Prove Np not equal P using Markov Random Field and Boolean Algebra Simplification

Sing Kuang Tan
Email: singkuangtan@gmail.com

June 1, 2021

Abstract

In this paper, we proved that Non-deterministic Polynomial time complexity (NP) is not equal to Polynomial time complexity (P). We developed the Boolean algebra that will infer the solution of two variables of a Non-deterministic Polynomial computation time Markov Random Field. We showed that no matter how we simplified the Boolean algebra, it can never run in Polynomial computation time (NP $\neq$ P). We also developed proof that all Polynomial computation time multi-layer Boolean algebra can be transformed to another Polynomial computation time multi-layer Boolean algebra where there are only 'Not' operations in the first layer. So in the process of simplifying the Boolean algebra, we only need to consider factorization operations that only assumes only 'Not' operations in the first layer. We also developed Polynomial computation time Boolean algebra for Markov Random Field Chain and 2sat problem represented in Markov Random Field form to give examples of Polynomial computation time Markov Random Field.

Contents

1 Introduction

Non-deterministic Polynomial (NP) problem refers to problem that requires exponential number of steps with respect to the size of the input to solve the problem. Whereas Polynomial (P) problem refers to problem that requires

polynomial number of steps with respect to the size of the input to solve the problem. It is an open problem which asks whether it is able to convert a NP problem to P problem ([1–4]). We will show that in this paper that no algorithm (represented as Boolean algebra and Markov Random Field) is able to solve NP problem in Polynomial time as the Boolean algebra cannot be factorized into another Boolean algebra that can be solved in Polynomial time (NP $\neq$ P).

In this paper, the proof of NP $\neq$ P is to show that the Boolean algebra of Markov Random Field of a NP problem cannot be factorized into quadratic form $(A + B + \ldots)(C + D + \ldots)$ and therefore the time complexity of the Boolean algebra cannot be simplified from NP complexity to Polynomial complexity. Note that since the Boolean algebra cannot be factorized into quadratic form, it also cannot be factorized into a more time efficient multi-layer Polynomial computation time Boolean algebra. In the process of factorizing the Boolean algebra, we use only factoring steps that assume there is only 'Not' operations in the first layer and no 'Not' operations in the second and higher layers. We have proved that 'Not' operations in second and higher layers can be simplified into first layer without changing the computation time complexity class, where Polynomial time complexity remains as Polynomial time complexity and NP time complexity remains as NP time complexity.

The prerequisite knowledge to understand this paper is the understanding of Markov Random Field, Boolean algebra and digital gates diagram in digital electronics.

The outline of this paper is how to represent NP problem as Markov Random Field and Boolean algebra (section 2), how to search for Boolean algebra that infers the solution (section 3), how to reformulate the Boolean algebra so that only the first layer contains 'Not' operations while keeping the time complexity as Polynomial (section 4), give an example of a Polynomial computation time multi-layer Boolean algebra for a 7 variables Markov Random Field Chain (section 5), proof that a 6 variables Markov Random Field NP problem cannot be simplified into Polynomial time using Boolean algebra simplification (section 6), generalized the proof to higher number of variables (section 7), and lastly also show how a 2sat problem can be represented in Markov Random Field format together with Boolean algebra which can be solved in Polynomial time (section 8). Finally, I will conclude my whole paper in section 9.

2 Discrete Markov Random Field definition

A NP problem can be represented by a discrete Markov Random Field. It has n variables $a_1, a_2, \ldots, a_n$.

A Markov random field with binary potential values looks something like this

$$f(a_1, a_2, \ldots, a_n) = \prod_{i,j} H_{i,j}(a_i, a_j)$$
$$H_{i,j}(a_i, a_j) = 0 \text{ or } 1.$$

For simplicity, we simply write

$$f(a_1, a_2, \ldots, a_n) = \prod_{i,j} H(a_i, a_j).$$

If $f(a_1 = v_1, a_2 = v_2, \ldots, a_n = v_n) = 1$ means $a_1 = v_1, a_2 = v_2, \ldots, a_n = v_n$ is a solution to the NP problem. Note that $H(a_i, a_j)$ is the same as $H(a_j, a_i)$ and there does not exist $H(a_i, a_i)$ and $H(a_j, a_j)$.

We assume that each variable can take a finite number of values. E.g. $v_i \in \{0, 1, \ldots, 10\}$, in this example it can take 11 values.

If

$$\sum_{v_1, v_2, \ldots, v_n} \prod_{i,j} H(a_i = v_i, a_j = v_j) > 0,$$

then the NP problem has a a solution. For

$$\hat{H}(a_k = v_k, a_l = v_l) = H(a_k = v_k, a_l = v_l) \sum_{v_1, v_2, \ldots, v_n \setminus v_k, v_l} \prod_{(i,j) \setminus (k,l)} H(a_i = v_i, a_j = v_j),$$

if $\hat{H}(a_k = v_k, a_l = v_l) = 0$ after evaluation the above formula, it means that $a_k = v_k$, $a_l = v_l$ is not part of a solution else $a_k = v_k$, $a_l = v_l$ is a solution to this NP problem. Note that $v_1, v_2, \ldots, v_n \setminus v_k, v_l$ means that all values v_1 to v_n not including v_k and v_l. The notation is similar for $(i, j) \setminus (k, l)$, which does not include the $i = k$ and $j = l$ variables.

Example of a four variables 4 coloring problem in Markov Random Field representation. The equation

$$\sum_{a_1, a_2, a_3, a_4} H(a_1, a_2) H(a_1, a_3) H(a_1, a_4) H(a_2, a_3) H(a_2, a_4) H(a_3, a_4) > 0$$

means that there is a solution for this problem. The variable a_i can take values 0,1,2 or 3.

We define

$$\begin{aligned}
&\hat{H}(a_1 = v_1, a_2 = v_2) \\
&= H(a_1 = v_1, a_2 = v_2) \\
&\sum_{a_1 = v_1, a_2 = v_2, a_3 = 0,1,\ldots 3, a_4 = 0,1,\ldots 3} H(a_1, a_3) H(a_1, a_4) H(a_2, a_3) H(a_2, a_4) H(a_3, a_4)
\end{aligned}$$

as the real value of $H(a_1 = v_1, a_2 = v_2)$ after combining all the constraints $H(a_i, a_j)$. The expression $\sum_{a_1 = v_1, a_2 = v_2, a_3 = 0,1,\ldots,3, a_4 = 0,1,\ldots,3}$ means that a_1 and a_2 can only take 1 value, a_3 and a_4 can take integer values between 0 to 3.

$\hat{H}(a_1 = v_1, a_2 = v_2)$ is the solution of this potential taking values v_1 and v_2, output either a value of 0 or 1. If v_1 and v_2 are the solution to this Markov Random Field, then $\hat{H}(a_1 = v_1, a_2 = v_2) = 1$, else it is a 0. Since this is a coloring problem, each potential is a discrete function with the following definition

$$H(a_i, a_j) = \begin{cases} 0 \text{ if } a_i = a_j \\ 1 \text{ if } a_i \neq a_j \end{cases} .$$

$H(a_i, a_j)$ is a discrete function with inputs of discrete a_i, a_j values and output a 0 or 1 value. $H(a_i, a_j)$ will be different for other types of NP problem that is not coloring problem. This Markov Random Field can represent any problem. It can represent other problems simply by using $H(a_i, a_j)$ functions with different 0 and 1 outputs different from the 4 coloring problem.

3 Search for multi-layer Polynomial time complexity Boolean algebra that infers the solution

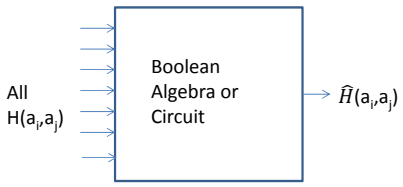


Figure 1: Multi-layer Boolean algebra that infers the solution

Figure 1 shows the Boolean algebra that will infer the solution of $\hat{H}(a_i, a_j)$. This solution represents the actual value of $H(a_i, a_j)$ after considering all the constraints in Markov Random Field. So to solve the NP problem, it is to find an optimal Boolean algebra that has the smallest computational complexity. We can find the optimal Boolean algebra by Boolean algebra simplification. The Boolean algebra has multi-layer operations. The computational complexity of this Boolean algebra can be derived from the number of operations to compute the final output value. If the simplified multi-layer Boolean algebra has Polynomial computational complexity then NP = P. Otherwise, NP $\neq$ P.

4 How to reformat the Boolean algebra so that only the first layer contains 'Not' operations

4.1 Proof of how to reformat the Boolean algebra using recursive Boolean algebra equations

In this section, we are going to prove the lemma below.

