**Estrategia QNodes**

Objetivo: Esta estrategia desarrolla un algoritmo sobre un sistema dinámico discreto, representado por un conjunto de elementos (en uno de dos estados posibles), que busca encontrar una partición del conjunto en dos subconjuntos que produzcan la menor perdida respecto al conjunto original.

**Descripción detallada**

- Contexto del Sistema:

a) El sistema completo está compuesto por elementos binarios que esta en un estado inicial (pueden estar en estado 0 o 1).

b) Cada elemento (que puede ser un nodo) se considera en dos instantes de tiempo: t y t+1.



c) Ejemplo de sistema completo: At, Bt, Ct, Dt, Et, Ft. El sistema tendrá n elementos.

d) El sistema se describe mediante una Matriz de Transición de Probabilidades (TPM). La matriz describe las probabilidades de transicionar de un estado en el presente a cualquier estado en el futuro. Filas: estados en t, Columnas: estados en t+1. Cada entrada TPM[i][j] es la probabilidad de pasar del estado i en t, al estado j en t+1.

e) Para un conjunto de elementos en t, calcular la distribución de probabilidades sobre los estados de los elementos en t+1 (Pelementost+1| Pelementost ). Esta distribución se obtiene sobre unas operaciones en la TPM teniendo en cuenta que los elementos que no están considerados tiene un tratamiento diferente (más adelante se describe este proceso).

f) El sistema parte de un estado inicial en el que se encuentran todos los elementos del mismo.

- Subconjunto de Análisis:

a) El análisis se puede realizar sobre el sistema completo o un subconjunto de elementos del mismo. Al sistema sobre el cual se hará el análisis se le llama Sistema candidato.

b) Ejemplo de sistema candidato: A, B, C, D.

c) Los elementos fuera del sistema candidato se consideran como **condiciones de background**.

d) De igual forma, se puede trabajar sobre un subconjunto del sistema candidato, para lo cual se realizan procesos de marginalización.

- Condiciones de Background:

a) Los elementos fuera del sistema candidato se fijan en su estado actual.

b) Estos elementos no se tienen en cuenta durante el análisis y se eliminan de la siguiente forma: Para los elementos en el presente o tiempo actual se fijan las condiciones de background y para los elementos en el futuro se hace marginalización

c) Sus estados influyen cuando se determinan las probabilidades de transición del sistema candidato a analizar.

- Matriz de Transición de Probabilidades (TPM):

a) La TPM del sistema candidato se deriva de la TPM del sistema completo.

b) Se obtiene condicionando la TPM completa sobre los estados en t de los elementos de background.

c) Luego se marginaliza sobre los elementos de background en t+1.

- Distribuciones de Probabilidad:

a) Para un subconjunto de elementos del sistema candidato, se calcula la distribución de probabilidad de los estados de los elementos en t+1 dado los estados de los elementos en t, que pertenecen al subconjunto del sistema candidato a analizar.

b) Esta distribución se obtiene marginalizando la TPM sobre los elementos no considerados del sistema candidato, tanto en t como en t+1.

- Distancia entre las distribuciones de probabilidades (EMD):

Esta es la forma como se determina que tan diferentes son dos distribuciones de probabilidades. Existen varias medidas de distancia. Inicialmente usaremos la EMD.

- División del sistema a analizar:

Se pretende hacer una división del sistema en dos subconjuntos naturalmente disyuntos Cada una de las partes puede tener elementos de t como de t+1. Por ejemplo: Parte 1= (Bt+1|{}) y Parte 2= P(At+1Ct+1|AtBtCt), pudiendo no quedar en alguna de las particiones elementos de t o elementos de t+1 (eso se ilustra con { }) . El objetivo es hallar la división del sistema que resulta en la menor diferencia (medida por EMD) entre la distribución de probabilidades del conjunto de elementos sin dividir y la distribución de probabilidades del conjunto de elementos dividido. En teoría para hallar el mínimo valor de esta diferencia tendríamos que revisar todas las posibles divisiones del conjunto.

