



INGENIERÍA DE SOFTWARE PARA VAGOS

(verificando software según la ley del menor esfuerzo)

Lic. Esteban Pavese

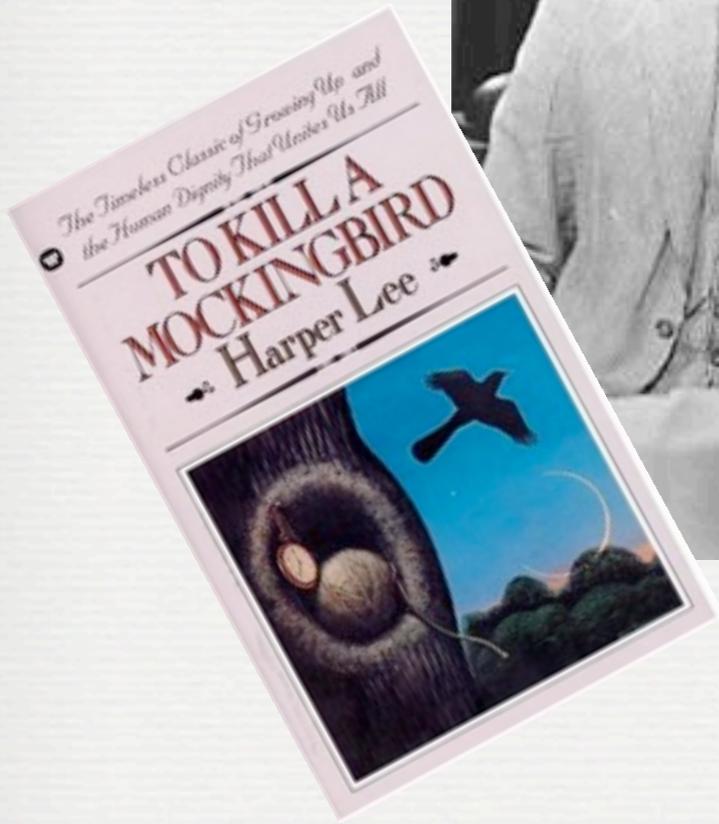
LA FUERZA DEL TRABAJO

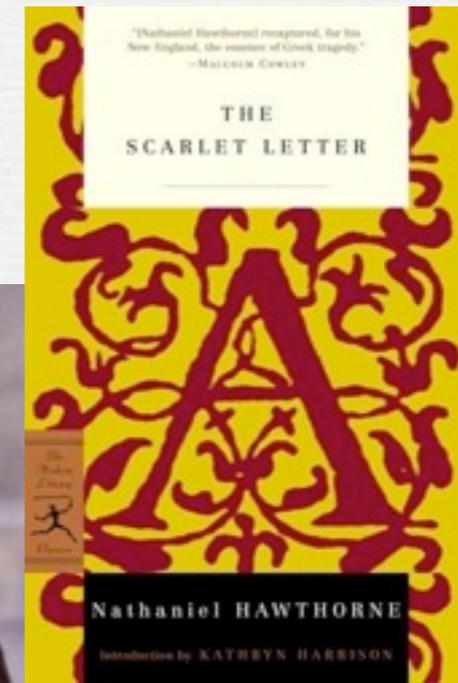
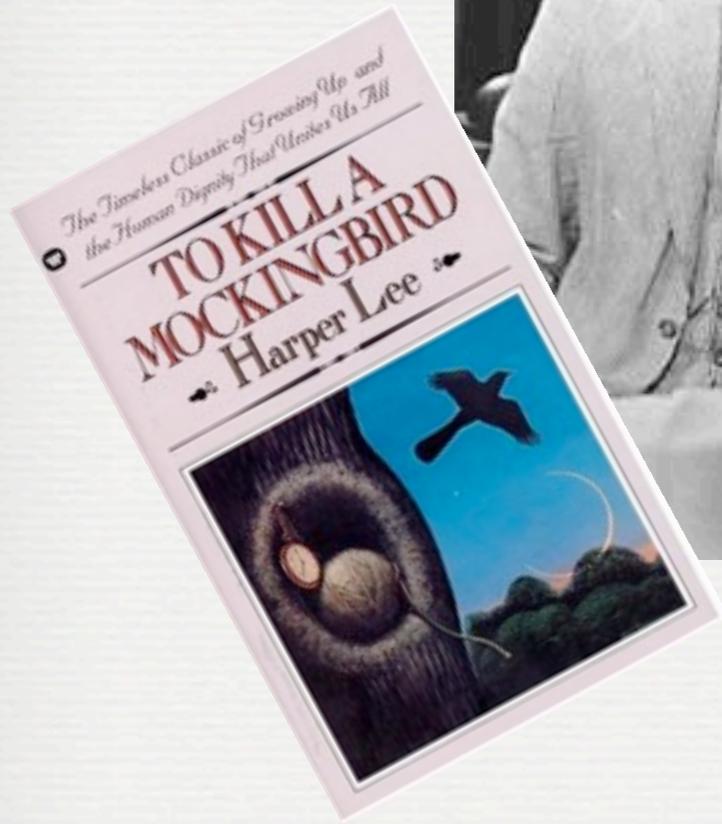


INGENIERÍA DE SOFTWARE PARA VAGOS

(verificando software según la ley del menor esfuerzo)

Lic. Esteban Pavese





QUÉ HACEMOS EN LaFHIS

Desarrollamos **procesos** y **herramientas** que asistan a los desarrolladores a lograr **software sin bugs**



