Figure 2: Bisection width b of Erdős–Rényi model as a function of the mean connectivity with $N = 100$ with algorithm 1. The data are compared with the exact average bisection for $N = 2000$ [4].

4 Decimation Guided Algorithm for BP

Although BP algorithm is an incomplete solvers for non-tree graphs but recently, considerable progress has been achieved in building efficient 'incomplete solvers' namely BP-guided decimation. These consist in iterating the following steps:

1) Run a message passing algorithm.

2) Use the result to choose a variable index $(i, j) \in \mathcal{E}$, and a value $h^*_{i\to j}$ for the corresponding variable.

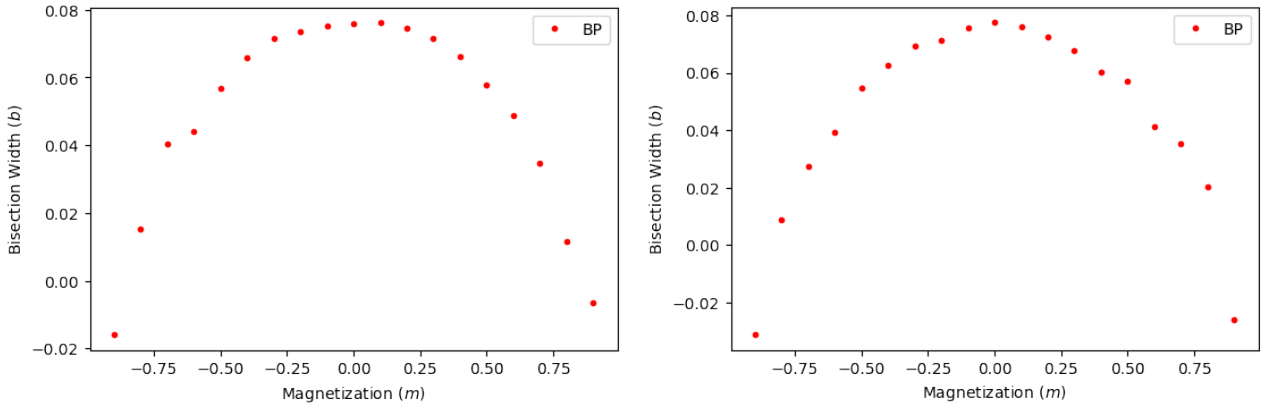
3) Replace the constraint satisfaction problem with the one obtained by fixing $h_{i\to j}$ to $h^*_{i\to j}$.

But there are still a couple of elements we need to specify. First of all, how the BP equations are modified when a non-empty subset U of the variables is fixed. One option would be to eliminate these variables from the factor graph, and reduce the clauses they belong to accordingly. A simpler approach is when $i \in U$. Explicitly, if the chosen value $h_{i\to j}$ satisfies clause a, then we set $h_{i\to j} = \infty$. If it does not, we set $h_{i\to j} = -\infty$ [5]. We use this method in the algorithm 3.

5 Random d-Regular Graphs

Random d-regular graphs are one of the most important graph models which each vertex has d edges and the edges connect the two corresponded vertices in a random way.

Now, as same as the above section, we want to use the BP equations in random d-regular graphs. The program that we should use here is shown in the algorithm 2. The results are illustrated in the Figure 3. Also, with using the algorithm 3, we can show that the bisection width is converged to the specific value as we increase the size of system. (Figure 4)



(a) BP

(b) BP-decimation

Figure 3: Bisection width b of random 3-regular graph as a function of the magnetization m with $N = 1000$ with algorithm 2 and 3

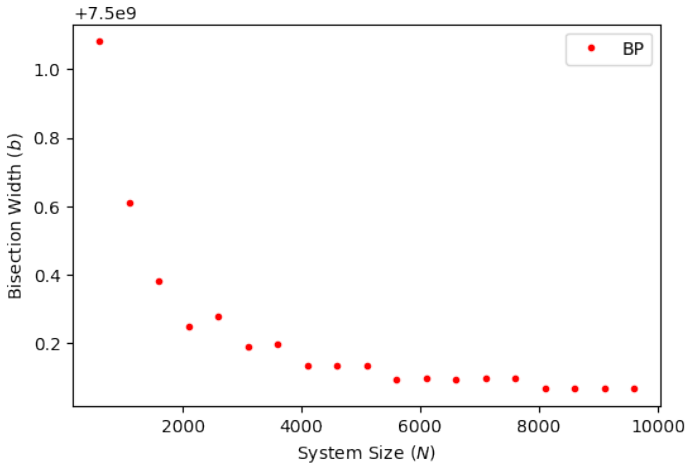


Figure 4: Bisection width b of regular d-random model as a function of the system size with $m = 0$ with algorithm 3.

