LA COMPLEJIDAD COMPUTACIONAL DE PREDECIR

J. ANDRÉS MONTOYA.

Abstract. El objetivo de este escrito es describir el estado del arte de una area de trabajo en la interface de la Fisica, las Matematicas y las ciencias de la computacion, a saber: el analisis de la complejidad computacional de predecir la dinamica de sistemas dinamicos discretos provenientes de la mecanica estadistica. En este escrito introducimos la terminologia basica y se establecen algunos hechos fundamentales de la teoria.

1. Introducción

¿Podemos predecir la evolucion de los fenomenos naturales? ¿Podemos predecir la evolucion de los sistemas dinamicos? Predecir es una de las tareas fundamentales de la ciencia natural. Es bien conocida la sentencia de P. S. Laplace afirmando que si se le hubiera permitido conocer las posiciones y velocidades de cada una de las particulas del universo en un instante dado, el habria podido predecir la evolucion del universo de ese instante en adelante. La afirmacion de Laplace es una afirmacion vana, dado que una descripcion exhaustiva del universo implica usar al universo en su totalidad (la cantidad de informacion a ser codificada excede al numero de particulas en el universo). Por otro lado podemos afirmar que el siglo XX presencio lo que podriamos llamar *la conciencia del caos,* esto es: el reconocimiento de que existen procesos dinamicos esencialmente impredecibles. En este escrito pretendemos analizar la complejidad computacional de predecir sistemas dinamicos finitos de los cuales:

(1) Es posible describir completamente las configuraciones del sistema.
(2) Es posible predecir la evolucion del sistema, si se conoce una descripcion total de la configuracion del sistema en un instante dado.

Asi pues, los sistemas dinamicos que pretendemos estudiar son predecibles en el siguiente sentido: si suponemos conocida la configuracion del sistema en un instante dado, podemos calcular, sin necesidad de observar el sistema nuevamente, una descripcion de la configuracion del sistema para cualquier instante posterior al instante inicial (esto es, al instante del cual se conoce la configuracion del sistema).

Esta nocion de predictibilidad, que podriamos llamar *simulabilidad*, es una nocion debil que no captura uno de los aspectos centrales de la nocion intuitiva de predictibilidad, a saber: La posibilidad de predecir los eventos que habran de ocurrir mucho antes de que estos ocurran.

Permitannos introducir la definicion de un problema algoritmico abstracto, definicion que pretendemos usar para aclarar algunos puntos oscuros en la discusion precedente

Key words and phrases. Automatas celulares, Clases de complejidad, Mecanica estadistica, Prediccion, Algoritmos eficientes, Sistemas dinamicos discretos.

Problema 1. *(El problema de la prediccion PP)*

- *Input:* $(\mathcal{S}$,$c,t)$, *donde* $\mathcal{S}$ *es un sistema dinamico,* c *es una configuracion inicial y* $t \in \mathbb{N}$.
- *Problema: Calcule la configuracion del sistema en el instante* t, *esto es: calcule la configuracion a la que accede el sistema tras* t *iteraciones empezando en la configuracion* c.

Podemos plantearnos diferentes preguntas relativas a la predictibilidad del sistema $\mathcal{S}$.

(1) (*Predictibilidad debil, calculabilidad del problema* PP) ¿Es posible calcular la configuracion del sistema $\mathcal{S}$ en el instante t?
(2) (*Predictibilidad fuerte, complejidad del problema* PP) ¿Es posible calcular la configuracion del sistema $\mathcal{S}$ en el instante t en mucho menos que t unidades de tiempo? O, lo que es lo mismo, ¿podemos predecir los eventos que ocurriran en el instante t mucho antes de que estos ocurran? O, lo que tambien es lo mismo, ¿existen algoritmos de tiempo sublineal (respecto a t) que resuelvan el problema PP?

Los sistemas que consideraremos en este escrito no son sistemas caoticos: es posible calcular las configuraciones a las que acceden estos sistemas para cualquier instante en el futuro ($t \geq 0$), si se conoce la configuracion del sistema en el instante 0. La pregunta que nos interesa considerar en este escrito es la pregunta referente a la complejidad del problema PP (o, lo que es lo mismo, la pregunta referente a la predictibilidad fuerte de los sistemas bajo consideracion). Los sistemas que consideraremos en este escrito son fuertemente simulables, es decir: dada una instancia $(\mathcal{S}, c, t)$ de PP, podemos calcular en t unidades de tiempo la configuracion del sistema $\mathcal{S}$ en el instante t, dado que podemos simular la dinamica del sistema empleando una unidad de tiempo por iteracion. La pregunta central referente a este tipo de sistemas es si podemos predecir (en el sentido fuerte) la dinamica del sistema, esto es: ¿podemos calcular la configuracion del sistema en el instante t, sin simular la dinamica iteracion por iteracion? ¿Existen algoritmos de tiempo sublineal que resuelvan el problema PP?

Relaciones con otros trabajos. Este trabajo esta estrechamente relacionado con el trabajo de los grupos de C. Moore y J. Machta, el lector interesado puede consultar las referencias [14], [15]. Estudiar la complejidad computacional de predecir sistemas dinamicos discretos(finitos) es esencialmente equivalente a estudiar la complejidad de simular dichos sistemas, esto es asi dado que los sistemas dinamicos que nos interesan pueden ser simulados en un procesador estandard, por lo que de entrada tenemos una manera de predecir algoritmicamente (esto es de calcular) el estado al que accede el sistema en un tiempo t arbitario. La pregunta central de los trabajos antes referidos es si existen *short cuts* (atajos) que nos permitan predecir la dinamica del sistema sin necesidad de simularlo, esto es: sin necesidad de observar la totalidad de la dinamica del sistema. ¿Podemos ser mas veloces que el sistema, arrivando en menor tiempo al estado final, gracias a algun atajo que nos permita acceder a este estado final evitando realizar una cantidad significativa de las transiciones que constituyen la dinamica del sistema? ¿Podemos simular la dinamica del sistema sin comportarnos de manera tan parsimoniosa como se comporta el sistema mismo? Los analisis de Machta y Moore se basan en la nocion de *P-completez* [16], una de las nociones centrales de la complejidad computacional y la nocion central de la *teoria de la paralelizabilidad*. Machta y Moore han logrado

probar, de una amplia variedad de sistemas, o bien que el problema de la prediccion asociado es P-completo, o bien que dicho problema admite algoritmos paralelos de *tiempo sublineal*.

2. La complejidad de predecir

En esta seccion intentaremos definir de manera precisa los problemas de prediccion que pretendemos analizar.

2.1. **Sistemas dinamicos discretos.** Un *sistema dinamico finito* es una tripla (G, Q, T) tal que

(1) G es un grafo finito.
(2) Q es un conjunto finito de estados.
(3) $T : Q^{V(G)} \to Q^{V(G)}$ es un operador que define la dinamica del sistema. Supondremos ademas que el operador T es *continuo*, esto es: dado $v \in V(G)$ y dadas $f_1, f_2 \in Q^{V(G)}$ dos *configuraciones*

$$\text{si } f_1 \restriction_{N(v)} = f_2 \restriction_{N(v)} \text{, entonces } (T(f_1))(v) = (T(f_2))(v)$$

donde $N(v)$ denota la vecindad de v en el grafo G.