Lemma 4.1 *A Polynomial computation time multi-layer Boolean algebra with 'Not' operations at the second or higher layer can be converted to another Polynomial computation time multi-layer Boolean algebra with 'Not' operations only at the first layer.*

The multi-layer boolean algebra can be represented by multiple recursive Boolean algebra equations with each equation in the form of

$$H_i = \bigvee_j H_j \text{ or } H_i = \bigwedge_j H_j \text{ or } H_i = \neg \bigvee_j H_j \text{ or } H_i = \neg \bigwedge_j H_j \text{ where } j < i.$$

And $H_i = H(a_k, a_l)$ for the first layer of this recursive Boolean algebra. H_i is the output of one of the intermediate nodes in the boolean algebra or circuit. Each node is either an 'And' operation or 'Or' operation sometimes with a 'Not' operation at the output of an intermediate node. The equations

$$H_i = \neg \bigvee_j H_j \text{ or } H_i = \neg \bigwedge_j H_j$$

can be simplified to

$$H_i = \bigwedge_j \neg H_j \text{ or } H_i = \bigvee_j \neg H_j$$

where the 'Not' operation is pushed to the previous layer using a De Morgan's theorem. Replace the simplified equations by

$$H_i = \bigwedge_j H'_j \text{ or } H_i = \bigvee_j H'_j.$$

Add additional nodes, $H'_i = \neg H_i$. If $H_i = \neg \bigvee_j H_j$ or $H_i = \neg \bigwedge_j H_j$, then $H'_i = \bigvee_j H_j$ or $H'_i = \bigwedge_j H_j$. If $H_i = \bigvee_j H_j$ or $H_i = \bigwedge_j H_j$, then $H'_i = \bigwedge_j H'_j$ or $H'_i = \bigvee_j H'_j$. Continue push the 'Not' operations until they are at the first layer, $H'_i = \neg H(a_k, a_l)$. So any multi-layer Boolean algebra only need 'Not' operations at the first layer.

4.2 An example of how to reformat a Boolean algebra

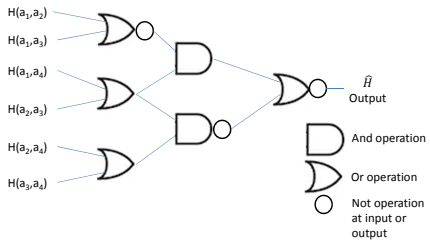
Figure 2 shows a graphical step-by-step example of how to transform a multi-layer Boolean algebra with 'Not' operations at any layers to 'Not' operations only at the first layer.

In the figure 2, the Boolean algebra can be represented in recursive form,

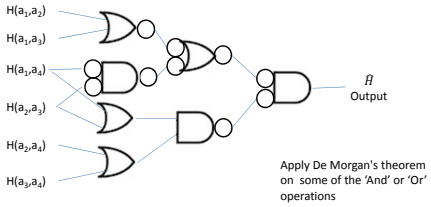
$$H_1 = H(a_1, a_2)\ ,\ H_2 = H(a_1, a_3)\ ,\ H_3 = H(a_1, a_4)\ ,\ H_4 = H(a_2, a_3)\ ,$$
$$H_5 = H(a_2, a_4)\ ,\ H_6 = H(a_3, a_4)\ ,\ H_7 = \neg(H_1 \vee H_2)\ ,\ H_8 = H_3 \vee H_4\ ,$$
$$H_9 = H_5 \vee H_6\ ,\ H_{10} = H_7 \wedge H_8\ ,\ H_{11} = \neg(H_8 \wedge H_9)\ ,\ H_{12} = \neg(H_{10} \vee H_{11})$$
$$\hat{H} = H_{12}.$$

The complements are

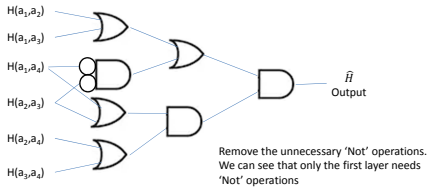
$$H'_1 = \neg H(a_1, a_2)\ ,\ H'_2 = \neg H(a_1, a_3)\ ,\ H'_3 = \neg H(a_1, a_4)\ ,\ H'_4 = \neg H(a_2, a_3)\ ,$$
$$H'_5 = \neg H(a_2, a_4)\ ,\ H'_6 = \neg H(a_3, a_4)\ ,\ H'_7 = H_1 \vee H_2\ ,\ H'_8 = \neg(H_3 \vee H_4)\ ,$$
$$H'_9 = \neg(H_5 \vee H_6)\ ,\ H'_{10} = \neg(H_7 \wedge H_8)\ ,\ H'_{11} = H_8 \wedge H_9\ ,\ H'_{12} = H_{10} \vee H_{11}$$
$$\hat{H} = H_{12}.$$



(a) Step 1



(b) Step 2



(c) Step 3

Figure 2: Graphical example of how to push all the 'Not' operations from second or higher layers into the first layer

Removing the not operation in the intermediate layers

$$
\begin{aligned}
&H_1 = H(a_1, a_2)\ ,\ H_2 = H(a_1, a_3)\ ,\ H_3 = H(a_1, a_4)\ ,\ H_4 = H(a_2, a_3)\ ,\\
&H_5 = H(a_2, a_4)\ ,\ H_6 = H(a_3, a_4)\ ,\ H_7 = H_1' \wedge H_2'\ ,\ H_8 = H_3 \vee H_4\ ,\\
&H_9 = H_5 \vee H_6\ ,\ H_{10} = H_7 \wedge H_8\ ,\ H_{11} = H_8' \vee H_9'\ ,\ H_{12} = H_{10}' \wedge H_{11}'\ ,\\
&H_1' = \neg H(a_1, a_2)\ ,\ H_2' = \neg H(a_1, a_3)\ ,\ H_3' = \neg H(a_1, a_4)\ ,\ H_4' = \neg H(a_2, a_3)\ ,\\
&H_5' = \neg H(a_2, a_4)\ ,\ H_6' = \neg H(a_3, a_4)\ ,\ H_7' = H_1 \vee H_2\ ,\ H_8' = H_3' \wedge H_4'\ ,\\
&H_9' = H_5' \wedge H_6'\ ,\ H_{10}' = H_7' \vee H_8'\ ,\ H_{11}' = H_8 \wedge H_9\ ,\ H_{12}' = H_{10} \vee H_{11}\\
&\hat{H} = H_{12}.
\end{aligned}
$$

Removing the unnecessary H terms

$$H_1 = H(a_1, a_2)\ ,\ H_2 = H(a_1, a_3)\ ,\ H_3 = H(a_1, a_4)\ ,\ H_4 = H(a_2, a_3)\ ,$$
$$H_5 = H(a_2, a_4)\ ,\ H_6 = H(a_3, a_4)\ ,\ H_8 = H_3 \vee H_4\ ,\ H_9 = H_5 \vee H_6\ ,$$
$$H_{12} = H'_{10} \wedge H'_{11}\ ,\ H'_3 = \neg H(a_1, a_4)\ ,\ H'_4 = \neg H(a_2, a_3)\ ,\ H'_7 = H_1 \vee H_2\ ,$$
$$H'_8 = H'_3 \wedge H'_4\ ,\ H'_{10} = H'_7 \vee H'_8\ ,\ H'_{11} = H_8 \wedge H_9$$
$$\hat{H} = H_{12}.$$

Note that the Boolean algebras before and after removal of 'Not' operations at second and above layers have Polynomial time complexity, same Polynomial time complexity before removal.

4.3 Expanding recursive Boolean algebra into sum of products and product of sums forms

Using the Boolean algebra in the previous subsection,

$$\hat{H} = \big((H_1 \vee H_2) \vee (H'_3 \wedge H'_4)\big) \wedge \big((H_3 \vee H_4) \wedge (H_5 \vee H_6)\big),$$

we expand it out into sum of products form

$$\begin{aligned}
&\hat{H} = \big(H_1 \vee H_2 \vee (H'_3 \wedge H'_4)\big) \wedge \big((H_3 \wedge H_5) \vee (H_3 \wedge H_6) \vee (H_4 \wedge H_5) \vee (H_4 \wedge H_6)\big) \\
&= (H_1 \wedge H_3 \wedge H_5) \vee (H_1 \wedge H_3 \wedge H_6) \vee (H_1 \wedge H_4 \wedge H_5) \vee (H_1 \wedge H_4 \wedge H_6) \vee (H_2 \wedge H_3 \wedge H_5) \\
&\vee (H_2 \wedge H_3 \wedge H_6) \vee (H_2 \wedge H_4 \wedge H_5) \vee (H_2 \wedge H_4 \wedge H_6) \vee (H'_3 \wedge H'_4 \wedge H_3 \wedge H_5) \\
&\vee (H'_3 \wedge H'_4 \wedge H_3 \wedge H_6) \vee (H'_3 \wedge H'_4 \wedge H_4 \wedge H_5) \vee (H'_3 \wedge H'_4 \wedge H_4 \wedge H_6).
\end{aligned}$$

We can also expand it into product of sums form

$$\begin{aligned}
\hat{H} &= \big((H_1 \vee H_2 \vee H'_3) \wedge (H_1 \vee H_2 \vee H'_4)\big) \wedge (H_3 \vee H_4) \wedge (H_5 \vee H_6) \\
&= (H_1 \vee H_2 \vee H'_3) \wedge (H_1 \vee H_2 \vee H'_4) \wedge (H_3 \vee H_4) \wedge (H_5 \vee H_6).
\end{aligned}$$

So to find the recursive Boolean algebra from sum of products form or product of sums form, we simply reverse the process of expanding the algebra by factoring it into multi-layer Boolean algebra. The multi-layer Boolean algebra is more computational efficient, and in later part of the paper, we will use it to try to find Polynomial computation time Boolean algebra from Non-Deterministic Polynomial computation time Boolean algebra by factoring the Boolean algebra. We will show that Polynomial computation time Boolean algebra cannot be found from Non-Deterministic Polynomial computation time Boolean algebra and NP $\neq$ P. So no matter whether we factor the sum of products form Boolean algebra or the product of sums Boolean algebra, we still can get the original recursive multi-layer Boolean algebra. Later we will express the Boolean algebra to solve the Non-Deterministic Polynomial time Markov Random Field using sum of products form as it is simpler and neater than product of sums form.