6 Discussion

By here, we used belief propagation and the decimation version for graph partitioning problem. Indeed, we check the behavior and validation region of BP for both random d-regular graph and Erdős–Rényi model. By our figures we can find out that the BP will give more accurate solution when the graph size is large about 10000 vertices and in our project we just use 100 and 1000 system sizes because of our limitations.

7 Acknowledgment

Here, it is needed to many thanks to Prof. Amir Daneshgar as the instructor of the "Topics in Optimization" course and my supervisor of this course project. Also, I thank Dr. MohammadHoessin Shojaeddin for his help to me in the theoretical part of this paper.

References

[1] P. Šulc, L. Zdeborová, *Belief propagation for graph partitioning*, J. Phys. A: Math. Theor. 43 (2010)
[2] M. Mézard, A. Montanari, *Information, Physics, and Computation*, Oxford University Press (2009)
[3] J.S. Yedidia, W.T. Freeman, Y. Weiss, *Constructing Free Energy Approximations and Generalized Belief Propagation Algorithms*, IEEE Transactions on Information Theory, Volume: 51, Issue: 7 (2005)
[4] Allon G. Percus, et. al., *The peculiar phase structure of random graph bisection*, Journal of Mathematical Physics 49, 125219 (2008)

[5] Andrea Montanari, Federico Ricci-Tersenghi, Guilhem Semerjian, *Solving Constraint Satisfaction Problems through Belief Propagation-guided decimation*, Proceedings of the 45th Annual Allerton Conference on Communication, Control, and Computing (Monticello, IL, USA), 352-359 (2007)

Algorithm 1 Belief propagation for partitioning of the Erdős–Rényi graphs

```
G = networkx.generators.random_graphs.erdos_renyi_graph(N, alpha / (N-1))
h = numpy.random.uniform(-1, 1)
for i ∈ V do
  for j ∈ G[i] do
    cavity_field[i][j] = numpy.random.uniform(-1,1)
  end for
end for
iteration = 0
repeat
  local_field = numpy.zeros(N)
  convergence = 0
  for i ∈ V do
    for k ∈ G[i] do
      local_field += max(1 + cavity_field[k][i], 0) - max(cavity_field[k][i], 1)
    end for
    for j ∈ G[i] do
      new_cavity_field[i][j] = h + local_field[i] - cavity_field[j][i]
      convergence += numpy.abs(new_cavity_field[i][j] - cavity_field[i][j])
      new_cavity_field[i][j] = memory * cavity_field[j][i] + (1 - memory) * new_cavity_field[i][j]
    end for
  end for
  local_field = numpy.sort(local_field)
  index = int(N * (1-m) / 2)
  h = -local_field[index]
until iteration <= maximum OR convergence < ϵ
Calculation of E is the same as the algorithm 1.
return E, h
```

Algorithm 2 Belief propagation for partitioning of d-regular random graphs

```
G = networkx.generators.random_graphs.random_regular_graph(d, N)
h = numpy.random.uniform(-1, 1)
for i ∈ V do
  for j ∈ G[i] do
    cavity_field[i][j] = numpy.random.uniform(-1,1)
  end for
end for
iteration = 0
local_field = 0
repeat
  for i ∈ V do
    for j ∈ G[i] do
      for k ∈ G[i] do
        local_field += max(1 + cavity_field[k][i], 0) - max(cavity_field[k][i], 1)
      end for
      cavity_field[i][j] = h + local_field
      local_field = 0
    end for
  end for
  iteration += 1
  h1 = h - 1
  h2 = h + 1
  while |h_1 − h_2| > ϵ do
    local_field = 0
    m = 0
    for i ∈ V do
      for k ∈ G[i] do
        local_field += max(1+cavity_field[k][i], 0) - max(cavity_field[k][i], 1)
        if h + local_field > 0 then
          m += 1/N
        end if
        if h + local_field < 0 then
          m -= 1/N
        end if
        local_field = 0
      end for
    end for
    if m < M then
      h1 = h
    end if
    if m > M then
      h2 = h
    end ifh = (h1 + h2) / 2
  end while
until iteration <= maximum
Calculation of E is the same as the algorithm 1.
return  E, h
```

Algorithm 3 Belief propagation-guided decimation for partitioning of any graph

```
Run Algorithm 1
iter = 1
S = 0
P = 0
while True do
  E, h, local-field, G, H = d-regular-random (N, d, m, memory = 0.7, epsilon = 1e-6)
  if iter % 2 == 1 then
    I = np.argmax(local-field)
  end if
  for j in G[I] do
    H[I][j] = np.Infinity
  end for
  S += 1
  if iter % 2 == 0 then
    I = np.argmin(local-field)
  end if
  for j in G[I] do
    H[I][j] = -np.Infinity
  end for
  P += -1
  iter += 1
  if np.abs(P) >= m or S >= m then
    break
  end if
end while
return  E, h
```