Ejemplos. A continuacion listamos algunos ejemplos de sistemas dinamicos finitos que consideramos de especial interes.

(1) Automatas celulares finitos [23].
(2) Redes neuronales discretas, como por ejemplo las redes de Hopfield [9].
(3) Dada $\mathcal{M}$ una maquina de Turing de espacio acotado y dado x un input de $\mathcal{M}$ es posible codificar la dinamica de la maquina $\mathcal{M}$ al procesar el input x como un sistema dinamico finito $\mathcal{G}(\mathcal{M}, x)$. De igual manera, dado $\mathcal{M}$ un automata de espacio acotado (como por ejemplo un automata regular, un automata de pila o un automata linealmente acotado) y dado x un input de $\mathcal{M}$, podemos codificar el sistema dinamico definido por la computacion de $\mathcal{M}$, en el input x, como un sistema dinamico finito $\mathcal{G}(\mathcal{M}, x)$.
(4) Son muchos los ejemplos interesantes de sistema dinamicos finitos provenientes de la mecanica estadistica, entre ellos quiseramos citar el modelo de pilas de arena [1], el modelo de la hormiga de Langton [10], el automata de Ising [11] , la *flipping Ising dynamics* [15], [22] *y* el modelo de caminantes eulerianos [17] y [6].

3. El problema de la prediccion: resultados preliminares

Sea $\mathcal{C}$ una coleccion de sistemas dinamicos finitos, definimos el problema $pred[\mathcal{C}]$ de la siguiente manera.

Definición 1. *(El problema de la prediccion: $pred[\mathcal{C}]$)*

- *Input:* $(\mathcal{M}, f, t)$, *donde* $\mathcal{M} = (G, Q, T) \in \mathcal{C}$, $f \in Q^{V(G)}$ *es una configuracion inicial para* $\mathcal{M}$ *y* t *es un entero positivo.*
- *Problema: Calcule la configuracion del modelo tras* t *iteraciones, esto es: calcule la configuracion a la que accede* $\mathcal{M}$ *tras* t *aplicaciones del operador* T *(*t *unidades de tiempo), si la configuracion inicial (la configuracion en el instante cero) es* f.

Sea (G,Q,T) un sistema dinamico finito. Note que el conjunto de las configuraciones posibles tiene un tamaño acotado por $|Q|^{|V(G)|}$, esto implica que tras a lo mas que $|Q|^{|V(G)|}$ iteraciones el sistema $((G,Q,T),f)$ accede a un punto fijo o a un ciclo, esto es: empieza a comportarse periodicamente. Podemos clasificar a la coleccion de los sistemas dinamicos finitos en dos clases: la clase de los sistemas de punto fijo y la clase de los sistemas periodicos. Lo anterior implica que si $t \gneqq |Q|^{|V(G)|}$, es posible entonces calcular la configuracion del sistema en el instante t en menos que t unidades de tiempo (en *mucho* menos que t unidades de tiempo si t es *inmenso*). Es por ello que restringiremos nuestra atencion a los siguientes problemas de prediccion

Problema 2. *(FSPP, prediccion del estado final)*

- *Input:* $(\mathcal{M},f)$, *donde* $(\mathcal{M},f)$ *es un sistema de punto fijo y* f *es una configuracion.*
- *Problema: Determine la configuracion final del sistema* $(\mathcal{M},f)$.

Problema 3. *(PSPP, prediccion de sistemas periodicos)*

- *Input:* $((G,Q,T),f,t)$, *donde* $((G,Q,T),f,t)$ *es un sistema periodico y* $t \leq |Q|^{|V(G)|}$.
- *Problema: Determine la configuracion del sistema en el instante* t.

Nota 1. *El lector interesado en adquirir el background basico de la teoria de la complejidad computacional puede consultar la excelente referencia* [16].

En lo que sigue centraremos nuestra atencion en sistemas de punto fijo, en el problema $FSPP$ y en algunas restricciones importantes del mismo.

Definición 2. *Dada* $\mathcal{C}$ *una clase de sistemas dinamicos finitos, el problema* $FPPP[\mathcal{C}]$ *es el subproblema de* $FSPP$ *constituido por el conjunto de instancias*

$$\{((G,Q,T),f) : (G,Q,T) \in \mathcal{C}\}$$

$PSPP[\mathcal{C}]$ *se define de manera similar.*

Nota 2. *Dada* $((G,Q,T),f)$ *una instancia de* $FSPP[\mathcal{C}]$ *(o de* $PSPP[\mathcal{C}]$*) es correcto asumir que el tamaño de la instancia es igual a* $|V(G)|$.

Es claro que la complejidad del problema $FSPP[\mathcal{C}]$ depende de la clase $\mathcal{C}$. A continuacion presentaremos y estudiaremos algunas restricciones de $FSPP$. Con estos ejemplos queremos ilustrar tres cosas, a saber:

(1) Que la complejidad de un problema $FSPP[\mathcal{C}]$ depende por completo de la clase $\mathcal{C}$.
(2) Que existen problemas de la forma $FSPP[\mathcal{C}]$, los cuales pese a ser no triviales, pueden ser resueltos en tiempo sublineal (teoremas 4 y 3).
(3) Que la teoria de la complejidad puede proveernos de herramientas con las cuales probar que ciertos problemas de prediccion no pueden ser resueltos en tiempo sublineal (teorema 7).

Sistemas lineales finitos. Sea $\mathbb{F}$ un campo finito y sea $n \geq 1$. Un $\mathbb{F}$-sistema lineal de dimension n es un sistema dinamico sobre el espacio vectorial $\mathbb{F}^n$ completamente determinado por una matriz $M \in GL_{\mathbb{F}}(n)$. Dada $M \in GL_{\mathbb{F}}(n)$, dado $v \in \mathbb{F}^n$ y dado $t \geq 1$, se define la dinamica del sistema $[M]$ de la siguiente manera:

$$[M](v,t) = M^t v$$

Esto es: el espacio de configuraciones es el espacio vectorial $\mathbb{F}^n$, y si el sistema inicia en la configuracion v, la configuracion alcanzada por el sistema tras t unidades de tiempo es la configuracion $M^t v$.

Consider el siguiente problema

Problema 4. *($\mathbb{F}$-$lin(n)$: prediccion de $\mathbb{F}$-sistemas lineales de dimension n)*

- *Input:* (M, v, t), *donde* $M \in GL_{\mathbb{F}}(n)$, $v \in \mathbb{F}^n$ *y* $t \geq 1$.
- *Problema: calcule* $M^t v$.

Es claro que el problema $\mathbb{F}$-$lin(n)$ puede ser resuelto en tiempo $O(t)$, para calcular $M^t v$ es suficiente calcular t productos matriciales entre matrices de orden n, note que n es una constante que no depende del input ¿podemos hacerlo mejor?.