4.4 Represent Boolean algebra using addition and multiplication

Boolean algebra can be represented using addition for 'Or' operation and multiplication for 'And' operation. E.g. $(A \vee B) \wedge (C \vee D)$. It can be written as, if $(A + B)(C + D) > 0$, then the Boolean operation give us an output 1, else it will output a 0. This addition and multiplication format is easier to read.

5 A Polynomial time multi-layer Boolean algebra that solves the Markov Random Field Chain problem

5.1 An example of a factorized Boolean algebra that has Polynomial time

Example of a Markov Random Field Chain with each variable a_i taking a value of either 0 or 1. The Markov Random Field can be solved by a 5 layers Boolean algebra. This Boolean algebra infers the value of $\hat{H}(a_6 = 0, a_7 = 0)$. This Boolean algebra has Polynomial complexity. The example of a 7 variables Markov Random Field Chain Boolean algebra is

$$\Big(\big(H(a_1 = 0, a_2 = 0)H(a_2 = 0, a_3 = 0) + H(a_1 = 0, a_2 = 1)H(a_2 = 1, a_3 = 0)\big)$$
$$\big(H(a_3 = 0, a_4 = 0)H(a_4 = 0, a_5 = 0) + H(a_3 = 0, a_4 = 1)H(a_4 = 1, a_5 = 0)\big)$$
$$+ \big(H(a_1 = 0, a_2 = 0)H(a_2 = 0, a_3 = 1) + H(a_1 = 0, a_2 = 1)H(a_2 = 1, a_3 = 1)\big)$$
$$\big(H(a_3 = 1, a_4 = 0)H(a_4 = 0, a_5 = 0) + H(a_3 = 1, a_4 = 1)H(a_4 = 1, a_5 = 0)\big)\Big) H(a_6 = 0, a_7 = 0).$$

This Markov Random Field Chain lacks of potential H terms (or constraints), H(a1,a3),H(a1,a4),H(a1,a5),H(a2,a4),H(a2,a5),H(a3,a5) as shown in figure 3. Because of these missing constraints, this Markov Random Field can be solved efficiently in Polynomial computation time.

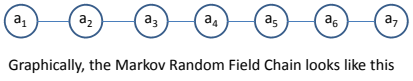


Figure 3: Graphical representation of a Markov Random Field Chain

Figure 4 shows the graphical form of the Polynomial time complexity Boolean algebra that solves the Markov Random Field Chain.

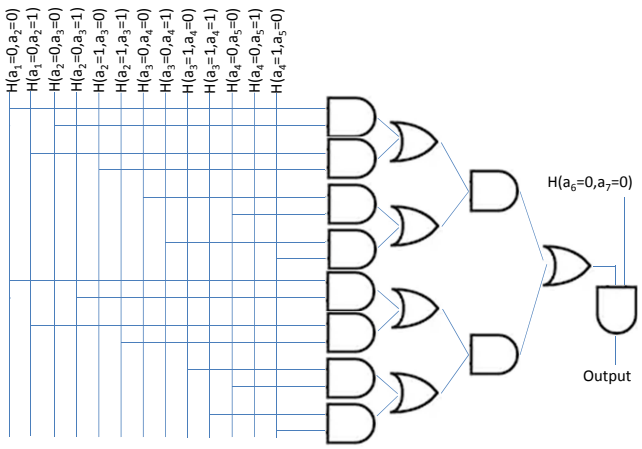


Figure 4: The graphical form of the Polynomial time complexity Boolean algebra that solves the Markov Random Field Chain

5.2 An example of an expanded Boolean algebra that has Non-Deterministic Polynomial time

After expansion, it looks like this

$$
\begin{aligned}
&H(a_1=0,a_2=0)H(a_2=0,a_3=0)H(a_3=0,a_4=0)H(a_4=0,a_5=0)H(a_6=0,a_7=0)\\
&+H(a_1=0,a_2=0)H(a_2=0,a_3=0)H(a_3=0,a_4=1)H(a_4=1,a_5=0)H(a_6=0,a_7=0)\\
&+H(a_1=0,a_2=0)H(a_2=0,a_3=1)H(a_3=1,a_4=0)H(a_4=0,a_5=0)H(a_6=0,a_7=0)\\
&+H(a_1=0,a_2=0)H(a_2=0,a_3=1)H(a_3=1,a_4=1)H(a_4=1,a_5=0)H(a_6=0,a_7=0)\\
&+H(a_1=0,a_2=1)H(a_2=1,a_3=0)H(a_3=0,a_4=0)H(a_4=0,a_5=0)H(a_6=0,a_7=0)\\
&+H(a_1=0,a_2=1)H(a_2=1,a_3=0)H(a_3=0,a_4=1)H(a_4=1,a_5=0)H(a_6=0,a_7=0)\\
&+H(a_1=0,a_2=1)H(a_2=1,a_3=1)H(a_3=1,a_4=0)H(a_4=0,a_5=0)H(a_6=0,a_7=0)\\
&+H(a_1=0,a_2=1)H(a_2=1,a_3=1)H(a_3=1,a_4=1)H(a_4=1,a_5=0)H(a_6=0,a_7=0).
\end{aligned}
$$

This expanded Boolean algebra has Non-Deterministic Polynomial time complexity, with exponential number of additions.

Figure 4 shows the graphical form of the NP time complexity Boolean algebra that solves the Markov Random Field Chain.

To find the Polynomial complexity 5 layers Boolean algebra from the expanded Boolean algebra, we need to factorize the expanded Boolean algebra in the reverse order it is expanded.

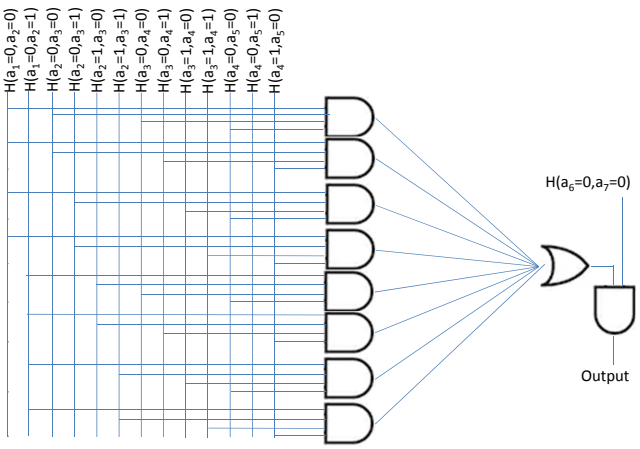


Figure 5: The graphical form of the NP time complexity Boolean algebra that solves the Markov Random Field Chain

6 A six variables Non-Deterministic Polynomial time Markov Random Field cannot be solved by Polynomial time multi-layer Boolean algebra, therefore NP not equal P

In this section, we will prove the lemma below.

Lemma 6.1 *All* 6 *variables Non-Deterministic Polynomial computation time Markov Random Field Boolean algebra cannot be factorized into quadratic form* $(A+B+\dots)(C+D+\dots)$ *or more simplified multi-layer Polynomial computation time Boolean algebra form.*

6.1 Definition of the Markov Random Field

A Boolean algebra that solves a 6 variables Markov Random Field looks like this below. This Boolean algebra infers the value of $\hat{H}(a_5 = 0, a_6 = 0)$,

$$\hat{H}(a_5 = 0, a_6 = 0) = \sum_{a_1,a_2,a_3,a_4,a_5=0,a_6=0} H(a_1, a_2)H(a_1, a_3)H(a_1, a_4)H(a_1, a_5)H(a_1, a_6)H(a_2, a_3)$$

$$H(a_2, a_4)H(a_2, a_5)H(a_2, a_6)H(a_3, a_4)H(a_3, a_5)H(a_3, a_6)H(a_4, a_5)H(a_4, a_6)H(a_5, a_6)$$

where $\sum_{a_5=0,a_6=0}$ means that a_5 and a_6 can only take the value 0. a_5 and a_6 are set to zero because the $\hat{H}(a_5 = 0, a_6 = 0)$ is what we want to infer. If

$\hat{H}(a_5 = 0, a_6 = 0) = 1$ means that $a_5 = a_6 = 0$ is the solution to this Markov Random Field. Else if $\hat{H}(a_5 = 0, a_6 = 0) = 0$ means that $a_5 = a_6 = 0$ is not the solution to this Markov Random Field. Note that this Boolean algebra is in sum of products form.