**Macroalgoritmo**

- Entrada de datos

* La TPM del sistema completo.
* El subconjunto de elementos a analizar (sistema candidato) aquí solo se requiere n los elementos en t.
* El subconjunto del sistema candidato a analizar (aquí deben darse los elementos tanto en t como en t+1, ya que no necesariamente se tendrán en t+1 los mismos elementos que en t)
* El estado actual de todos los elementos del sistema

- Proceso:

Comenzar con un conjunto V de todos los elementos en t y en t+1. Es decir, V tendrá los nodos del subsistema del conjunto candidato a analizar.

a) Inicializar W0 = ∅ y W1 = {v1}, donde v1 es un elemento arbitrario de V.

b) Iteración Principal: Para i = 2 hasta n (donde n es el número de elementos en V) se calcula :

* Encontrar vi ∈ V \ Wi-1 que minimiza: g(Wi-1 ∪ {vi}) - g({vi})

donde g(X) es la función EMD entre la distribución de probabilidades resultante de P( X) ⊗ P( X’) y la distribución del sistema sin dividir, asi: EMD(P(X) ⊗ P( X’), P(V) )

* Establecer Wi = Wi-1 ∪ {vi}

c) Construcción de Pares:

* El par (vn-1, vn) forma un "par candidato".

d) Recursión:

* Si |V| > 2, repetir el proceso con V' = V \ {vn-1, vn} ∪ {u}, donde u representa la unión de vn-1 y vn.
* Continuar hasta que |V| = 2.

e) Evaluación Final:

Para cada par candidato (a, b) encontrado evaluar la división que separa {b} del resto de nodos.

La división con el menor valor de diferencia es la solución al problema.

**Nota: Es importante tener en cuenta que cuando se va a calcular una distribución de probabilidades sobre algún subconjunto del sistema candidato en t+1, podemos tomar el producto de todos los repertorios individuales de los elementos en el subconjunto gracias a la independencia condicional, teniendo en cuenta la siguiente definición:**

**p(ABCt+1|ABCt) = p(At+1|ABCt ) X p(Bt+1|ABCt) X p(Ct+1|ABCt)**

**Ejemplo**

Supongamos que tenemos un sistema con 4 nodos, por ejemplo: V = {at,bt,at+1,bt+1}. Además,

**Paso 1**: Se define la función **g** a usar

En este caso, usamos la **g** =**EMD** como se explicó anteriormente:

**g** **= EMD(P(M) ⊗ P(M̄)**, **P(V))** donde M es un subconjunto de V y M̄ es su complemento.

**Paso 2:** Según el algoritmo descrito anteriormente comenzamos con el subsistema a trabajar representado en el conjunto V = {at, bt, at+1, bt+1}.

**Paso 3**: Encontrar un par candidato

a) Se toma un elemento inicial cualquiera, por ejemplo, v₁ = at.

b) Se construye la secuencia:

W₀ = ∅

W₁ = {at}

c) Para i = 2, 3, 4, se calcula:

v₂ = arg min[g(W₁ ∪ {u}) - g({u})] para u ∈ { bt,at+1,bt+1},

= arg min[EMD({at} ∪ {bt} ) - g({bt})]

=arg min [EMD (P({}|atbt) ⊗ P(at+1bt+1|{}), P(V)) - EMD(P({}|bt) ⊗ P(at+1bt+1|at), P(V))

= 0,4

Y así se prueba con todos los demás elementos del conjunto V – {at}.

Supongamos que al terminar todas estas iteraciones v₂ = bt+1

Luego definimos W2 y seguimos con el cálculo de v3

W2 = {at, bt+1}

v₃ = arg min[g(W₂ ∪ {u}) - g({u})] para u ∈ { bt,at+1}, supongamos que cuando este ciclo termine v₃ = bt

…

W3 = {at, bt+1,bt} y seguimos con el calculo de v4

v₄ = sería el nodo que falta, por tanto v₄ = at+1

"Recordar que Wi:= Wi−1 ∪ {vi}"

Al finalizar la primera iteración del ciclo externo tendríamos secuencia: v₁ = at, v₂ = bt+1, v₃ = bt, v₄ = at+1

d) El par candidato es (v₃, v₄) = (bt, at+1)

**Paso 4**:

a)Considerar lo siguiente:

1)U₁ = { at+1} (último elemento del par candidato) .