Teorema 1. *Para todo $\mathbb{F}$ y para todo n, existe un algoritmo $\mathcal{M}_{\mathbb{F},n}$ que permite resolver el problema $\mathbb{F}$-$lin(n)$ en tiempo $O(\log(t))$.*

Proof. Dado (M, v, t) un input de $\mathbb{F}$-$lin(n)$ podemos calcular la matriz M^t en tiempo $O(\log(t))$ usando el algoritmo de exponenciacion rapida. Note que una vez calculada M^t, podemos calcular $[M](v,t)$ realizando un producto matricial adicional (calculando el producto de M^t por v), lo cual toma tiempo constante dado que los ordenes de la matriz y del vector que debemos multiplicar son constantes que no dependen del input □

Sistemas afines finitos. Sea $\mathbb{F}$ un campo finito y sea $n \geq 1$. Un $\mathbb{F}$-sistema afin de dimension n es un sistema dinamico sobre el espacio vectorial $\mathbb{F}^n$ completamente determinado por una matriz $M \in GL_{\mathbb{F}}(n)$ y por un vector $v \in \mathbb{F}^n$. Dada $M \in GL_{\mathbb{F}}(n)$, dados $v, w \in \mathbb{F}^n$ y dado $t \geq 1$, se define la dinamica del sistema $[M, v]$ de la siguiente manera:

$$\begin{aligned} \text{Si } t &= 1 \text{ se define } [M,v](w,t) = Mw + v \\ \text{Si } t &\geq 1 \text{ se tiene que } [M,v](w,t) = M([M,v](w,t-1)) + v \end{aligned}$$

Considere el problema

Problema 5. *($\mathbb{F}$-$afin(n)$: prediccion de $\mathbb{F}$-sistemas afines de dimension n)*

- *Input:* (M, v, w, t), *donde* $M \in GL_{\mathbb{F}}(n)$, $v, w \in \mathbb{F}^n$ *y* $t \geq 1$.
- *Problema: calcule* $[M, v](w, t)$.

Es claro que el problema $\mathbb{F}$-$afin(n)$ puede ser resuelto en tiempo $O(t)$, (para calcular $M^t v$ es suficiente calcular t productos matriciales, y una cantidad lineal de sumas entre vectores de dimension fija), ¿podemos hacerlo mejor?.

Teorema 2. *Para todo $\mathbb{F}$ y para todo n, existe un algoritmo $\mathcal{N}_{\mathbb{F},n}$ que permite resolver el problema $\mathbb{F}$-$afin(n)$ en tiempo $O(\log(t))$.*

Proof. Dado (M, v, w, t) un input de $\mathbb{F}$-$afin(n)$ podemos calcular el conjunto

$$\left\{M^i : i \leq t\right\}$$

en tiempo $O(\log(t))$ usando el algoritmo de exponenciacion rapida y t procesadores. Note que

$$[M,v](w,t) = M^t w + \sum_{i=0}^{t-1} M^i v$$

Tenemos entonces que, una vez calculado el conjunto $\left\{M^i : i \leq t\right\}$, podemos calcular en tiempo constante el conjunto $\left\{M^i v : i \leq t-1\right\} \cup \left\{M^t w\right\}$ usando t procesadores. Una vez hecho este computo podemos calcular $M^t w + \sum_{i=0}^{t-1} M^i v$ en tiempo $O(\log(t))$. En total, es necesario emplear $O(\log(t))$ unidades de tiempo para correr el algoritmo $\mathcal{N}_{\mathbb{F},n}$ en el input (M,v,w,t) □

Automatas regulares. Dado $\mathcal{M} = (Q, q_0, F, \delta)$ un automata regular y dado $x \in \Sigma^*$, podemos ver al par $(\mathcal{M}, x)$ como un sistema de Boltzmann $(G(\mathcal{M},x), \Omega, T)$.

Dado $\mathcal{M}$ un automata regular y dado x un input de $\mathcal{M}$ podemos codificar el par $(\mathcal{M}, x)$ como el sistema de Boltzmann $(G(\mathcal{M},x), \Omega, T)$ definido por:

(1) $G(\mathcal{M},x)$ es el grafo lineal definido por el orden usual sobre el conjunto $\{1, ..., |x|\}$.
(2) $\Omega = Q \times \{0,1\} \times \Sigma$. Dada $x \in \Sigma^*$, y dado $i \leq |x|$ un vertice de $G(\mathcal{M},x)$ el estado de i en el instante t es igual a (q, ϵ, a), donde q es el estado interno de la maquina en el instante t, $\epsilon = 1$ si y solo si en el instante t la cabeza de la maquina esta ubicada sobre la celda i y $a = x_i$. Diremos que una configuracion $f : V(G(\mathcal{M},x)) \to \Omega$ es valida si y solo si existe un unico $i \leq |x|$ tal que $\pi_2(f(i)) = 1$ y para todo $i \leq |x|$ se tiene que $\pi_3(f(i)) = x_i$.
(3) T es el operador definido por: dada $f = \left((q, \epsilon_1, x_1), ..., \left(q, \epsilon_{|x|}, x_{|x|}\right)\right)$ una configuracion valida del sistema $(\mathcal{M}, x)$ se tiene que
   - Dado $i \leq |x|$ si $\epsilon_{i-1} = 1$, entonces $T(f)(i) = (\delta(x_{i-1}, q), 1, x_i)$.
   - Si $\epsilon_j = 1$ y $j \neq i-1$ se tiene que $T(f)(i) = (\delta(x_j, q), 0, x_i)$.

Nota 3. *Es posible codificar la computacion de cualquier automata de espacio acotado usando ideas similares a las empleadas en este ejemplo.*

Sea $\mathcal{REG}$ la clase de sistemas de Boltzmann constituida por los pares $(\mathcal{M}, x)$ tales que $\mathcal{M}$ es un automata regular. El problema de prediccion, asociado a la clase $\mathcal{REG}$, que quisieramos estudiar es el problema:

Problema 6. *($FSPP[\mathcal{REG}]$: prediccion de automatas regulares)*

- *Input:* $(\mathcal{M}, x)$, *donde* $(\mathcal{M}, x) \in \mathcal{REG}$.
- *Problema: Determine si* $\mathcal{M}$ *acepta* x.

Nota 4. *La definicion de automata regular implica que el tiempo de computo empleado por un tal automata al procesar un input x es igual a $|x|$ (el simbolo $|x|$ denota la longitud del input x), donde el tiempo de computo de $\mathcal{M}$ al procesar x es simplemente el numero de transiciones realizados por la maquina durante la computacion. Esto implica que el problema $FSPP[\mathcal{REG}]$ puede ser resuelto en tiempo $O(|x|)$, para ello es suficiente dada $(\mathcal{M}, x)$ una instancia de $FSPP[\mathcal{REG}]$, simular el automata $\mathcal{M}$ en el input x. La pregunta que quisieramos considerar es la siguiente: ¿es posible resolver $FSPP[\mathcal{REG}]$ de manera mas eficiente?*

Teorema 3. *Es posible resolver el problema* $FSPP\left[\mathcal{REG}\right]$ *en tiempo* $O\left(\log^2\left(n\right)\right)$ *usando un numero polinomial de procesadores*

Proof. Este teorema es un corolario del teorema 4 que enuciaremos y probaremos mas adelante □

Automatas de pila. Sea $\mathcal{PIL}$ la clase de sistemas de Boltzmann constituida por pares de la forma $\left(\mathcal{M},x\right)$, tal que $\mathcal{M}$ es un automata de pila y x es un input de $\mathcal{M}$. Nos interesa estudiar el problema de prediccion $FSPP\left[\mathcal{PIL}\right]$ definido por