QUÉ HACEMOS EN LaFHIS

Desarrollamos procesos y herramientas que asistan a los desarrolladores a lograr software ~~sin bugs~~
con menos bugs



QUÉ HACEMOS EN LaFHIS

Nos interesa asegurar que el software funciona “bien”

- Ante un pedido P , el software eventualmente responde Q
- Si dos procesos A y B utilizan un recurso exclusivo X , nunca lo hacen al mismo tiempo, y eventualmente ambos logran usarlo
- Siempre que se abre una conexión al servidor, se cierra posteriormente

QUÉ HACEMOS EN LaFHIS

También nos interesa dejarlo portarse “mal” (un poco)

- El servidor puede colgarse, pero no por más de 2 horas
- El router tiene que poder enviar al menos 10^6 paquetes entre fallas
- La probabilidad de que los frenos no se activen tras 1 segundo de presionados debe ser menor a 10^{-4}

TESTING

IDEA:

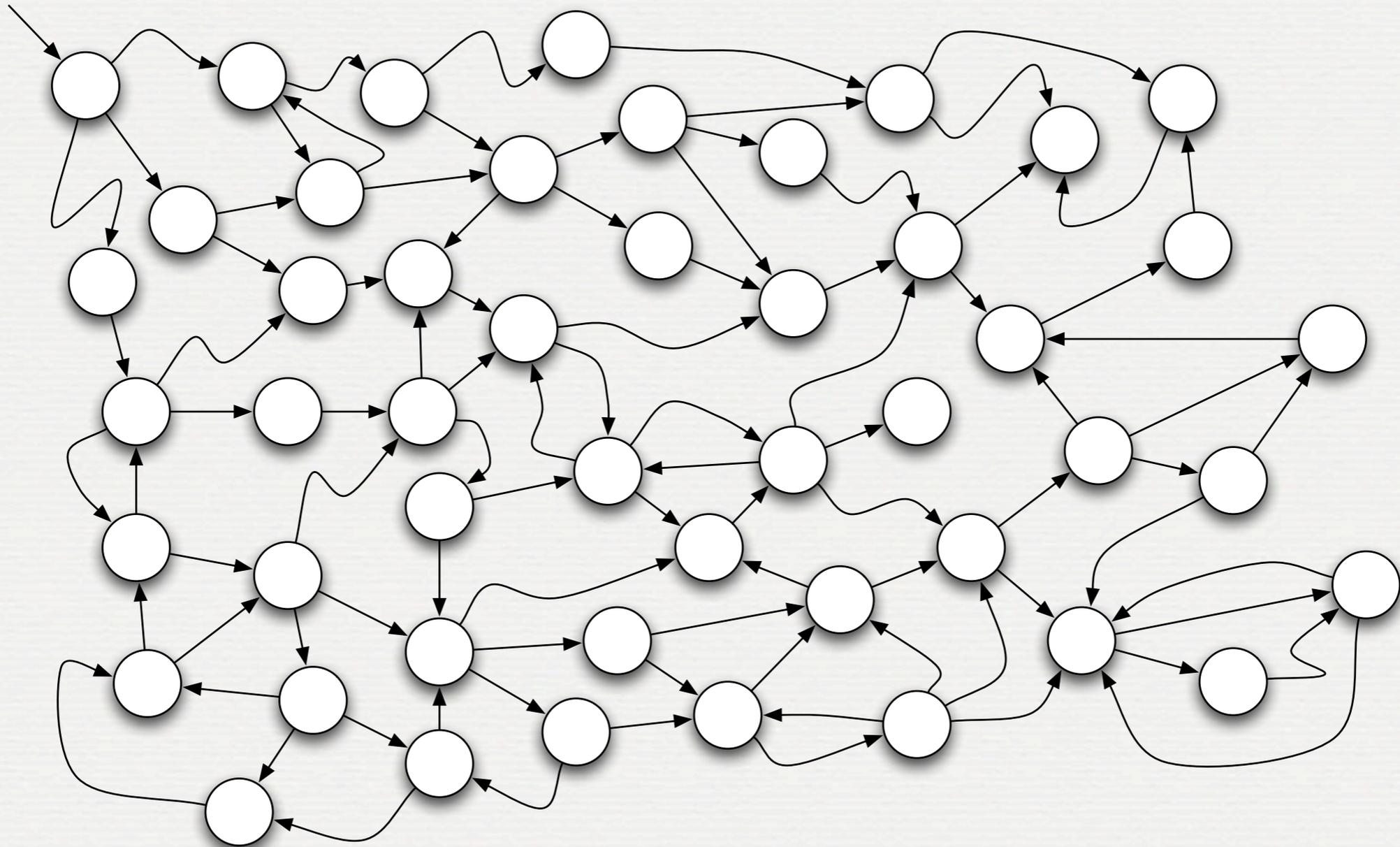
Inferencia de la correctitud del software o propiedad medible a partir de **muestras**