6.2 How to factorize the terms

For example, some product terms of the Boolean algebra can be factorized into this form

$$H(a_5 = 0, a_6 = 0) \sum_{a_1,a_2,a_3,a_4,a_5=0,a_6=0} H(a_1, a_2)H(a_1, a_3)H(a_1, a_4)H(a_1, a_5)H(a_1, a_6)H(a_2, a_3)$$
$$H(a_2, a_4)H(a_2, a_5)H(a_2, a_6)H(a_3, a_4)H(a_3, a_5)H(a_3, a_6)H(a_4, a_5)H(a_4, a_6)$$

Or

$$H(a_1 = 0, a_2 = 0) \sum_{a_1=0,a_2=0,a_3,a_4,a_5=0,a_6=0} H(a_1, a_3)H(a_1, a_4)H(a_1, a_5)H(a_1, a_6)H(a_2, a_3)H(a_2, a_4)$$
$$H(a_2, a_5)H(a_2, a_6)H(a_3, a_4)H(a_3, a_5)H(a_3, a_6)H(a_4, a_5)H(a_4, a_6)H(a_5, a_6).$$

The grammar to transform a set of boolean algebra terms into a factored form is e.g.

$$\sum_{a_1=0,a_2=0,a_3,a_4,a_5=0,a_6=0} H(a_1 = 0, a_2 = 0)H(a_1, a_3)H(a_1, a_4)H(a_1, a_5)H(a_1, a_6)$$
$$H(a_2, a_3)H(a_2, a_4)H(a_2, a_5)H(a_2, a_6)H(a_3, a_4)$$
$$H(a_3, a_5)H(a_3, a_6)H(a_4, a_5)H(a_4, a_6)H(a_5 = 0, a_6 = 0)$$
$$\Longrightarrow$$
$$H(a_1 = 0, a_2 = 0)H(a_5 = 0, a_6 = 0)$$
$$\sum_{a_1=0,a_2=0,a_3,a_4,a_5=0,a_6=0} H(a_1, a_3)H(a_1, a_4)H(a_1, a_5)H(a_1, a_6)$$
$$H(a_2, a_3)H(a_2, a_4)H(a_2, a_5)H(a_2, a_6)H(a_3, a_4)$$
$$H(a_3, a_5)H(a_3, a_6)H(a_4, a_5)H(a_4, a_6)$$

(1)

6.3 Only one sum of products representation of the Boolean algebra to solve the Markov Random Field

Given the 6 variables Markov Random Field representation of a NP problem,

$$\sum_{a_1,a_2,a_3,a_4,a_5=0,a_6=0} H(a_1,a_2)H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_3)$$
$$H(a_2,a_4)H(a_2,a_5)H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)H(a_5,a_6).$$

The equation above shows the Boolean algebra to solve the Markov Random Field using sum of products representation.

If the product term misses 1 factor, e.g. $H(a_1,a_2)$, then

$$H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_3)H(a_2,a_4)H(a_2,a_5)H(a_2,a_6)$$
$$H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)H(a_5=0,a_6=0)$$

term will recognize $H(a_1,a_2)=0$ and

$$H(a_1,a_3)=H(a_1,a_4)=H(a_1,a_5)=H(a_1,a_6)=H(a_2,a_3)=H(a_2,a_4)=H(a_2,a_5)=H(a_2,a_6)$$
$$=H(a_3,a_4)=H(a_3,a_5)=H(a_3,a_6)=H(a_4,a_5)=H(a_4,a_6)=H(a_5=0,a_6=0)=1$$

as the solution, which is incorrect as the constraint $H(a_1,a_2)$ is not satisfied, $H(a_1,a_2)=0$. If the product term have an additional factor, e.g $H(a_5=1,a_6=1)$, then

$$H(a_1,a_2)H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_3)H(a_2,a_4)H(a_2,a_5)$$
$$H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)H(a_5=0,a_6=0)H(a_5=1,a_6=1),$$

it will miss out the solution

$$H(a_1,a_2)=H(a_1,a_3)=H(a_1,a_4)=H(a_1,a_5)=H(a_1,a_6)=H(a_2,a_3)=H(a_2,a_4)=H(a_2,a_5)$$
$$=H(a_2,a_6)=H(a_3,a_4)=H(a_3,a_5)=H(a_3,a_6)=H(a_4,a_5)=H(a_4,a_6)=H(a_5=0,a_6=0)=1$$

as the solution when $H(a_5=1,a_6=1)=0$.

6.4 Unable to factorize the terms into quadratic form (A+B+...)(C+D+...)

For example for factored term1 and term2,

$$\text{Term1} = H(a_1=0,a_2=0)H(a_5=0,a_6=0)$$
$$\Big(\sum_{a_1=0,a_2=0,a_3,a_4,a_5=0,a_6=0} H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_3)H(a_2,a_4)$$
$$H(a_2,a_5)H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)\Big)$$
$$\text{Term2} = H(a_2=1,a_3=0)H(a_5=0,a_6=0)$$
$$\Big(\sum_{a_1,a_2=1,a_3=0,a_4,a_5=0,a_6=0} H(a_1,a_2)H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_4)$$
$$H(a_2,a_5)H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)\Big),$$

term1 and term2 cannot be factorized into this form $\big(H(a_1=0,a_2=0)H(a_5=0,a_6=0)+H(a_2=1,a_3=0)H(a_5=0,a_6=0)\big)(*)$ because the 2 product factors of term1 and term2 are different with $a_2=0$ for term1 and $a_2=1$ for term2.

$$\Big(\sum_{a_1=0,a_2=0,a_3,a_4,a_5=0,a_6=0} H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_3)H(a_2,a_4)$$
$$H(a_2,a_5)H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)\Big)$$

is not equal to

$$\Big(\sum_{a_1,a_2=1,a_3=0,a_4,a_5=0,a_6=0} H(a_1,a_2)H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_4)$$
$$H(a_2,a_5)H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)\Big).$$

The gammar of not able to factorize the terms into $(A+B+\ldots)(C+D+\ldots)$ is shown below.

$$\begin{aligned}
&H(a_1=0,a_2=0)H(a_5=0,a_6=0)\\
&\Big(\sum_{a_1=0,a_2=0,a_3,a_4,a_5=0,a_6=0} H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_3)H(a_2,a_4)\\
&H(a_2,a_5)H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)\Big)\\
&\wedge\\
&H(a_2=1,a_3=0)H(a_5=0,a_6=0)\\
&\Big(\sum_{a_1,a_2=1,a_3=0,a_4,a_5=0,a_6=0} H(a_1,a_2)H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_4)\\
&H(a_2,a_5)H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)\Big)\\
&\not\Rightarrow\\
&\big(H(a_1=0,a_2=0)H(a_5=0,a_6=0)+H(a_2=1,a_3=0)H(a_5=0,a_6=0)\big)(*)
\end{aligned} \tag{2}$$

6.5 How to combine two factorized terms

For another example of factored term1 and term2

$$\text{Term1} = H(a_1 = 0, a_2 = 0)H(a_5 = 0, a_6 = 0)$$
$$\Big(\sum_{a_1=0,a_2=0,a_3,a_4,a_5=0,a_6=0} H(a_1, a_3)H(a_1, a_4)H(a_1, a_5)H(a_1, a_6)H(a_2, a_3)H(a_2, a_4)$$
$$H(a_2, a_5)H(a_2, a_6)H(a_3, a_4)H(a_3, a_5)H(a_3, a_6)H(a_4, a_5)H(a_4, a_6)\Big)$$
$$\text{Term2} = H(a_2 = 0, a_3 = 0)H(a_5 = 0, a_6 = 0)$$
$$\Big(\sum_{a_1,a_2=0,a_3=0,a_4,a_5=0,a_6=0} H(a_1, a_2)H(a_1, a_3)H(a_1, a_4)H(a_1, a_5)H(a_1, a_6)H(a_2, a_4)$$
$$H(a_2, a_5)H(a_2, a_6)H(a_3, a_4)H(a_3, a_5)H(a_3, a_6)H(a_4, a_5)H(a_4, a_6)\Big)$$

can be combined into this form

$$H(a_1 = 0, a_2 = 0)H(a_2 = 0, a_3 = 0)H(a_1 = 0, a_3 = 0)H(a_5 = 0, a_6 = 0)$$
$$\Big(\sum_{a_1=0,a_2=0,a_3=0,a_4,a_5=0,a_6=0} H(a_1, a_4)H(a_1, a_5)H(a_1, a_6)H(a_2, a_4)H(a_2, a_5)$$
$$H(a_2, a_6)H(a_3, a_4)H(a_3, a_5)H(a_3, a_6)H(a_4, a_5)H(a_4, a_6)\Big)$$
$$+ H(a_2 = 0, a_3 = 0)H(a_5 = 0, a_6 = 0)$$
$$\sum_{a_1>1,a_2=0,a_3=0,a_4,a_5=0,a_6=0} H(a_1, a_2)H(a_1, a_3)H(a_1, a_4)H(a_1, a_5)H(a_1, a_6)$$
$$H(a_2, a_4)H(a_2, a_5)H(a_2, a_6)H(a_3, a_4)H(a_3, a_5)H(a_3, a_6)H(a_4, a_5)H(a_4, a_6)$$
$$+ H(a_1 = 0, a_2 = 0)H(a_5 = 0, a_6 = 0)$$
$$\sum_{a_1=0,a_2=0,a_3>1,a_4,a_5=0,a_6=0} H(a_1, a_3)H(a_1, a_4)H(a_1, a_5)H(a_1, a_6)H(a_2, a_3)$$
$$H(a_2, a_4)H(a_2, a_5)H(a_2, a_6)H(a_3, a_4)H(a_3, a_5)H(a_3, a_6)H(a_4, a_5)H(a_4, a_6).$$

Note that $\sum_{a_1>1}$ means that for all values of a_1 except $a_1 = 0$. Still it cannot be factorized into $(A + B + \ldots)(C + D + \ldots)$ form (quadratic form). It is in $A(B + C + \ldots) + D$ form.