Problema 7. *(*$FSPP\left[\mathcal{PIL}\right]$ *: prediccion de automatas de pila)*

- *Input:* $\left(\mathcal{M},x\right)$*, donde* $\left(\mathcal{M},x\right)\in\mathcal{PIL}$*.*
- *Problema: Determine si* $\mathcal{M}$ *acepta* x*.*

Teorema 4. *El problema* $FSPP\left[\mathcal{PIL}\right]$ *puede ser resuelto en tiempo* $O\left(\log^2\left(n\right)\right)$ *usando* $O\left(n^{15}\right)$ *procesadores.*

Proof. Es suficiente notar que el problema $FSPP\left[\mathcal{PIL}\right]$ es un subproblema del *parsing problem* para lenguajes libres de contexto. En [18] Russo prueba que el parsing problem para lenguajes libres de contexto puede ser resuelto en tiempo $O\left(\log^2\left(n\right)\right)$ usando $O\left(n^{15}\right)$ procesadores. □

Corolario 1. *El problema* $FSPP\left[\mathcal{REG}\right]$ *puede ser resuelto en tiempo* $O\left(\log^2\left(n\right)\right)$ *usando* $O\left(n^{15}\right)$ *procesadores*

Proof. Simplemente note que todo automata regular es un automata de pila. □

Maquinas de tiempo real. Las maquinas de tiempo real son maquinas de Turing, con una o multiples cintas, que al procesar un input x emplean $|x|$ unidades de tiempo. Diremos que un problema L es no trivial si y solo si procesar una instancia de L implica leer cada uno de los caracteres de x. Si L es no trivial, todo algoritmo secuencial que resuelva L emplea como minimo, al procesar una instancia x, $|x|$ unidades de tiempo. En este sentido las maquinas de tiempo real son maquinas secuenciales optimas, por lo que poder predecir en tiempo sublineal cualquier maquina de tiempo real es un hecho completamente no trivial. Dado $N\geq 1$ usaremos el simbolo $\mathcal{R}_N$ para denotar la coleccion de las maquinas de tiempo real y N cintas de trabajo. Dada $\mathcal{M}\in\mathcal{R}_N$ y dado x un input de $\mathcal{M}$ una *configuracion posible* de $\mathcal{M}$ es una descripcion del estado de la maquina en un instante dado, en terminos mas precisos una configuracion posible de $\mathcal{M}$, en el input x, es una tupla

$$\left(w_1,p_1,w_2,p_2,...,w_N,p_N,pos,q\right)$$

donde $w_1,w_2,...,w_N\in\left\{0,1\right\}^{\log(|x|)}$ son palabras que representan los contenidos de las cintas de trabajo, $p_1,p_2,...,p_N\leq|x|$ son enteros positivos que representan las posiciones de las cabezas de las N cintas de trabajo, $pos\leq|x|$ es un numero que representa la posicion de la cabeza lectora y q es el estado de la maquina.

Considere el problema $FSPP\left[\mathcal{R}_N\right]$ definido por

Problema 8. *(*$FSPP\left[\mathcal{R}_N\right]$ *: prediccion de maquinas de tiempo real).*

- *Input:* $\left(\mathcal{M},x\right)$*, donde* $\mathcal{M}$ *es una maquina de Turing de tiempo real con* N *cintas de trabajo y* x *es un input de* $\mathcal{M}$*.*

- *Problema: calcule el estado al que accede la maquina $\mathcal{M}$, al procesar el input x, en el instante $|x|$.*

Teorema 5. *Para todo $N \geq 1$ se tiene que el problema $FSPP[\mathcal{R}_N]$ puede ser resuelto en tiempo $O\left(\frac{n}{\log(n)}\right)$.*

Proof. Considere el algoritmo de prediccion descrito a continuacion:

Sea $(\mathcal{M}, x)$ un input de $FSPP[\mathcal{R}]$. Sea

$$w = (w_1, p_1, w_2, p_2, ..., w_N, p_N, pos, q)$$

una configuracion posible de $\mathcal{M}$ en el input x. Si se satisfacen las desigualdades

$$\begin{array}{rcl} |w_1|, |w_2|, ..., |w_N| & \leq & 2\log(|x|) \\ & & y \\ |p_1|, |p_2|, ..., |p_N| & = & \log(|x|) \end{array}$$

diremos que w es una *configuracion buena*. Dada w una configuracion buena, usaremos el simbolo $\mathcal{M}(w, x)$ para denotar la configuracion a la que accede $\mathcal{M}$ tras $\log(|x|)$ unidades de tiempo, si en el instante cero la maquina se encuentra en la configuracion w y la palabra x es el contenido de la cinta de entrada. Note que el conjunto de las configuraciones buenas tiene un tamaño acotado por $|x|^{2N+1}+|Q| = M$. Podemos usar M procesadores para calcular en tiempo $O(\log(|x|))$ una tabla de la funcion $ev_{\mathcal{M}}$ definida por

$$ev_{\mathcal{M},x}(w) = \mathcal{M}(w, x)$$

Podemos usar la tabla de la funcion $ev_{\mathcal{M},x}$ para cacular en tiempo $O\left(\frac{n}{\log(n)}\right)$ el estado final de $\mathcal{M}$, al procesar el input x. Para ello podemos partir la palabra x en $\frac{|x|}{2\log(|x|)}$ bloques de longitud $2\log(|x|)$. Usando la tabla y la configuracion inicial $c_0 = (\varepsilon, o, \varepsilon, 0, ..., \varepsilon, 0, 0, q_0)$ podemos calcular en una unidad de tiempo la configuracion a la que accede $\mathcal{M}$ tras $\log(|x|)$ unidades de tiempo si la computacion inicia con la maquina $\mathcal{M}$ en la configuracion c_0.

Suponga que hemos podido calcular en i unidades de tiempo la configuracion a la que accede $\mathcal{M}$ tras $i\log(|x|)$ unidades de tiempo iniciando en c_0. Llamemos c_i a esta configuracion y supongamos que c_i es igual a

$$(w_1, p_1, w_2, p_2, ..., w_N, p_N, pos, q)$$

Sea $c_i^b = (u_1, \log(|x|), u_2, \log(|x|), ..., u_N, \log(|x|), pos, q)$ la configuracion buena definida por:

Dado $k \leq N$ la palabra u_i es igual a

$$u_i = \begin{cases} w_i[p_i - \log(|x|), ..., p_i + \log(|x|)] \text{ si } \log(|x|) \leq p_i \leq |w_i| + \log(|x|) \\ \blacksquare^{\log(|x|)-p_i} w[0, ..., p_i + \log(|x|)] \text{ si } p_i \leq |w_i| + \log(|x|) \text{ y } p_i \nleq \log(|x|) \\ w_i[p_i - \log(|x|), ..., |w_i|] \square^{\log(|x|)-(|w_i|-p_i)} \text{ si } \log(|x|) \leq p_i \text{ y } p_i \ngeq |w_i| - \log(|x|) \\ \blacksquare^{\log(|x|)-p_i} w[0, ..., |w_i|] \square^{\log(|x|)-(|w_i|-p_i)} \text{ en caso contrario} \end{cases}$$

donde dada w el simbolo, $w[i, ..., j]$ denota la subpalabra de w ubicada entre la $(i+1)$-esima posicion y la j-esima posicion, $\blacksquare$ es un simbolo especial que indica el inicio de la cinta (cuando la cabeza lectora escanea el simbolo $\blacksquare$ se mueve una posicion a la derecha) y $\square$ es el caracter especial que se usa para indicar celda vacia.