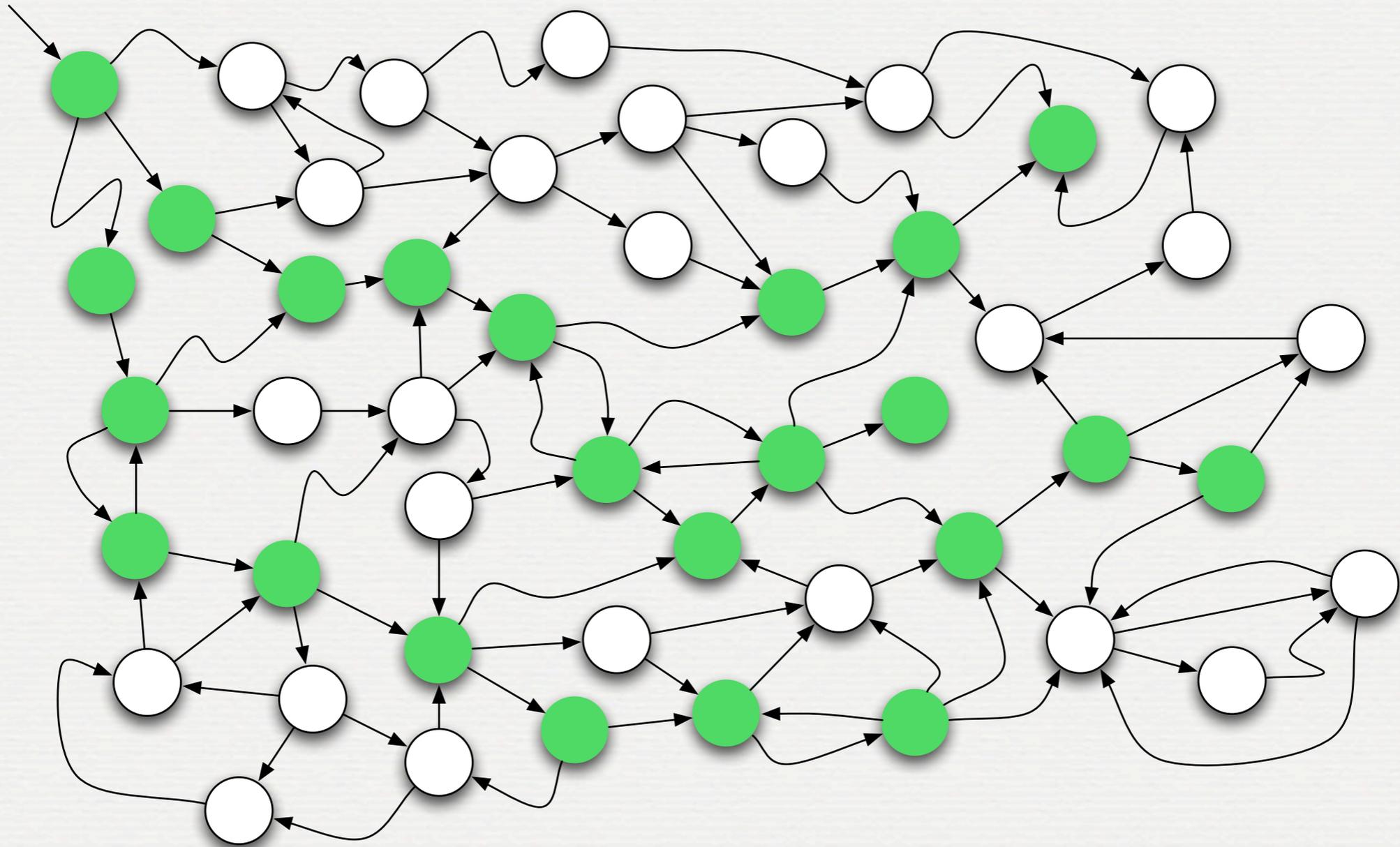
El proceso de test puede ser tan simple o complicado como uno quiera...

Pero **no puede** garantizar la ausencia de fallas, y las propiedades medibles son problemáticas

TESTING



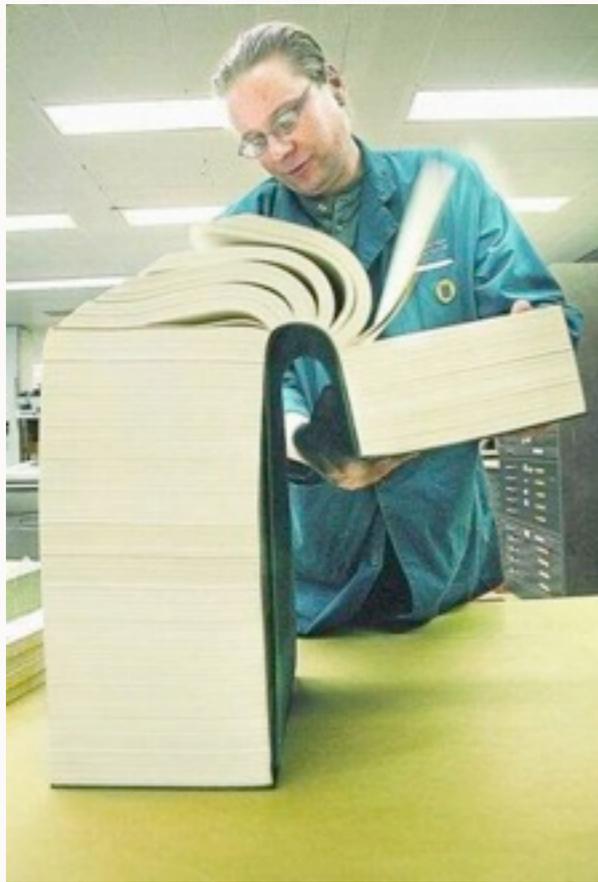
TESTING



MODEL CHECKING

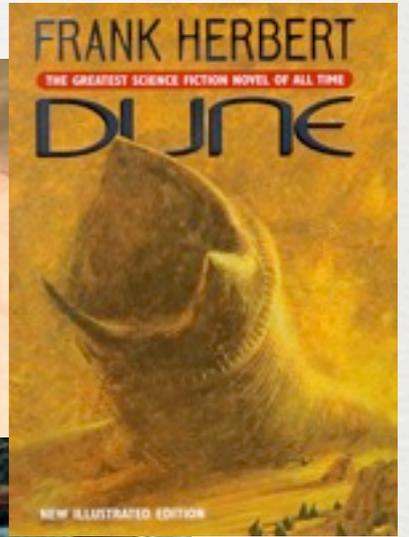
IDEA:

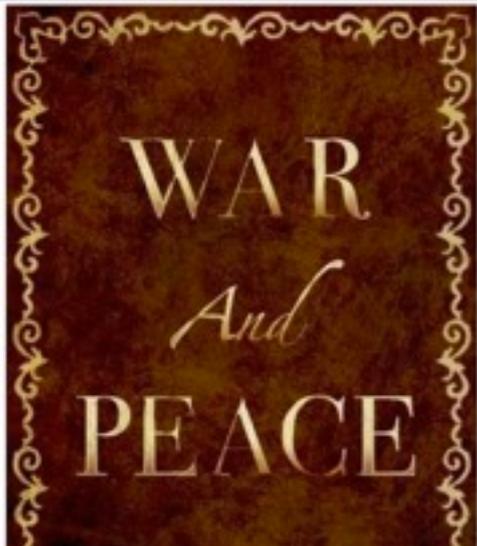
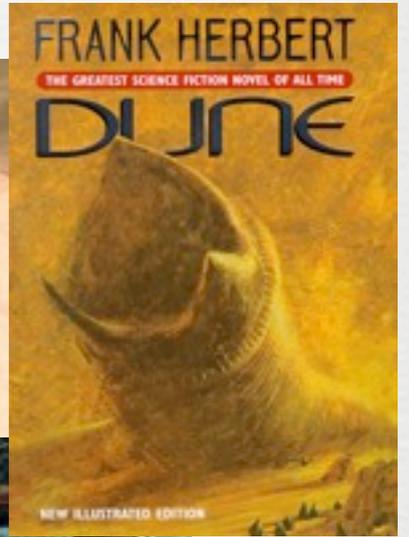
Exploración **exhaustiva** de los posibles estados del sistema



Garantiza satisfacción (o no) de las propiedades chequeadas

Pero puede ser **muy costoso** en tiempo y memoria...





CUÁL ES EL IDEAL?



Ninguna de las dos es ideal, pero...

	Eficiente	Completo	Garantías	Vago-friendly
Testing				
Model checking				

CUÁL ES EL IDEAL?



Ninguna de las dos es ideal, pero...

	Eficiente	Completo	Garantías	Vago-friendly
Testing				
Model checking				

CUÁL ES EL IDEAL?



Ninguna de las dos es ideal, pero...

	Eficiente	Completo	Garantías	Vago-friendly
Testing				
Model checking				

CUÁL ES EL IDEAL?



Ninguna de las dos es ideal, pero...

	Eficiente	Completo	Garantías	Vago-friendly
Testing				
Model checking				

CUÁL ES EL IDEAL?

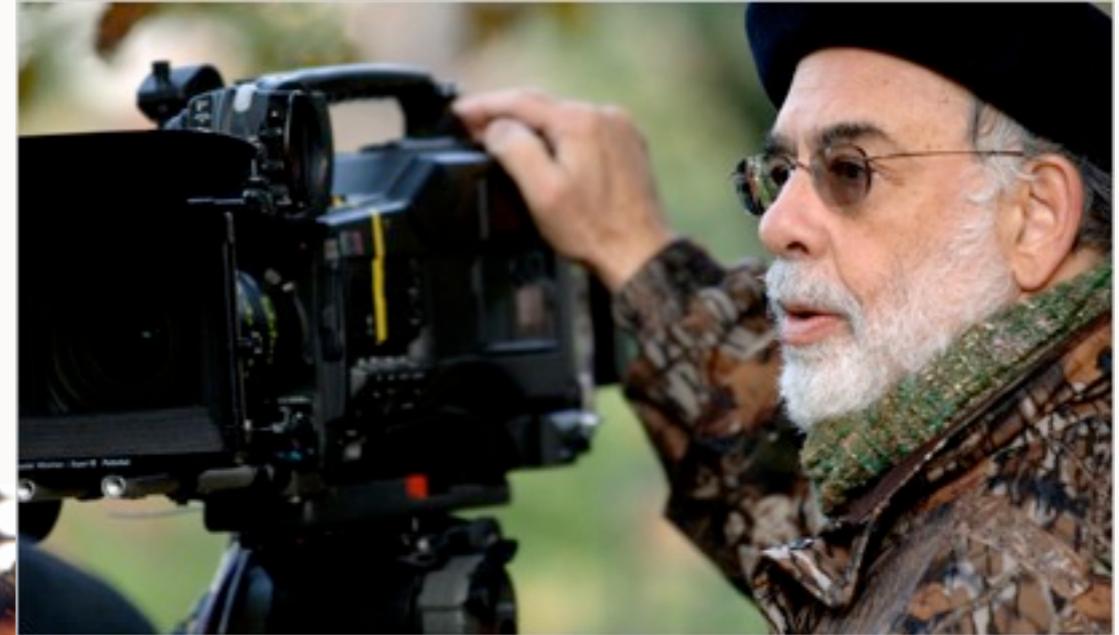
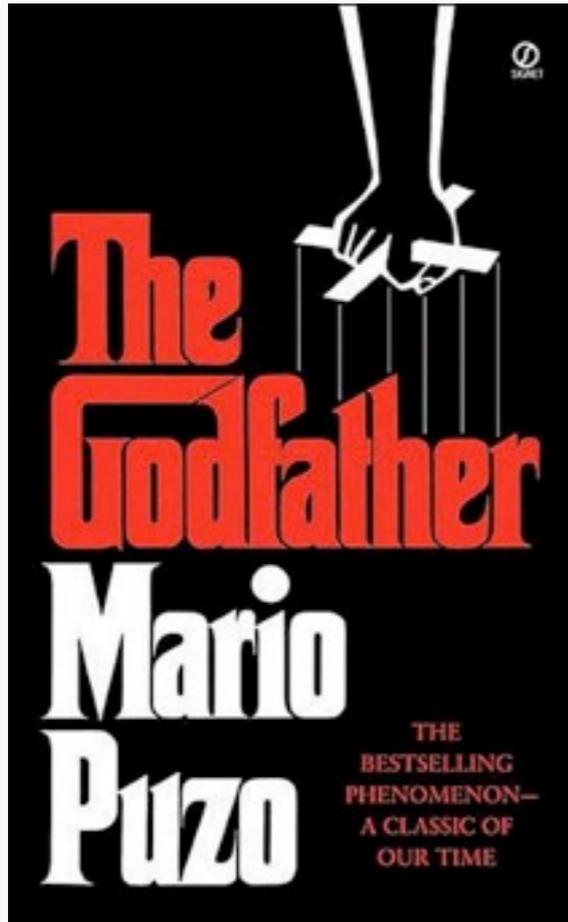