The grammar to combine factorized terms is shown below.

$$
\begin{aligned}
&H(a_1=0,a_2=0)H(a_5=0,a_6=0)\\
&\Big(\sum_{a_1=0,a_2=0,a_3,a_4,a_5=0,a_6=0} H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_3)H(a_2,a_4)\\
&H(a_2,a_5)H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)\Big)\\
&\wedge\\
&H(a_2=0,a_3=0)H(a_5=0,a_6=0)\\
&\Big(\sum_{a_1,a_2=0,a_3=0,a_4,a_5=0,a_6=0} H(a_1,a_2)H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_4)\\
&H(a_2,a_5)H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)\Big)\\
&\Longrightarrow\\
&H(a_1=0,a_2=0)H(a_2=0,a_3=0)H(a_1=0,a_3=0)H(a_5=0,a_6=0)\\
&\Big(\sum_{a_1=0,a_2=0,a_3=0,a_4,a_5=0,a_6=0} H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_4)H(a_2,a_5)\\
&H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)\Big)\\
&+H(a_2=0,a_3=0)H(a_5=0,a_6=0)\\
&\sum_{a_1>1,a_2=0,a_3=0,a_4,a_5=0,a_6=0} H(a_1,a_2)H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)\\
&H(a_2,a_4)H(a_2,a_5)H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)\\
&+H(a_1=0,a_2=0)H(a_5=0,a_6=0)\\
&\sum_{a_1=0,a_2=0,a_3>1,a_4,a_5=0,a_6=0} H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_3)\\
&H(a_2,a_4)H(a_2,a_5)H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)
\end{aligned}
\tag{3}
$$

6.6 How to expand a term by multiply (u+u') to a term

$$\text{Term1} = H(a_1 = 0, a_2 = 0)H(a_5 = 0, a_6 = 0)$$
$$\Big(\sum_{a_1=0,a_2=0,a_3,a_4,a_5=0,a_6=0} H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_3)H(a_2,a_4)$$
$$H(a_2,a_5)H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)\Big)$$
$$\text{Term2} = H(a_2 = 1, a_3 = 0)H(a_5 = 0, a_6 = 0)$$
$$\Big(\sum_{a_1,a_2=1,a_3=0,a_4,a_5=0,a_6=0} H(a_1,a_2)H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_4)$$
$$H(a_2,a_5)H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)\Big)$$

For the term1, even if you expand the product term, e.g.

$$\Big(\sum_{a_1=0,a_2=0,a_3,a_4,a_5=0,a_6=0} H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_3)H(a_2,a_4)$$
$$H(a_2,a_5)H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)\Big)\big(H'(a_1 = 1, a_2) + H(a_1 = 1, a_2)\big)$$
$$= H'(a_1 = 1, a_2) \sum_{a_1=0,a_2=0,a_3,a_4,a_5=0,a_6=0} H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_3)H(a_2,a_4)$$
$$H(a_2,a_5)H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)$$
$$+ H(a_1 = 1, a_2) \sum_{a_1=0,a_2=0,a_3,a_4,a_5=0,a_6=0} H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_3)H(a_2,a_4)$$
$$H(a_2,a_5)H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6),$$

the 2 terms term1 and term2 still cannot be factored into $(A+B+\ldots)(C+D+\ldots)$ format (quadratic format) because the a_2 values are still different (term1 $a_2 = 0$ and term2 $a_2 = 1$). Note that $H'(a_i, a_j) = 1 - H(a_i, a_j)$, which means in Boolean algebra it is a 'Not' operation.

The grammar to expand the product term is e.g.

$$\sum_{a_1=0,a_2=0,a_3,a_4,a_5=0,a_6=0} H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_3)H(a_2,a_4)$$
$$H(a_2,a_5)H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)$$
$$\Longrightarrow$$
$$\Big(\sum_{a_1=0,a_2=0,a_3,a_4,a_5=0,a_6=0} H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_3)H(a_2,a_4)$$
$$H(a_2,a_5)H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)\Big)$$
$$\big(H'(a_1=1,a_2)+H(a_1=1,a_2)\big) \tag{4}$$

6.7 How to expand a factorized term

$$\text{Term1} = H(a_1=0,a_2=0)H(a_5=0,a_6=0)$$
$$\Big(\sum_{a_1=0,a_2=0,a_3,a_4,a_5=0,a_6=0} H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_3)H(a_2,a_4)$$
$$H(a_2,a_5)H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)\Big)$$
$$\text{Term2} = H(a_2=1,a_3=0)H(a_5=0,a_6=0)$$
$$\Big(\sum_{a_1,a_2=1,a_3=0,a_4,a_5=0,a_6=0} H(a_1,a_2)H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_4)$$
$$H(a_2,a_5)H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)\Big)$$

Even if you add the new term below using

$$\text{Term3} = H(a_1=0,a_2=0)H(a_5=0,a_6=0)H'(a_1=1,a_2=0)\big(H(a_1=1,a_2=0)$$
$$\sum_{a_1=0,a_2=0,a_3,a_4,a_5=0,a_6=0} H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_3)H(a_2,a_4)$$
$$H(a_2,a_5)H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)$$
$$+\sum_{a_1=0,a_2=0,a_3,a_4,a_5=0,a_6=0} H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_3)H(a_2,a_4)$$
$$H(a_2,a_5)H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)\big),$$

term3 cannot be combined with term2 into the form $(A+B+\ldots)(C+D+\ldots)$ because a_2 values is 0 for term3 and 1 for term2.

Another grammar to expand a factorized term is e.g.

$$
\begin{aligned}
&H'(a_1=1,a_2=0)H(a_5=0,a_6=0)H(a_1=0,a_2=0)\sum_{a_1=0,a_2=0,a_3,a_4,a_5=0,a_6=0}\\
&H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_3)H(a_2,a_4)\\
&H(a_2,a_5)H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)\\
&\Longrightarrow\\
&H'(a_1=1,a_2=0)H(a_5=0,a_6=0)H(a_1=0,a_2=0)\big(H(a_1=1,a_2=0)\\
&\sum_{a_1=0,a_2=0,a_3,a_4,a_5=0,a_6=0}H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_3)H(a_2,a_4)\\
&H(a_2,a_5)H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)+\\
&\sum_{a_1=0,a_2=0,a_3,a_4,a_5=0,a_6=0}H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_3)H(a_2,a_4)\\
&H(a_2,a_5)H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)\big)
\end{aligned}
\tag{5}
$$

6.8 Summary of this section

Therefore the six variables Boolean algebra

$$
\begin{aligned}
&\sum_{a_1,a_2,a_3,a_4,a_5=0,a_6=0}H(a_1,a_2)H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_3)\\
&H(a_2,a_4)H(a_2,a_5)H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)H(a_5,a_6)
\end{aligned}
$$

can never be factorized into $(A+B+\ldots)(C+D+\ldots)$ 2-layer Boolean algebra (quadratic form), so it can also not able to factorized into 3 or higher layers Boolean algebra. It is factorized using grammar rules from equations 1, 2, 3, 4 and 5.

The Boolean algebra can never be executed in Polynomial time as it cannot be factorized into multi-layer Polynomial time Boolean algebra. Therefore Non-Deterministic Polynomial time algorithm cannot be converted to Polynomial time algorithm, and NP $\neq$ P. This six variables case can be easily extended to 7 or higher number of variables.

6.9 Example of how to factorize the Boolean algebra

Example of a factorization of the 6 variables Boolean algebra is

$$\hat{H}(a_5 = 0, a_6 = 0) = \sum_{a_1,a_2,a_3,a_4,a_5=0,a_6=0} H(a_1,a_2)H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_3)$$
$$H(a_2,a_4)H(a_2,a_5)H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)H(a_5,a_6)$$
$$= \sum_{a_1,a_2,a_5=0,a_6=0} H(a_1,a_2)H(a_5,a_6)$$
$$\sum_{a_3,a_4} H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_3)H(a_2,a_4)$$
$$H(a_2,a_5)H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)$$
$$+ \sum_{a_1,a_3,a_5=0,a_6=0} H(a_1,a_3)H(a_5,a_6)$$
$$\sum_{a_2,a_4} H(a_1,a_2)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_3)H(a_2,a_4)$$
$$H(a_2,a_5)H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)$$
$$+ \sum_{a_1,a_4,a_5=0,a_6=0} H(a_1,a_4)H(a_5,a_6)$$
$$\sum_{a_2,a_3} H(a_1,a_2)H(a_1,a_3)H(a_1,a_5)H(a_1,a_6)H(a_2,a_3)H(a_2,a_4)$$
$$H(a_2,a_5)H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)$$
$$+ \sum_{a_1,a_5=0,a_6=0} H(a_1,a_5)H(a_5,a_6)$$
$$\sum_{a_2,a_3,a_4} H(a_1,a_2)H(a_1,a_3)H(a_1,a_4)H(a_1,a_6)H(a_2,a_3)H(a_2,a_4)$$
$$H(a_2,a_5)H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)$$