Podemos usar la tabla de $ev_{\mathcal{M},x}$, la configuracion c_i y la configuracion c_i^b para calcular en tiempo constante la configuracion c_{i+1}. Tenemos entonces que podemos

usar $O(M)$ procesadores para calcualr en tiempo $O\left(\frac{|x|}{\log(|x|)}\right)$ la configuracion $c_{\frac{|x|}{\log(|x|)}}$ que es la configuracion final. □

Nota 5. *Hemos logrado un ahorro o aceleramiento (speed up) de orden logaritmico que nos permite resolver (para todo $N \geq 1$) el problema $FSPP[\mathcal{R}_N]$ en tiempo $O\left(\frac{n}{\log(n)}\right)$. ¿Es posible hacerlo mejor?*

Automatas linealmente acotados y maquinas de Turing.

Sea $p : \mathbb{N} \to \mathbb{N}$ una funcion computable y $\mathcal{T}_p$ la coleccion de las maquinas de Turing de espacio acotado por p. Suponga que $p(X)$ es una funcion polinomial (i.e. que puede ser acotada por un polinomio). Dada $\mathcal{M} \in \mathcal{T}_p$ y dado x un input de $\mathcal{M}$ podemos definir un sistema de Boltzman $(G_{\mathcal{M},x}, Q_{\mathcal{M},x}, T_{\mathcal{M},x})$ que codifica la computacion de $\mathcal{M}$ en el input x, sea $(G_{\mathcal{M},x}, Q_{\mathcal{M},x}, T_{\mathcal{M},x})$ el sistema de Boltzman definido por:

- Sea $n = |x|$. El grafo $G_{\mathcal{M},x}$ es un grafo constituido por dos componentes conexas, la primera que, denotamos con el simbolo $I_{\mathcal{M},x}$, es el grafo lineal con universo $\{1, ..., n\}$; la segunda que denotamos con el simbolo $W_{\mathcal{M},x}$, es el grafo lineal con universo $\{1, ..., p(n)\}$.
- $Q_{\mathcal{M},x,} = \Sigma_{\mathcal{M}} \times \{0, 1\} \times Q_{\mathcal{M}}$, donde $\Sigma_{\mathcal{M}}$ es el alfabeto de $\mathcal{M}$ y $Q_{\mathcal{M}}$ es el coooonjunto de estados internos de $\mathcal{M}$.
- ahora, definiremos la nocion de configuracion posible. Dado $f \in Q_{\mathcal{M},x}^{V(G_{\mathcal{M},x})}$ diremos que f es una configuracion posible si y solo si:

(1) Para todo $i \in I_{\mathcal{M},x}$ se tiene que $f(i) = (x_i, \epsilon, q)$, con $\epsilon \in \{0, 1\}$.
(2) Existe exactamente un $i \in I_{\mathcal{M},x}$ tal que $\pi_2(f(i)) = 1$.
(3) Existe exactamente un $j \in W_{\mathcal{M},x}$ tal que $\pi_2(f(j)) = 1$.
(4) Existe exactamente un $q \in Q_{\mathcal{M}}$ tal que para todo $i \in I_{\mathcal{M},x} \cup W_{\mathcal{M},x}$ se tiene que $\pi_3(f(i)) = q$.

Note que el conjunto de configuraciones posibles de $(G_{\mathcal{M},x}, Q_{\mathcal{M},x}, T_{\mathcal{M},x})$ es, esencialmente, el conjunto de configuraciones a las que puede acceder $\mathcal{M}$ a lo largo de su computacion en el input x.

- La definicion de $T_{\mathcal{M},x}$ depende de la definicion de

$$\delta_{\mathcal{M}} : Q_{\mathcal{M}} \times \Sigma_{\mathcal{M}} \times \Sigma_{\mathcal{M}} \to Q_{\mathcal{M}} \times \Sigma_{\mathcal{M}} \times \{\rightarrow, \leftarrow, \diamondsuit\} \times \{\rightarrow, \leftarrow, \diamondsuit\}$$

que es la funcion de transicion de la maquina $\mathcal{M}$. Sea $f = (v_1, ..., v_n, w_1, ..., w_m)$ una configuracion posible y sean i y j dos posiciones para las cuales las segundas componentes de v_i y w_j son iguales a 1.

$$T_{\mathcal{M},x}(f) = (v_1^*, ..., v_n^*, w_1^*, ..., w_m^*)$$

se define de la siguiente manera:

(1) Para todo $k \in I_{\mathcal{M},x}$ se tiene que $\pi_1(v_k^*) = x_k$.
(2) Dado $k \in I_{\mathcal{M},x}$ se tiene que $\pi_2(v_k^*) = 1$ si y solo si
o $k = i - 1$ y

$$\pi_3(\delta_{\mathcal{M}}(\pi_3(v_1), \pi_1(v_i), \pi_1(w_j))) = \leftarrow$$

o $k = i$ y

$$\pi_3(\delta_{\mathcal{M}}(\pi_3(v_1), \pi_1(v_i), \pi_1(w_j))) = \diamondsuit$$

o $k = i + 1$ y

$$\pi_3 \left(\delta_{\mathcal{M}} \left(\pi_3 \left(v_1 \right), \pi_1 \left(v_i \right), \pi_1 \left(w_j \right) \right) \right) = \rightarrow$$

(3) Dado $k \in W_{\mathcal{M},x}$ se tiene que $\pi_2 \left(w_k^* \right) = 1$ si y solo si
o $k = j - 1$ y

$$\pi_4 \left(\delta_{\mathcal{M}} \left(\pi_3 \left(v_1 \right), \pi_1 \left(v_i \right), \pi_1 \left(w_j \right) \right) \right) = \leftarrow$$

o $k = i$ y

$$\pi_4 \left(\delta_{\mathcal{M}} \left(\pi_3 \left(v_1 \right), \pi_1 \left(v_i \right), \pi_1 \left(w_j \right) \right) \right) = \diamondsuit$$

or $k = i + 1$ and

$$\pi_4 \left(\delta_{\mathcal{M}} \left(\pi_3 \left(v_1 \right), \pi_1 \left(v_i \right), \pi_1 \left(w_j \right) \right) \right) = \rightarrow .$$

(4) Se tiene ademas que:
- $\pi_1 \left(w_j^* \right) = \pi_2 \left(\delta_{\mathcal{M}} \left(\pi_3 \left(v_1 \right), \pi_1 \left(v_i \right), \pi_1 \left(w_j \right) \right) \right)$.
- Para todo $k \in I_{\mathcal{M},x}$ se tiene que

$$\pi_3 \left(v_k^* \right) = \pi_1 \left(\delta_{\mathcal{M}} \left(\pi_3 \left(v_1 \right), \pi_1 \left(v_i \right), \pi_1 \left(w_j \right) \right) \right)$$