Ninguna de las dos es ideal, pero...

	Eficiente	Completo	Garantías	Vago-friendly
Testing				
Model checking				

¿Podemos quedarnos con lo mejor de cada uno?

CUÁL ES EL IDEAL?



PENSANDO AL REVÉS...

¿Cómo se adapta un libro a una película?

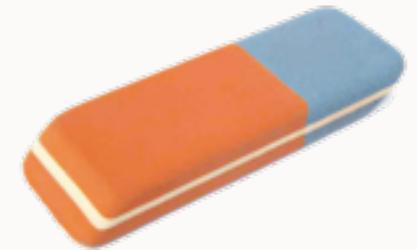


Parece simple...



Mantener lo importante...

y descartar lo que no lo es



¿*Cuánto* es lo importante?

¿*Dónde* está lo que queremos mantener?

JURAN X PARETO

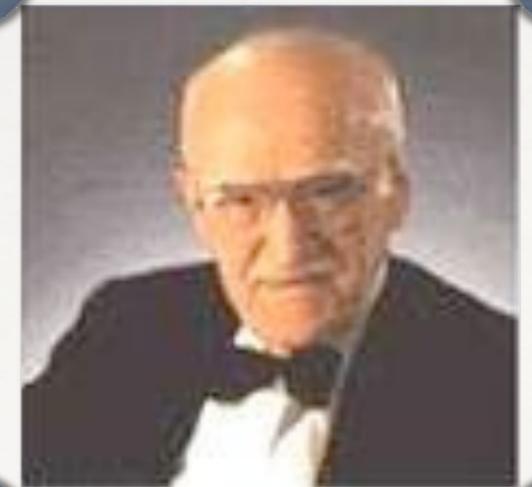


Vilfredo Pareto (Italia, ~1920)

“El 80% de la tierra de Italia está en manos del 20% de la población”

Joseph Juran (EE.UU., ~1960)

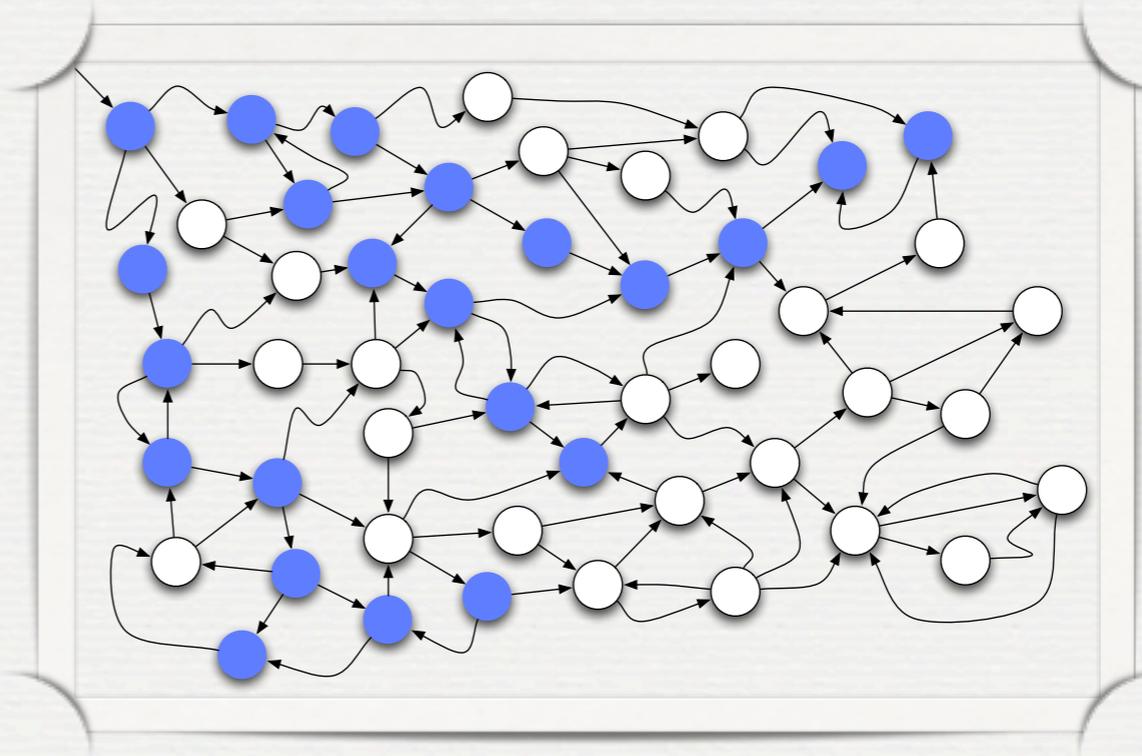
Aplica las ideas de Pareto a gestión de calidad



¿CÓMO SE TRADUCE ESTO EN IS?