$$+ \sum_{a_1,a_5=0,a_6=0} H(a_1,a_6)H(a_5,a_6)$$
$$\sum_{a_2,a_3,a_4} H(a_1,a_2)H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_2,a_3)H(a_2,a_4)$$
$$H(a_2,a_5)H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)$$
$$+ \sum_{a_2,a_3,a_5=0,a_6=0} H(a_2,a_3)H(a_5,a_6)$$
$$\sum_{a_1,a_4} H(a_1,a_2)H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_4)$$
$$H(a_2,a_5)H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)$$
$$+ \sum_{a_2,a_4,a_5=0,a_6=0} H(a_2,a_4)H(a_5,a_6)$$
$$\sum_{a_1,a_3} H(a_1,a_2)H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_3)$$
$$H(a_2,a_5)H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)$$
$$+ \sum_{a_2,a_5=0,a_6=0} H(a_2,a_5)H(a_5,a_6)$$
$$\sum_{a_1,a_3,a_4} H(a_1,a_2)H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_3)$$
$$H(a_2,a_4)H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)$$

$$+ \sum_{a_2,a_5=0,a_6=0} H(a_2,a_6)H(a_5,a_6)$$
$$\sum_{a_1,a_3,a_4} H(a_1,a_2)H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_3)$$
$$H(a_2,a_4)H(a_2,a_5)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)$$
$$+ \sum_{a_3,a_4,a_5=0,a_6=0} H(a_3,a_4)H(a_5,a_6)$$
$$\sum_{a_1,a_2} H(a_1,a_2)H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_3)$$
$$H(a_2,a_4)H(a_2,a_5)H(a_2,a_6)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)$$
$$+ \sum_{a_3,a_5=0,a_6=0} H(a_3,a_5)H(a_5,a_6)$$
$$\sum_{a_1,a_2,a_4} H(a_1,a_2)H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_3)$$
$$H(a_2,a_4)H(a_2,a_5)H(a_2,a_6)H(a_3,a_4)H(a_3,a_6)H(a_4,a_5)H(a_4,a_6)$$
$$+ \sum_{a_3,a_5=0,a_6=0} H(a_3,a_6)H(a_5,a_6)$$
$$\sum_{a_1,a_2,a_4} H(a_1,a_2)H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_3)$$
$$H(a_2,a_4)H(a_2,a_5)H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_4,a_5)H(a_4,a_6)$$

$$+ \sum_{a_4,a_5=0,a_6=0} H(a_4,a_5)H(a_5,a_6)$$
$$\sum_{a_1,a_2,a_3} H(a_1,a_2)H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_3)$$
$$H(a_2,a_4)H(a_2,a_5)H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_6)$$
$$+ \sum_{a_4,a_5=0,a_6=0} H(a_4,a_6)H(a_5,a_6)$$
$$\sum_{a_1,a_2,a_3} H(a_1,a_2)H(a_1,a_3)H(a_1,a_4)H(a_1,a_5)H(a_1,a_6)H(a_2,a_3)$$
$$H(a_2,a_4)H(a_2,a_5)H(a_2,a_6)H(a_3,a_4)H(a_3,a_5)H(a_3,a_6)H(a_4,a_5).$$

This Boolean algebra is too complex to show in graphical form.

7 Generalized to higher number of variables Markov Random Field

In this section, we want to prove the lemma below.

Lemma 7.1 *All Non-Deterministic Polynomial computation time Markov Random Field Boolean algebra cannot be factorized into quadratic form* $(A + B + \dots)(C + D + \dots)$ *or more simplified multi-layer Polynomial computation time Boolean algebra form.*

7.1 A generalized Markov Random Field

The equation

$$\hat{H}(a_k = v_k, a_l = v_l) = H(a_k = v_k, a_l = v_l) \sum_{v_1, v_2, \dots, v_n \setminus v_k, v_l} \prod_{(i,j)\setminus(k,l)} H(a_i = v_i, a_j = v_j)$$

shows the Boolean algebra to determine whether $a_k = v_k$ and $a_l = v_l$ is the solution of the Markov Random Field. When $\hat{H}(a_k = v_k, a_l = v_l) = 1$ means that $a_k = v_k$ and $a_l = v_l$ is the solution, else when $\hat{H}(a_k = v_k, a_l = v_l) = 0$ means that $a_k = v_k$ and $a_l = v_l$ is not the solution.

7.2 How to factorize the terms

All terms can be factorized using the formula below. Factorized term1 is

$$H(a_{i_1} = v_{i_1}, a_{i_2} = v_{i_2}) H(a_{i_3} = v_{i_3}, a_{i_4} = v_{i_4}) \dots H(a_{i_{N_1-1}} = v_{i_{N_1-1}}, a_{i_{N_1}} = v_{i_{N_1}})$$
$$\sum_{a_{i_1} = v_{i_1}, \dots, a_{i_{N_1}} = v_{i_{N_1}}, a_{j_1^{(1)}}, \dots, a_{j_{M_1}^{(1)}}} \prod_{i,j \in \{i_1, \dots, i_{N_1}, j_1^{(1)}, \dots, j_{M_1}^{(1)}\}, i \neq j} H(a_i, a_j)$$

for all l and m in a_{i_l} and $a_{j_m^{(1)}}$, $i_l \neq j_m^{(1)}$. Given another factorized term2

$$H(a_{k_1} = v_{k_1}, a_{k_2} = v_{k_2}) H(a_{k_3} = v_{k_3}, a_{k_4} = v_{k_4}) \dots H(a_{k_{N_2-1}} = v_{k_{N_2-1}}, a_{k_{N_2}} = v_{k_{N_2}})$$
$$\sum_{a_{k_1} = v_{k_1}, \dots, a_{k_{N_2}} = v_{k_{N_2}}, a_{j_1^{(2)}}, \dots, a_{j_{M_2}^{(2)}}} \prod_{i,j \in \{k_1, \dots, k_{N_2}, j_1^{(2)}, \dots, j_{M_2}^{(2)}\}, i \neq j} H(a_i, a_j)$$

for all l and m in a_{k_l} and $a_{j_m^{(2)}}$, $k_l \neq j_m^{(2)}$. This two factorized term cannot be further factorized into $(A + B \dots)(C + D \dots)$ format (quadratic format) as long as there is a variable a_{i_l}, a_{k_m} with different values $v_{i_l} \neq v_{k_m}$ where $i_l = k_m$. Note that $j_m^{(1)}$ and $j_m^{(2)}$ are not the same index. $a_i, a_j \in \{a_{i_1}, \dots, a_{i_{N_1}}, a_{j_1^{(2)}}, \dots, a_{j_{M_1}^{(1)}}\}$ means that a_i and a_j can be one of the variables in the set $\{*\}$.

The grammar to factorize a set of terms is shown below.

$$\sum_{a_{i_1}=v_{i_1},\ldots,a_{i_{N_1}}=v_{i_{N_1}},a_{j_1^{(1)}},\ldots,a_{j_{M_1}^{(1)}}} H(a_{i_1}=v_{i_1},a_{i_2}=v_{i_2})H(a_{i_3}=v_{i_3},a_{i_4}=v_{i_4})\ldots H(a_{i_{N_1-1}}=v_{i_{N_1-1}},a_{i_{N_1}}=v_{i_{N_1}}) \prod_{i,j\in\{i_1,\ldots,i_{N_1},j_1^{(1)},\ldots,j_{M_1}^{(1)}\},i\neq j} H(a_i,a_j)$$

$$\Longrightarrow$$

$$H(a_{i_1}=v_{i_1},a_{i_2}=v_{i_2})H(a_{i_3}=v_{i_3},a_{i_4}=v_{i_4})\ldots H(a_{i_{N_1-1}}=v_{i_{N_1-1}},a_{i_{N_1}}=v_{i_{N_1}}) \sum_{a_{i_1}=v_{i_1},\ldots,a_{i_{N_1}}=v_{i_{N_1}},a_{j_1^{(1)}},\ldots,a_{j_{M_1}^{(1)}}} \prod_{i,j\in\{i_1,\ldots,i_{N_1},j_1^{(1)},\ldots,j_{M_1}^{(1)}\},i\neq j} H(a_i,a_j)$$

(6)

The grammar of not able to factorize the terms into $(A+B+\ldots)(B+C+\ldots)$

is shown below.