- Para todo $k \in W_{\mathcal{M},x}$ se tiene que

$$\pi_3 \left(w_k^* \right) = \pi_1 \left(\delta_{\mathcal{M}} \left(\pi_3 \left(v_1 \right), \pi_1 \left(v_i \right), \pi_1 \left(w_j \right) \right) \right)$$

Considere el problema

Problema 9. *($FSSP\left[\mathcal{T}_p\right]$: prediccion de maquinas de Turing de espacio acotado por $p\left(X\right)$)*

- *Input: $\left(\mathcal{M}, x\right)$, donde $\mathcal{M} \in \mathcal{T}_p$ y x es un input de $\mathcal{M}$.*
- *Problema: calcule el estado final de la computacion de $\mathcal{M}$ en el input x.*

Teorema 6. *Si $p\left(X\right) \in \Omega\left(n\right)$ el problema $FSSP\left[\mathcal{T}_p\right]$ es $PSPACE$ duro.*

Proof. Primero que todo debemos definir la manera en que mediremos el tamaño de una instancia del problema FSP. Sea (G, Q, T) una instancia de FSP. Definimos $\left|(G, Q, T)\right|$, el tamaño de (G, Q, T), como $\left|V\left(G\right)\right| \left|Q\right|$, donde $V\left(G\right)$ el el conjunto de vertices de G. Note que toda configuracion f, sobre (G, Q, T), puede ser codificada como un vector v_f de dimension $\left|V\left(G\right)\right|$ y tal que cada una de sus entradas es un numero en el intervalo $\{1, ..., \left|Q\right|\}$. Podemos, en consecuencia, codificar a f usando una cantidad de espacio acotada por $O\left(\left|V\left(G\right)\right| \log\left(\left|Q\right|\right)\right)$. Por otro lado es claro que podemos calcular el punto fijo determinado por f simulando la dinamica del sistema (G, Q, T), cuando la configuracion inicial es f. Para tal fin podemos usar un algoritmo de simulacion ingenuo el cual guarda, en cada etapa de la computacion, la configuracion actual (i.e. el algoritmo guarda una y sola una configuracion, la ultima configuracion a la que accede el sistema (G, Q, T)). Tenemos entonces que es posible resolver el problema FSP usando una cantidad de espacio acotada por $O\left(\left|V\left(G\right)\right| \log\left(\left|Q\right|\right)\right)$ (usando espacio polinomial).

Sea $\mathcal{M}$ una maquina de Turing de espacio polinomial y sea $x = x_1 ... x_n$ un input de $\mathcal{M}$. Podemos suponer que $\mathcal{M}$ es una maquina de decision. Podemos pensar en el par $(\mathcal{M}, x)$ como en un sistema dinamico finito $(G_{\mathcal{M},x}, Q_{\mathcal{M},x}, T_{\mathcal{M},x})$. Considere el problema

- *Input: $\left(\mathcal{M}, x\right)$, donde $\mathcal{M}$ es una maquina de espacio polinomial y x es un input de $\mathcal{M}$.*
- *Problem: decida si $\mathcal{M}$ acepta x.*

Es claro que el problema asi definido es $PSPACE$-duro. Mostraremos que es posible reducir este problema al problema FSP. La reduccion se define de la siguiente manera:

Dado $(\mathcal{M}, x)$, calculamos $((G_{\mathcal{M},x}, Q_{\mathcal{M},x}, T_{\mathcal{M},x}), f_x)$, donde f_x es la configuracion posible de $\mathcal{M}$ determinada por x. Calculamos $FP(f_x)$, el punto fijo de f_x. Supongamos que $FP(f_x) = (v_1, ..., v_n, w_1, ..., w_m)$, calculamos $\pi_3(v_1)$. Si $\pi_3(v_1)$ es un estado de aceptacion de $\mathcal{M}$,aceptamos el par $(\mathcal{M}, x)$, en caso contrario rechazamos.

Es facil verificar que la reduccion es valida, y que es una reduccion que se puede calcular empleando espacio polinomial. Tenemos entonces que el problema FSP es $PSPACE$-duro □

Sea $p(X) = X$, la coleccion de los automatas linealmente acotados es la subclase de $\mathcal{T}_p$ constituida por las maquinas en $\mathcal{T}_p$ que cuentan con una unica cinta. Sea $\mathcal{LBA}$ la coleccion de los automatas linelamente acotados y sea $FSPP[\mathcal{LBA}]$ el problema definido por

Problema 10. *($FSPP[\mathcal{LBA}]$: prediccion de automatas linealmente acotados)*

- *Input:* $(\mathcal{M}, x)$*, donde* $(\mathcal{M}, x) \in \mathcal{LBA}$.
- *Problema: Decida si el automata* $\mathcal{M}$ *acepta* x.

El problema $FSPP[\mathcal{LBA}]$ no es otra cosa que el problema de aceptacion para automatas lineales el cual es $PSPACE$ completo [19], por lo que $FSPP[\mathcal{LBA}]$ es $PSPACE$ completo. Es importante comentar que existe un automata lineal, llamemoslo $\mathcal{M}_0$, tal que el problema de aceptacion para $\mathcal{M}_0$ (i.e. el problema de reconocimiento del lenguaje $L(\mathcal{M}_0)$) es $PSPACE$ completo, tenemos entonces que el problema de prediccion restringido a instancias de la forma $(\mathcal{M}_0, x)$, donde x es un input de $\mathcal{M}_0$, tambien es $PSPACE$ completo.

Teorema 7. $FSPP[\mathcal{LBA}]$ *no puede ser resuelto en tiempo sublineal, usando un numero polinomial de procesadores.*

Proof. Dada $f : \mathbb{N} \to \mathbb{N}$ una funcion tal que $f \in o(n)$, si $FSPP[\mathcal{LBA}]$ pudiera ser resuelto en tiempo $O(f)$ usando un numero polinomial de procesadores, entonces $FSPP[\mathcal{LBA}]$ perteneceria a $SPACE(f)$, pero esto es imposible dado que $FSPP[\mathcal{LBA}]$ es $PSPACE$ completo y el *Space Hierarchy Theorem* [16] implica que $SPACE(f) \subsetneqq PSPACE$. □

Nota 6. *No intentaremos ir un nivel mas arriba en la jerarquia de Chomsky* [16]. *Recuerde que esta jerarquia corresponde a:*

$$\mathcal{REG} \subsetneqq \mathcal{PIL} \subsetneqq \mathcal{LBA} \subsetneqq \mathcal{TURING}$$

Esto es: el siguiente nivel, esta constituido por la coleccion de las maquinas de Turing generales (irrestrictas). Note que esta ultima coleccion no puede ser vista como una clase de sistemas dinamicos finitos, dado que no es posible acotar el espacio de trabajo (el numero de celdas, de las cintas de la maquina, usadas durante la computacion) de una maquina de Turing arbitraria en terminos del tamaño del input. Los tres tipos de automatas estudiados en los ejemplos anteriores son automatas de espacio acotado, y es precisamente esto lo que nos permite pensar en sus computaciones como en dinamicas de sistemas dinamicos finitos.