2)Aquí llegaríamos a una partición que sería (at+1 ) y (bt+1at bt}) y esta sería nuestra primera partición candidata, que se arma dejando el ultimo elemento del par candidato en un subconjunto (para este ejemplo, at+1) y los otros elementos en otro

b) Fusionar (v₃, v₄) o sea (bt, at+1) en un nuevo elemento u' = { bt, at+1}

**Paso 5: Recursión**

Se repite el proceso encontrando un nuevo par candidato con el nuevo conjunto V que es mas pequeño y que llamaremos V' = {at, bt+1, u'}

**Paso 6: Continuar la recursión** donde tenemos V' = {at, bt+1, u'}, donde u' = { bt, at+1}

Aquí volvemos a empezar

6.1. Encontrar un nuevo par candidato en V':

a) Elegimos un elemento cualquiera, por ejemplo v₁ = at.

b) Se construye la secuencia:

W₀ = ∅

W₁ = {at}

c) Se calcula para i= 2, 3

v₂ = arg min[g(W₁ ∪ {u}) - g({u})] para u ∈ {bt+1, u'}

= arg min[EMD({at} ∪ {bt+1} ) - g({bt+1})]se procede como se explicó previamente. Supongamos que después de todas las revisiones se elige bt+1

v₃ = se elige rápido ya que solo quedaría un elemento restante que sería u’

Supongamos que esto resulta en: v₁ = at, v₂ = bt+1, v₃ = u'

El par candidato es (v2, v3) =(bt+1, u')

6.2. Considerar lo siguiente:

a) U₂ = {u'} = { bt, at+1} último elemento del par candidato. Aquí llegaríamos a otra partición que sería ( at+1 bt )y (bt+1 at ) y esta sería nuestra segunda partición candidata, que se arma dejando el ultimo elemento del par candidato en un subconjunto (para este ejemplo { bt, at+1} y los otros elementos en otro, {bt+1, at }.

b) Fusionar bt+1 y u' en un nuevo elemento u'' = { bt+1, bt, at+1}

6.3. Recursión final:

Ahora tenemos V'' = { at, u''} = { at, { bt+1, bt, at+1}}

Como solo quedan dos elementos, no necesitamos encontrar otro par candidato

La última partición a considerar es { at } y { bt+1, bt, at+1}

**Paso final.** Se revisan las particiones:

A lo largo de este proceso, hemos generado las siguientes particiones candidatas:

Del ciclo 1 {at+1} y {bt+1 at bt }}

Del ciclo 2: {at+1 bt } y {bt+1 at  }

Del ciclo 3: { at } y { bt+1, bt, at+1}

-   Para para cada una de estas particiones se calcula la función g=EMD.

**Supongamos** que los valores de g son:

EMD(P(at+1|{}) ⊗ P(bt+1 |at bt ), P(V)) = 0,8

EMD (P(at+1| bt ) ⊗ P(bt+1 |at  ), P(V)) = 0,5

EMD (P({}|at ) ⊗ P(at+1 bt+1| bt,), P(V))= 0,7

 Por último se selecciona la partición con el valor mínimo de EMD.

En este caso, sería (at+1|bt ) y (bt+1 |at ) ya que tiene el valor mínimo de g=EMD = 0.5.

Observaciones (no olvidar para cuando se haga el seguimiento)

En cada etapa de la recursión, cuando encontramos un par candidato (a,b), consideramos dos posibilidades:

- Tomar {b} como un conjunto de la partición.

- Fusionar a y b, y continuar con el proceso.

Cada vez que tomamos la primera opción (separar {b}), generamos una partición candidata.

La fusión en la segunda opción no genera una partición por sí misma, sino que reduce el tamaño del conjunto para la siguiente iteración.

El proceso continúa hasta que solo quedan dos elementos, momento en el cual se genera la última partición candidata.

Al final, tendremos N-1 particiones candidatas (donde N es el número original de elementos), cada una correspondiente a una etapa diferente del proceso recursivo.

Este proceso evita tener que considerar todas las 2N-1 particiones posibles.

La clave está en que, en cada paso, el algoritmo busca la partición óptima entre las generadas por separar el último elemento del par candidato o entre las que se generarán en las siguientes iteraciones con los elementos fusionados.

- Consideraciones de Eficiencia:

Analizar la complejidad temporal del algoritmo implementado.

Analizar la complejidad espacial del algoritmo implementado