$$
\begin{aligned}
&H(a_{i_1}=v_{i_1},a_{i_2}=v_{i_2})H(a_{i_3}=v_{i_3},a_{i_4}=v_{i_4})\dots H(a_{i_{N_1-1}}=v_{i_{N_1-1}},a_{i_{N_1}}=v_{i_{N_1}})\\
&\sum_{a_{i_1}=v_{i_1},\dots,a_{i_{N_1}}=v_{i_{N_1}},a_{j_1^{(1)}},\dots,a_{j_{M_1}^{(1)}}}\ \prod_{i,j\in\{i_1,\dots,i_{N_1},j_1^{(1)},\dots,j_{M_1}^{(1)}\},i\neq j} H(a_i,a_j)\\
&\wedge\\
&H(a_{k_1}=v_{k_1},a_{k_2}=v_{k_2})H(a_{k_3}=v_{k_3},a_{k_4}=v_{k_4})\dots H(a_{k_{N_2-1}}=v_{k_{N_2-1}},a_{k_{N_2}}=v_{k_{N_2}})\\
&\sum_{a_{k_1}=v_{k_1},\dots,a_{k_{N_2}}=v_{k_{N_2}},a_{j_1^{(2)}},\dots,a_{j_{M_2}^{(2)}}}\ \prod_{i,j\in\{k_1,\dots,k_{N_2},j_1^{(2)},\dots,j_{M_2}^{(2)}\},i\neq j} H(a_i,a_j)\\
&\nRightarrow\\
&\big(H(a_{i_1}=v_{i_1},a_{i_2}=v_{i_2})H(a_{i_3}=v_{i_3},a_{i_4}=v_{i_4})\dots H(a_{i_{N_1-1}}=v_{i_{N_1-1}},a_{i_{N_1}}=v_{i_{N_1}})\\
&+\\
&H(a_{k_1}=v_{k_1},a_{k_2}=v_{k_2})H(a_{k_3}=v_{k_3},a_{k_4}=v_{k_4})\dots H(a_{k_{N_2-1}}=v_{k_{N_2-1}},a_{k_{N_2}}=v_{k_{N_2}})\big)\\
&(*)
\end{aligned}
\tag{7}
$$

7.3 How to combine factorized terms

If there exists pairs of variables $a_{i_l}=v_{i_l}$, $a_{k_m}=v_{k_m}$with $i_l=k_m$, and $v_{i_l}=v_{k_m}$, and it does not exists pairs of variables $a_{i_l}=v_{i_l}$,$a_{k_m}=v_{k_m}$with $i_l=k_m$, and $v_{i_l}\neq v_{k_m}$, then the two factorized term can only be factorized into $A(B+C+\dots)+D$ format. To be specific, let us write the condition needed to factorize the two terms (we introduce T new variables a_{s_t}),

$$\forall_{s_t,i_l,k_m} i_l=k_m=s_t\rightarrow v_{i_l}=v_{k_m}.$$

If the above condition is True then the two factorized term can be factorized into

$$\Big(\prod_{i,j\in\{s_1,s_2,\ldots,s_T\},i\neq j} H(a_i=v_i,a_j=v_j)\Big)$$
$$\sum_{a_{s_1}=v_{s_1},\ldots,a_{s_T}=v_{s_T},a_{j_1^{(3)}},\ldots,a_{j_M^{(3)}}} \prod_{i,j\in\{s_1,\ldots,s_T,j_1^{(3)},\ldots,j_M^{(3)}\},i\neq j} H(a_i,a_j)$$

+ additional terms.

Note that $s_t \in (\{i_1,i_2,\ldots,i_{N_1}\}\cup\{k_1,k_2,\ldots,k_{N_2}\})$ and $\{j_1^{(3)},j_2^{(3)},\ldots,j_{M_3}^{(3)}\} \in (\{j_1^{(1)},j_2^{(1)},\ldots,j_{M_1}^{(1)}\}\cap\{j_1^{(2)},j_2^{(2)},\ldots,j_{M_2}^{(2)}\})$. $s_t \in \{s_1,s_2,\ldots,s_T\}$, $i_l \in \{i_1,i_2,\ldots,i_{N_1}\}$ and $k_m \in \{k_1,k_2,\ldots,i_{N_2}\}$.

After combining the factorized terms, it is still unable to factorize the terms into $(A+B+\ldots)(C+D+\ldots)$ form.

The grammar to transform a set of Boolean algebra terms into a factored

form is

$$
\begin{aligned}
&H(a_{i_1}=v_{i_1},a_{i_2}=v_{i_2})H(a_{i_3}=v_{i_3},a_{i_4}=v_{i_4})\dots H(a_{i_{N_1-1}}=v_{i_{N_1-1}},a_{i_{N_1}}=v_{i_{N_1}})\\
&\sum_{a_{i_1}=v_{i_1},\dots,a_{i_{N_1}}=v_{i_{N_1}},a_{j_1^{(1)}},\dots,a_{j_{M_1}^{(1)}}}\ \prod_{i,j\in\{i_1,\dots,i_{N_1},j_1^{(1)},\dots,j_{M_1}^{(1)}\},i\neq j} H(a_i,a_j)\\
&\wedge\\
&H(a_{k_1}=v_{k_1},a_{k_2}=v_{k_2})H(a_{k_3}=v_{k_3},a_{k_4}=v_{k_4})\dots H(a_{k_{N_2-1}}=v_{k_{N_2-1}},a_{k_{N_2}}=v_{k_{N_2}})\\
&\sum_{a_{k_1}=v_{k_1},\dots,a_{k_{N_2}}=v_{k_{N_2}},a_{j_1^{(2)}},\dots,a_{j_{M_2}^{(2)}}}\ \prod_{i,j\in\{k_1,\dots,k_{N_2},j_1^{(2)},\dots,j_{M_2}^{(2)}\},i\neq j} H(a_i,a_j)\\
&\implies\\
&\Big(\prod_{i,j\in\{s_1,s_2,\dots,s_T\},i\neq j} H(a_i=v_i,a_j=v_j)\Big)\\
&\sum_{a_{s_1}=v_{s_1},\dots,a_{s_T}=v_{s_T},a_{j_1^{(3)}},\dots,a_{j_{M_3}^{(3)}}}\ \prod_{i,j\in\{s_1,\dots,s_T,j_1^{(3)},\dots,j_{M_3}^{(3)}\},i\neq j} H(a_i,a_j)\\
&+\text{additional terms}\\
&\text{where}\\
&\forall_{s_t,i_l,k_m} i_l=k_m=s_t\rightarrow v_{i_l}=v_{k_m}\\
&s_t\in\{s_1,s_2,\dots,s_T\}=(\{i_1,i_2,\dots,i_{N_1}\}\cup\{k_1,k_2,\dots,k_{N_2}\})\\
&\{j_1^{(3)},j_2^{(3)},\dots,j_{M_3}^{(3)}\}\in(\{j_1^{(1)},j_2^{(1)},\dots,j_{M_1}^{(1)}\}\cap\{j_1^{(2)},j_2^{(2)},\dots,j_{M_2}^{(2)}\})\\
&s_t\in\{s_1,s_2,\dots,s_T\},\ i_l\in\{i_1,i_2,\dots,i_{N_1}\}\text{ and }k_m\in\{k_1,k_2,\dots,i_{N_2}\}
\end{aligned}
\tag{8}
$$

7.4 How to expand a term by multiply (u+u') to a term

Every term can be expanded by multiplying $u + u'$ term,

$$H(a_{i_1}, a_{i_2})H(a_{i_3}, a_{i_4}) \dots H(a_{i_{N-1}}, a_{i_N})(u + u').$$

$u + u'$ can be e.g. $H(a_1 = 0, a_2) + H'(a_1 = 0, a_2)$. $H'() = 1 - H()$, which is a 'Not' operation. Since $u + u'$ is always equal to 1, it can be multiplied to a term without changing the meaning. Using this expansion, we can add complement terms. But after expanding factorized term1 and term2, we still cannot further factorized term1 and term2 into $(A + B + \dots)(C + D + \dots)$ form (quadratic form).

The grammar to expand the product term is

$$\boxed{\begin{array}{l} H(a_{i_1}, a_{i_2})H(a_{i_3}, a_{i_4}) \dots H(a_{i_{N-1}}, a_{i_N}) \\ \quad \Longrightarrow \\ H(a_{i_1}, a_{i_2})H(a_{i_3}, a_{i_4}) \dots H(a_{i_{N-1}}, a_{i_N})(u + u'). \end{array}} \tag{9}$$

7.5 How to expand a factorized term

Another way of expanding the factorized term is

$$\begin{array}{l} H(a_{i_1}, a_{i_2})H(a_{i_3}, a_{i_4}) \dots H(a_{i_{N-1}}, a_{i_N})u' \\ \big(v \times u + H(a_{j_1}, a_{j_2})H(a_{j_3}, a_{j_4}) \dots H(a_{j_{M-1}}, a_{j_M})\big). \end{array}$$

u, u' can be e.g. $H(a_1 = 0, a_2)$, $H'(a_1 = 0, a_2)$. Using this expansion, we can add complement terms. v can be any product sum of $H()$ potentials as $u \times u'$ will always be False or 0. But after expanding factorized term1 and term2, we still cannot further factorized term1 and term2 into $(A + B + \dots)(C + D + \dots)$ form (quadratic form).

Another grammar to expand a factorized term is

$$\boxed{\begin{array}{l} H(a_{i_1}, a_{i_2})H(a_{i_3}, a_{i_4}) \dots H(a_{i_{N-1}}, a_{i_N})u' \\ H(a_{j_1}, a_{j_2})H(a_{j_3}, a_{j_4}) \dots H(a_{j_{M-1}}, a_{j_M}) \\ \quad \Longrightarrow \\ H(a_{i_1}, a_{i_2})H(a_{i_3}, a_{i_4}) \dots H(a_{i_{N-1}}, a_{i_N})u' \\ \big(v \times u + (a_{j_1}, a_{j_2})H(a_{j_3}, a_{j_4}) \dots H(a_{j_{M-1}}, a_{j_M})\big). \end{array}} \tag{10}$$

7.6 An analogy of the Boolean algebra factorization

An analogy to explain whether a Boolean algebra can or cannot be factorized into quadratic form $(A + B + \dots)(C + D + \dots)$ is the example below. The equation below

$$\begin{array}{r} AC + AD + BC + BD \\ = (A + B)(C + D) \end{array}$$

can be factorized into quadratic form $(A+B)(C+D)$. Whereas the equation below

$$\begin{aligned} & AC+AD+BE+BF \\ = & A(C+D)+B(E+F) \end{aligned}$$

cannot be factorized into quadratic form e.g. $(A+B)(C+D)$.