La acotacion del espacio de trabajo, de los automatas antes estudiados, implica tambien que los problemas de prediccion asociados son computables. No es este el

caso con las maquinas de Turing generales, es un hecho conocido que el problema de la parada [16] *para maquinas de Turing generales es no decidible. Note que el problema de la parada es un problema de prediccion (prediga si la computacion de la maquina tiene un punto fijo).*

Nota 7. *Respecto al problema $PSSP$ y sus restricciones podemos comentar que existen algunos trabajos que estudian la complejidad algoritmica de los mismos. A manera de ejemplo podemos citar el trabajo de Floreen y Orponen* [9]*, quienes muestran que el problema de la predicion para redes de Hopfield es NP-duro.*

4. Sistemas de tiempo-real

En la seccion anterior estudiamos varios ejemplos de clases de sistemas de Boltzmann no triviales, en algunos casos pudimos exhibir algoritmos de tiempo sublineal que resuelven el problema de prediccion asociado a la clase, en algotros casos pudimos mostrar que el problema no puede ser resuelto en tiempo poly-logaritmico usando para ello nociones apropiadas de *completes* y *dureza* ($PSPACE$ completes). En lo que queda de este trabajo consideraremos unicamente sistemas de tiempo-real (que definiremos mas adelante), centraremos nuestra atencion en un tipo especial de sistema de tiempo-real.

Definición 3. *Dada $\mathcal{C}$ una clase de sistemas de punto fijo, diremos que $\mathcal{C}$ es una clase de sistemas de tiempo real si y solo si para todo $(G,Q,T) \in \mathcal{C}$ y para toda configuracion inicial f se tiene que el sistema (G,Q,T,f) alcanza un punto fijo tras a lo mas $|V(G)|$ iteraciones.*

Dada $\mathcal{C}$ una clase de tiempo-real, el problema de prediccion naturalmente asociado a esta clase es el problema definido por:

Problema 11. *($RTP[\mathcal{C}]$, real-time prediction of $\mathcal{C}$-systems)*

- *Input: (G,Q,T,f), donde $(G,Q,T) \in \mathcal{C}$ y f es una configuracion inicial del sistema.*
- *Problema: calcule la configuracion a la que accede el sistema tras $|V(G)|$ iteraciones.*

Se conjetura REF que es posible predecir en tiempo polilogaritmico la dinamica de los sistemas de tiempo-real, en lo que sigue mostraremos que la conjetura es falsa, obtendremos este resultado estudiando un tipo especial de sistemas de tiempo real: las pilas de arena tri-dimensionales.

4.1. Un estudio de caso: el modelo abeliano de pilas de arena.

Dado $n \geq 1$ el reticulo tri-dimensional de orden n es la grilla $\mathcal{L}_n = (V,E)$, donde $V = [n] \times [n]$ y E es igual a

$$\{((n,m),(s,t)) : |n-s| + |m-t| = 1\}$$

El reticulo de arena de orden n, que denotaremos con el simbolo $\mathcal{L}_n^A$ puede ser obtenido a partir de $\mathcal{L}_n$ agregando un vertice s, que llamaremos el *sumidero.* Adicionlamente, dado v un vertice en el borde de $\mathcal{L}_n$ agregamos $4 - \deg_{\mathcal{L}_n}(v)$ aristas conectando el sumidero y el vertice v. Note que para todo $v \in V(\mathcal{L}_n)$ se tiene que $\deg_{\mathcal{L}_n^A}(v) = 4$. Una configuracion sobre $\mathcal{L}_n^A$ es una funcion $f : [n] \times [n] \to \mathbb{N}$. Podemos pensar en una configuracion como en una funcion que a cada vertice de $\mathcal{L}_n^A$, distinto del sumidero, le asigna una cierta cantidad de granos de arena (una

pila de arena). Dado $i \in [n]$ y dada f una configuracion sobre $\mathcal{L}_n^A$ diremos que i es f-inestable si y solo si $f(i) \geq 4$. Un vertice f-inestable puede regalar un grano de arena a cada uno de sus vecinos y volverse estable. La regla de *toppling* (desbarrancamiento, disparo) que controla la dinamica del sistema esta definida por:

Supongamos fijado un oirden lineal sobre el conjunto $[n] \times [n]$. Si f_t es la configuracion del sistema en el instante t, la configuracion f_{t+1} esta definida por

$$f_{t+1}(i) = \begin{cases} f_t(i) - 4 \text{ si } i \text{ es menor vertice que es } f_t\text{-inestable} \\ f_t(i) + 1 \text{ si el menor vertice } f_t\text{-inestable es vecino de } i \\ f_t(i), \text{ en otro caso} \end{cases}$$

Dada f inestable, podemos aplicar la regla de toppling repetidas veces, si la aplicamos n veces obtenemos una secuencia

$$f = f_0, f_1, ..., f_n$$

de configuraciones. El sumidero nunca *dispara* granos de arena. El sumidero puede ser entendido como el espacio vacio que circunda la tabla (el reticulo) sobre el que estan dispuestas las pilas de arena: una vez un grano de arena cae al espacio vacio se pierde. El sumidero convierte al modelo de pilas de arena en un sistema disipativo, y en cuanto sistema disipativo es de esperar que se estabilize en algun momento.

Definición 4. *Dada f una configuaracion sobre $\mathcal{L}_n^A$ diremos que f es estable si y solo si para todo $i \leq n$ se tiene que $f(i) \leq 3$.*

Sea f inestable y sea $f_0, f_1, ..., f_i, ...$ la secuencia de configuraciones generada por f al aplicar la regla de toppling. El teorema fundamental de las pilas de arena afirma que si f es inestable existe N tal que $f_N = f_{N+1}$. Sea N el numero natural definido por

$$N = \min_i \{f_i = f_{i+1}\}$$

Diremos que f_N es la *estabilizacion* de f, que $f_0, ..., f_N$ es la *avalancha* generada por f y que N es la longitud de la avalancha generada por f. Usaremos el simbolo $st(f)$ para denotar la estabilizacion de f y el simbolo $A(f)$ para denotar la longitud de la avalancha generada por f

Definición 5. *Dada f una configuracion estable, diremos que f es una configuracion critica si y solo si no existe $A \subseteq [n] \times [n]$ tal que para todo $v \in A$ se tiene que $f(v) \lneqq \deg_A(v)$, donde $\deg_A(v)$ es igual a*

$$|\{u \in A : \{u, v\} \in E\}|$$

Las configuraciones criticas juegan un papel muy importante en la teoria del modelo de pilas de arena, dado que el conjunto de configuraciones criticas es el *atractor* del sistem, (si realizamos el siguiente experimento: elegimos un vertice, agregamos un grano de arena en este vertice, estabilizamos; entonces despues de un cierto numero de repeticiones obtenemos una configuracion, y de este instante en adelante todas las configuraciones a las que accedamos seran criticas REF).

Dado $n \geq 1$ usamos el simbolo δ_n para denotar la configuracion definida por

$$\delta_n(v) = \#\text{ de aristas que conectan a } v \text{ con el sumidero}$$

Teorema 8. *f es critica si y solo si tiene al estabilizar la configuracion $f + \delta_n$ todo vertice dispara exactamente una vez.*

El teorema anterior es un corolario del teorema de Dhar REF, este teorema nos permite diseñar un algoritmo de tiempo lineal (tiempo real) para el reconocimiento de las configuraciones criticas. Considere el siguiente problema

Problema 12. *($RR\left[2\right]$, Reconocimiento de configuraciones criticas bidimensionales)*

- *Input:*(n,f), *donde* $n\in\mathbb{N}^{+}$ *y* f *es una configuracion estable sobre* $\mathcal{L}_{n}^{A}$.
- *problema: decida si* f *es critica.*

Note que dado (n,f) un input de $RR\left[2\right]$, decidir si la configuracion f es critica o no, es equivalente a predecir la configuracion alcanzada por el sistema en el instante n^{2} cuando $f+\delta_{n}$ es la configuracion inicial, esto es asi dado que.