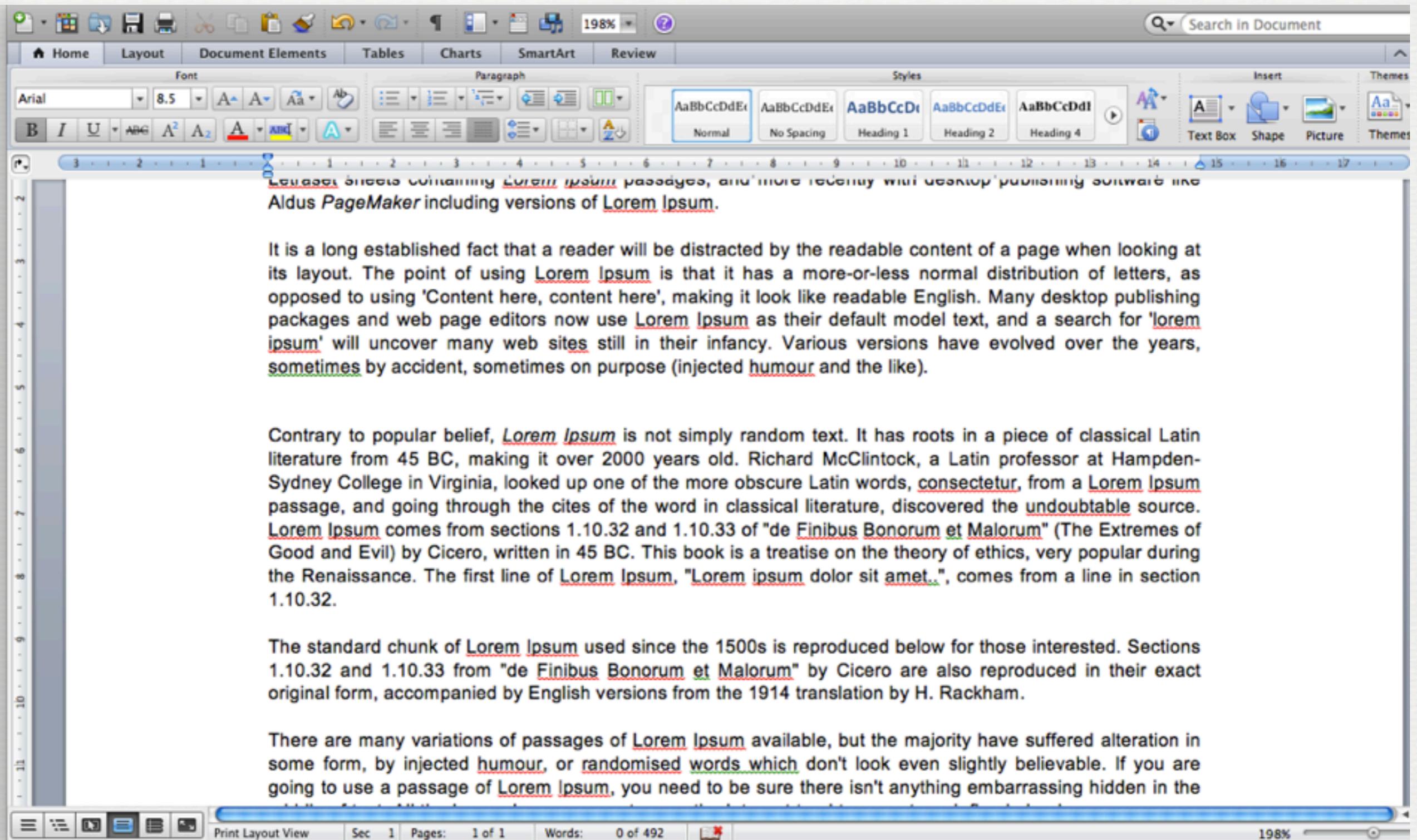


¿Existe este 20% importante del software?



¿Y quién determina cuál es ese 20% más importante?