7.7 Summary of this section

Since using grammar rules in equations 6, 7, 8, 9 and 10 cannot factorized the Boolean algebra of a Non-Deterministic Polynomial time Markov Random Field to quadratic form $(A+B+\ldots)(C+D+\ldots)$, Non-Deterministic Polynomial time problem can never be converted to Polynomial time problem, therefore NP $\neq$ P.

8 2sat problem in Markov Random Field representation

8.1 2sat problem definition

A 2sat problem can be defined as

$$\bigwedge_{i,j} \left((\neg A_{i,j} \vee a_i \vee a_j) \wedge (\neg A_{i',j} \vee \neg a_i \vee a_j) \wedge (\neg A_{i,j'} \vee a_i \vee \neg a_j) \wedge (\neg A_{i',j'} \vee \neg a_i \vee \neg a_j) \right)$$

where $A_{i,j}$, $A_{i',j}$, $A_{i,j'}$ and $A_{i',j'}$ determine whether the terms exist or not. E.g. if $A_{i,j}=0$ (which means false), the term $(a_i \vee a_j)$ does not exist. Else if $A_{i,j}=1$ means that the term $(a_i \vee a_j)$ exists. In our paper, True or False can also be represented as 1 or 0. An example a 2sat problem instance is

$$(\neg a_1 \vee a_2) \wedge (\neg a_2 \vee \neg a_3) \wedge (a_3 \vee a_4) \wedge (\neg a_4 \vee a_5)$$

can be represented with

$$A_{1',2}=1, A_{2',3'}=1, A_{3,4}=1, A_{4',5}=1$$

and other $A_{i,j}=0$ for $(i,j) \notin \{(1',2),(2',3'),(3,4),(4',5)\}$.

8.2 An algorithm to solve 2sat problem

2sat problem can be solved using the algorithm 1 below. This is the standard algorithm to solve it.

Algorithm 1: 2sat problem algorithm

Result: Return a True or False whether a 2sat problem has solution

```
for a_i in {a_1, a_2, ..., a_n} do
    for a_i = True or False do
        Set values of all other variables to unknown (neither True or
         False) except a_i;
        Then using the 2sat clauses, try to infer other variables values;
        if there is a contradiction, e.g. variable a_1 is both True and
         False then
            The 2sat has no solution, return False;
    end
end
The 2sat has a solution, return True;
```

8.3 How to solve a four variables 2sat problem

For a four variables 2 values Markov Random Field problem, it is the same as a 2sat problem. The 2sat problem can be represented as

$$\sum_{a_1,a_2,a_3=0,a_4=0} H(a_1,a_2)H(a_1,a_3)H(a_1,a_4)H(a_2,a_3)H(a_2,a_4)H(a_3,a_4),$$

where a_1, a_2, a_3, a_4 can only take values of either 0 or 1. It can be simplified to

$$\begin{aligned}
&\sum_{a_1,a_2,a_3=0,a_4=0} H(a_1,a_2)H(a_2,a_3)H(a_3,a_4)H(a_1,a_4) \\
&+ H(a_1,a_3)H(a_3,a_4)H(a_2,a_4)H(a_1,a_2) \\
&+ H(a_1,a_3)H(a_2,a_3)H(a_2,a_4)H(a_1,a_4) \\
&+ H(a_1,a_2)H(a_2,a_3)H(a_1,a_3) \\
&+ H(a_2,a_3)H(a_3,a_4)H(a_2,a_4) \\
&+ H(a_3,a_4)H(a_1,a_4)H(a_1,a_3) \\
&+ H(a_1,a_2)H(a_2,a_4)H(a_1,a_4)
\end{aligned}$$

where the above equation represents all loop constraints.

Loop constraints can be represented in graphical form of Markov Random Field.

Figure 6 shows a graphical representation of all possible loops that can be extracted for the Markov Random Field. A 2sat problem can be simplified into solving a set of Markov Random Field loops.

A product term can be solved using dynamic programming

$$\begin{aligned}
&\sum_{a_1,a_2,a_3,a_4} H(a_1,a_2)H(a_2,a_3)H(a_3,a_4)H(a_1,a_4) \\
&= \sum_{a_1,a_2} H(a_1,a_2) \sum_{a_3} H(a_2,a_3)H(a_1,a_3) \sum_{a_4} H(a_3,a_4)H(a_1,a_4).
\end{aligned}$$

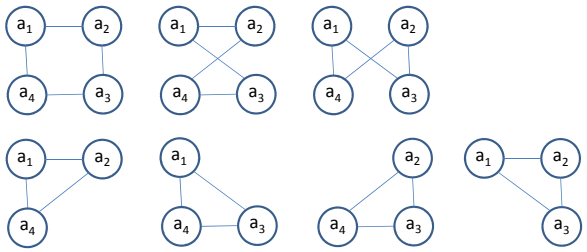


Figure 6: Graphical representation of all possible loops in the Markov Random Field

We ignored $a_3 = a_4 = 0$ in this example to explain the idea better. $a_3 = a_4 = 0$ can be added without any problem. This can be further written into a dynamic programming multi-layer Boolean algebra

$$H^0(a_i, a_j) = H(a_i, a_j)$$
$$H^l(a_i, a_j) = H^{l-1}(a_i, a_j) \sum_{a_k} H^{l-1}(a_i, a_k) H^{l-1}(a_j, a_k)$$

where $1 <= l <= 4$.

Note that $H^0()$, $H^l()$ and $H^{l-1}()$ are the indexes of H (does not mean H is raised to the power of 0, l or $l-1$).

8.4 How to solve a five or more variables 2sat problem

The generalized loop constraint for 5 or more variables is

$$H(a_{i_1}, a_{i_2}) H(a_{i_2}, a_{i_3}) \dots H(a_{i_{N-1}}, a_{i_N}) H(a_{i_N}, a_{i_1})$$

or some cases

$$H(a_{i_1}, a_{i_2}) H(a_{i_2}, a_{i_N - 1}) H(a_{i_{N-1}}, a_{i_N}) H(a_{i_N}, a_{i_1}),$$
$$H(a_{i_1}, a_{i_{N-1}}) H(a_{i_{N-1}}, a_{i_N}) H(a_{i_N}, a_{i_1}).$$

Each loop constraint can be solved by dynamic programming

$$H(a_{i_1}, a_{i_2}) = H(a_{i_1}, a_{i_2}) \sum_{a_{i_3}} H(a_{i_2}, a_{i_3}) H(a_{i_1}, a_{i_3})$$
$$\dots \sum_{a_{i_{N-1}}} H(a_{i_{N-2}}, a_{i_{N-1}}) H(a_{i_1}, a_{i_{N-1}})$$
$$\sum_{a_{i_N}} H(a_{i_{N-1}}, a_{i_N}) H(a_{i_1}, a_{i_N}).$$

This can be further written into a dynamic programming multi-layer Boolean algebra

$$H^0(a_i, a_j) = H(a_i, a_j)$$
$$H^l(a_i, a_j) = H^{l-1}(a_i, a_j) \sum_{a_k} H^{l-1}(a_i, a_k) H^{l-1}(a_j, a_k)$$

where $1 <= l <= n$.

Note that $H^0()$, $H^l()$ and $H^{l-1}()$ are the indexes of H (does not mean H is raised to the power of 0, l or $l-1$). n is the number of a_i variables.

9 Conclusion

We have proved Lemma 4.1 and Lemma 7.1, therefore NP $\neq$ P. To make the Boolean algebra simplification proof possible, we have explained how to represent NP problem using Discrete Markov Random Field, solve Markov Random Field using Boolean algebra operations, and reformulate the Boolean algebra so that only the first layer contains 'Not' operations while keeping the time complexity of the Boolean algebra to Polynomial time. To make the proof more complete, we have shown example of how a 6 variables NP problem represented in Markov Random Field can be factored and why the factored form is still NP in terms of time complexity, have shown how to factor a Polynomial time Markov Random Field Chain, and have shown algorithm of how to solve a 2sat (represented as Markov Random Field problem with only 2 possible values for each variable) in Polynomial time.

A quick explanation of why a Non-Deterministic Polynomial time Boolean algebra cannot be factorized into Polynomial time Boolean algebra, you can look at subsection 6.9. This Boolean algebra cannot be factorized into $(A+B+\ldots)(C+D+\ldots)$ form and therefore it cannot be factorized into higher order factorized Polynomial time form.

References

[1] S. Cook, "The p versus np problem," *The millennium prize problems*, pp. 87–104, 2006.

[2] L. Fortnow, "The status of the p versus np problem," *Communications of the ACM*, vol. 52, no. 9, pp. 78–86, 2009.

[3] S. Aaronson, "P $\stackrel{?}{=}$ NP," in *Open problems in mathematics*. Springer, 2016, pp. 1–122.

[4] M. Sipser, "The history and status of the p versus np question," in *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, 1992, pp. 603–618.