Proposición 1. *f es critica si y solo si $st\left(f+\delta_{n}\right)=f$.*

Proof. Sea f una configuracion sobre $\mathcal{L}_{n}$ y suponga que f es critica. Dado v un vertice de $\mathcal{L}_{n}$, se tiene que $st\left(f+\delta_{n}\right)\left(v\right)$ is equal to $f\left(v\right)+\delta_{n}\left(v\right)-P_{v}+G_{v}$, donde P_{v} denota el numero de granos que v pierde durante el proceso de estabilizacion y G_{v} denota el numero de los granos que v recibe durante el proceso. Note que P_{v} es igual a $4d_{v}$ con d_{v} igual al numero de ocasiones en que el vertice v dispara. Sabemos que para todo v la cantidad d_{v} es igual a 1. Se tiene entonces que $P_{v}=4$. Por otro lado se tiene que para todo v el numero G_{v} es igual a $\deg_{\mathcal{L}_{n}}\left(v\right)$, por lo que para todo v se tiene que $\delta_{n}\left(v\right)+G_{v}=4$. Tenemos entonces que para todo $v\in V\left(\mathcal{L}_{n}\right)$ la ecuacion

$$st\left(f+\delta_{n}\right)\left(v\right)=f\left(v\right)+\delta_{n}\left(v\right)-P_{v}+G_{v}=f\left(v\right)$$

vale. Es facil verificar que si f no es critica, la configuracion $st\left(f+\delta_{n}\right)$ es diferente de f. □

Usando el teorema Dhar y la proposicion anterior podemos diseñar un algoritmo de tiempo lineal que resuelva el problema $RR\left[2\right]$, sea *Burning* el algoritmo definido a continuacion.

En el input (n,f) el algoritmo Burning hace lo siguiente:

(1) Simula la avalancha generada por $f+\delta_{n}$ y calcula $st\left(f+\delta_{n}\right)$.
(2) Verifica que $st\left(f+\delta_{n}\right)$ sea igual a f, si es el caso acepta, en caso contrario rechaza.

De lo anterior tenemos que el algoritmo es correcto, y tenemos ademas que su tiempo de computo es n^{2} (asumiendo que cada dispara puede ser simulado en una unidad de tiempo). Note que resolver el problema $RR\left[2\right]$ (de interes para los fisicos estadisticos) es equivalente a resolver el problema de prediccion.

Problema 13. *($PCA\left[2\right]$, Prediccion con arena)*

- *Input:* $\left(n,f,n^{2}\right)$, *donde* $n\geq1$ *es un entero y* f *es una configuracion sobre* $\mathcal{L}_{n}^{A}$.
- *Problema: calcula la configuracion a la que accede el sistema tras* n^{2} *unidades de tiempo.*

tenemos entonces que resolver el problema $RR\left[2\right]$ en tiempo sublineal (sub-real) es equivalente a predecir la dinamica del modelo de pilas de arena bidimensional, y la existencia de un tal algoritmo de prediccion es equivalente a la existencia de un algoritmo para $RR\left[2\right]$ mas eficiente que el mejor algoritmo conocido: ¡predecir es lo mismo que optimizar algoritmos!

En lo que queda de este escrito consideraremos la pregunta: ¿existen algoritmos de tiempo sublineal para el problema RR[2]?

5. Conclusiones y problemas

Agradecimientos: El autor quisiera agradecer a VIE-UIS y a Colciencias proyecto 111518925292.

Referencias

[1] P. Bak. *How Nature Works: The Science of Self-organized Criticality.* New York, Copernicus, 1996.
[2] C. Bennett. Logical depth and physical complexity. En *The Universal Turing Machine: A Half Century Survey* (paginas 227-257), editado por R. Herken, Oxford University Press, Oxford, 1998.
[3] B. Bjorner, L. Lovasz. Chip Firing Games on Directed Graphs. Journal of Algebaic combinatorics, 1(1991): 305-328.
[4] C. Cercignani, R. Penrose. *Ludwig Boltzmann: The Man Who Trusted Atoms.* Oxford University Press, Oxford, 2004.
[5] A. Condon. A Theory of Strict P-completeness. STACS(1992):33-44.
[6] J. Cooper, J. Spencer. Simulating a Random Walk with Constant Error. Combinatory, Probability and Computation. 15(6): 815-822, 2006.
[7] D. Dhar. The Abelian Sandpile Model of Self-organized Criticality. AIP conference proceedings 248(1991):124-136.
[8] A. Engel. Why Does the Probabilistic Abacus Works? Educational Studies in Mathematics, 7(1-2;1976):56-69.
[9] P. Floreen, P. Orponen. ON the Complexity of Analyzing Hopfield Nets. Complex Systems, 3(1989):577-587.
[10] A. Gajardo, E. Goles, A. Moreira. Complexity Of Langton´s Ant. Discrete applied mathematics, 117(2002):41-50.
[11] Goles E, Martinez S. *Complex Systems.* Kluwer, Dordrecht, 2001.
[12] C. Langton. Studying Artificial Life with Cellular Automata. Physica D 22(1986):120-149.
[13] J. Machta, K. Moriarty, R. Greenlaw. Parallel Algorithm and Dynamic Exponent for Diffusion-limited Agreggation. Physical Review E55, 6211, 1997.
[14] C. Mejia. El modelo de pilas de arena sobre grafos dirijidos y algo de complejidad. Revista integracion: temas de Matematicas, 24(2;2006):101-116.
[15] C. Moore. Majority-voting Cellular Automata, Ising Dynamics And P-completeness. Journal of statistical physics 88(1997):795-805.
[16] C. Papadimitriou. *Computational Complexity.* Addison wesley, Reading MA, 1994.
[17] V. Priezzhev, D. Dhar, A. Dhar, S. Krishnamurthy. Eulerian Walkers as a Model of Self-organized Criticality. Physical Review Letters, 77:5079-5082, 1996.
[18] W Ruzzo. On Uniform Circuit Complexity. Journal of Computer and Systems Sciences. 22 (1981):365-383.
[19] K. Sutner. On the computational Complexity of Finite Cellular Automata. Journal of Computing and Systems Sciences, 50(1):87-102, 1995.
[20] C. Thompson. *Mathematical Statistical Mechanics.* Princenton univ. press, Princenton NY, 1972.
[21] E. Toumpakari. *On the abelain sandpile model.* Ph.D. Thesis, Universidad de Chicago, 2005.
[22] D. Welsh. *Complexity: Knots, Colourings and Counting.* Cambridge University Press, Cambridge (UK), 1993.
[23] Wolfram S. *Cellular Automata And Complexity.* Wolfram research, U. champaign IL, 1994.

Universidad Industrial de Santander
E-mail address: juamonto@uis.edu.co, amontoyaa@googlemail.edu.co,