PARETO EN MS WORD

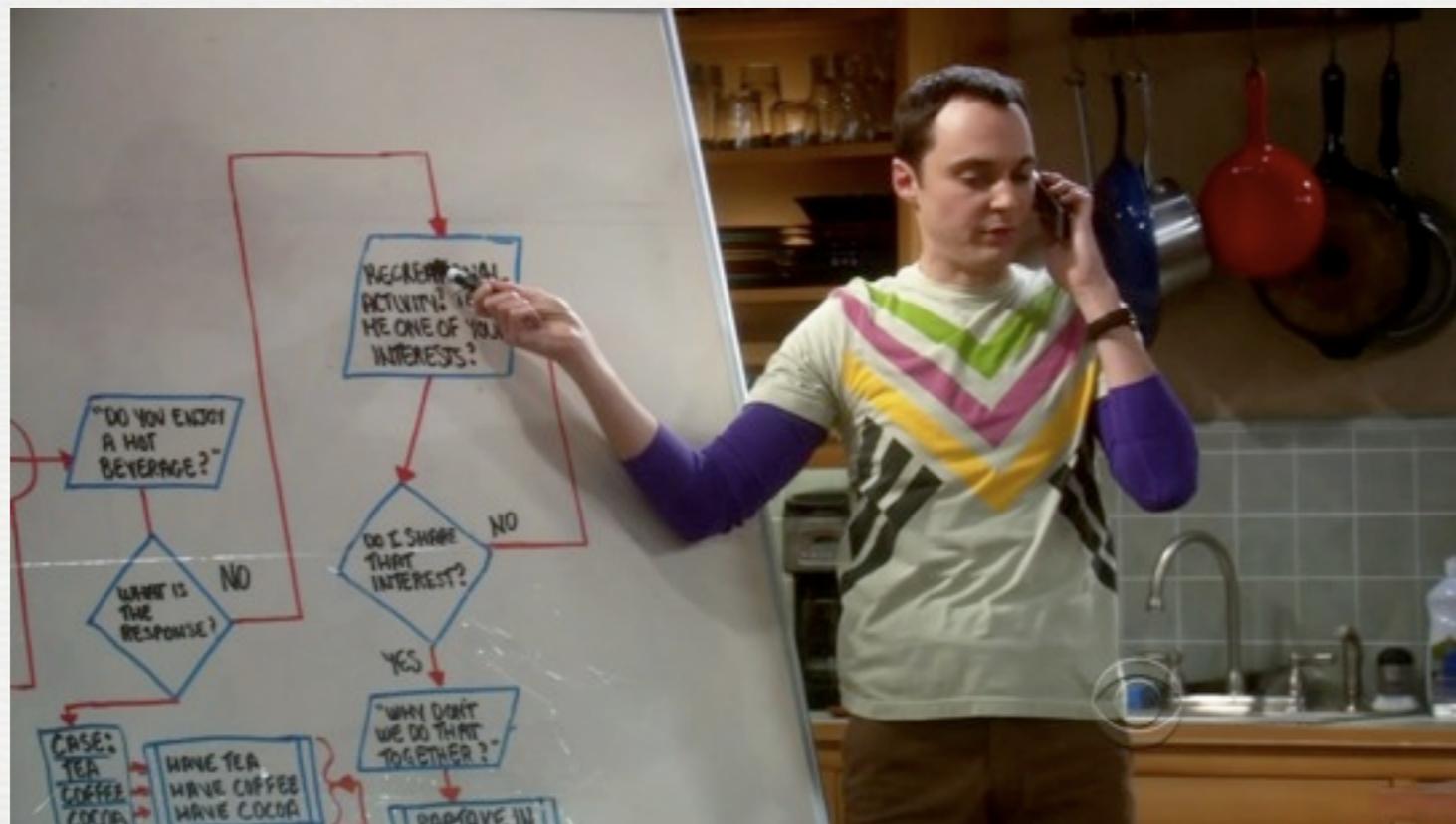


PERFILES DE USUARIO



Modelo de comportamiento del *usuario* del software

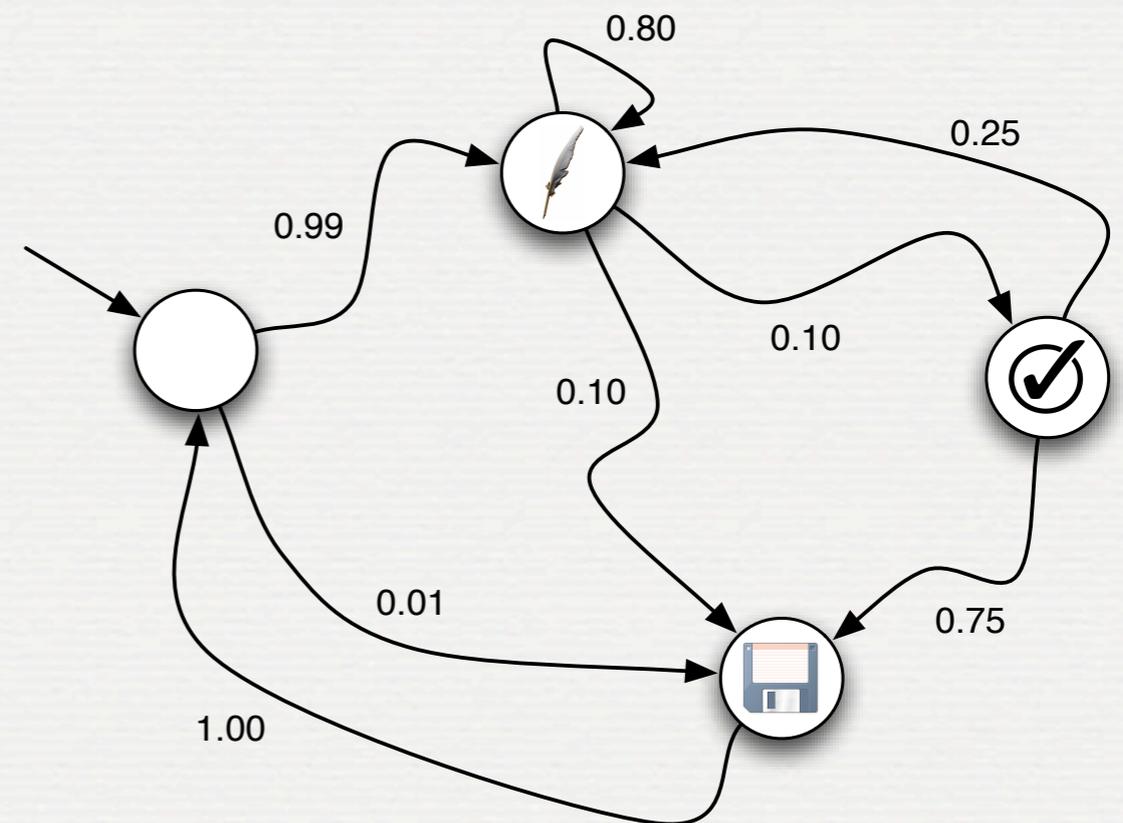
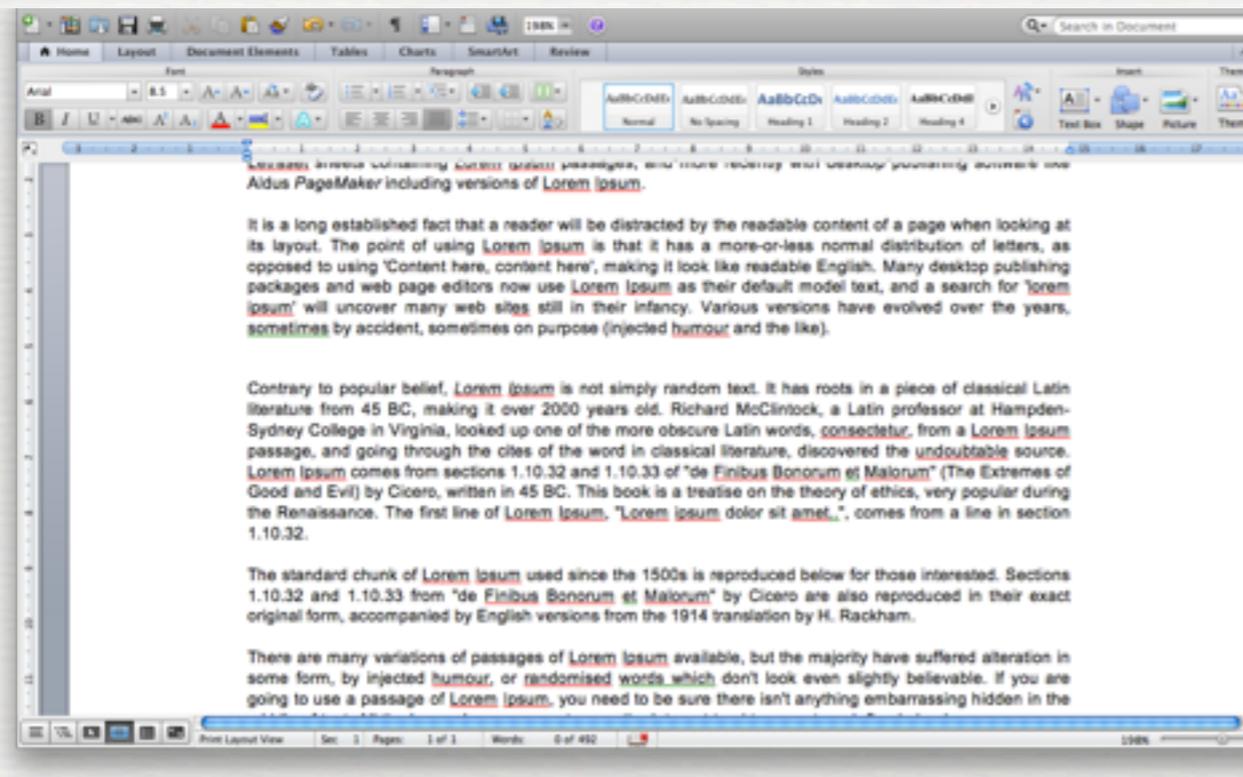
Especifica, para cada instante, lo que el usuario es propenso a hacer



PERFILES DE USUARIO



Formalmente ...

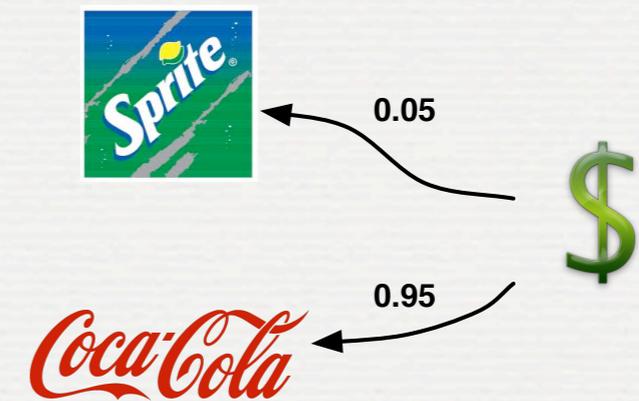
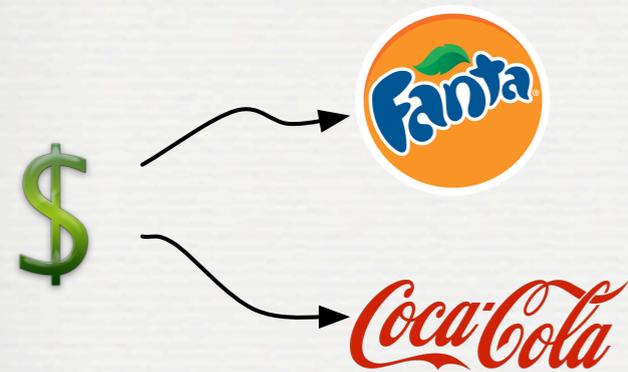


para simplificar las cosas, estos modelos son *sin memoria* (ie, Markovianos)

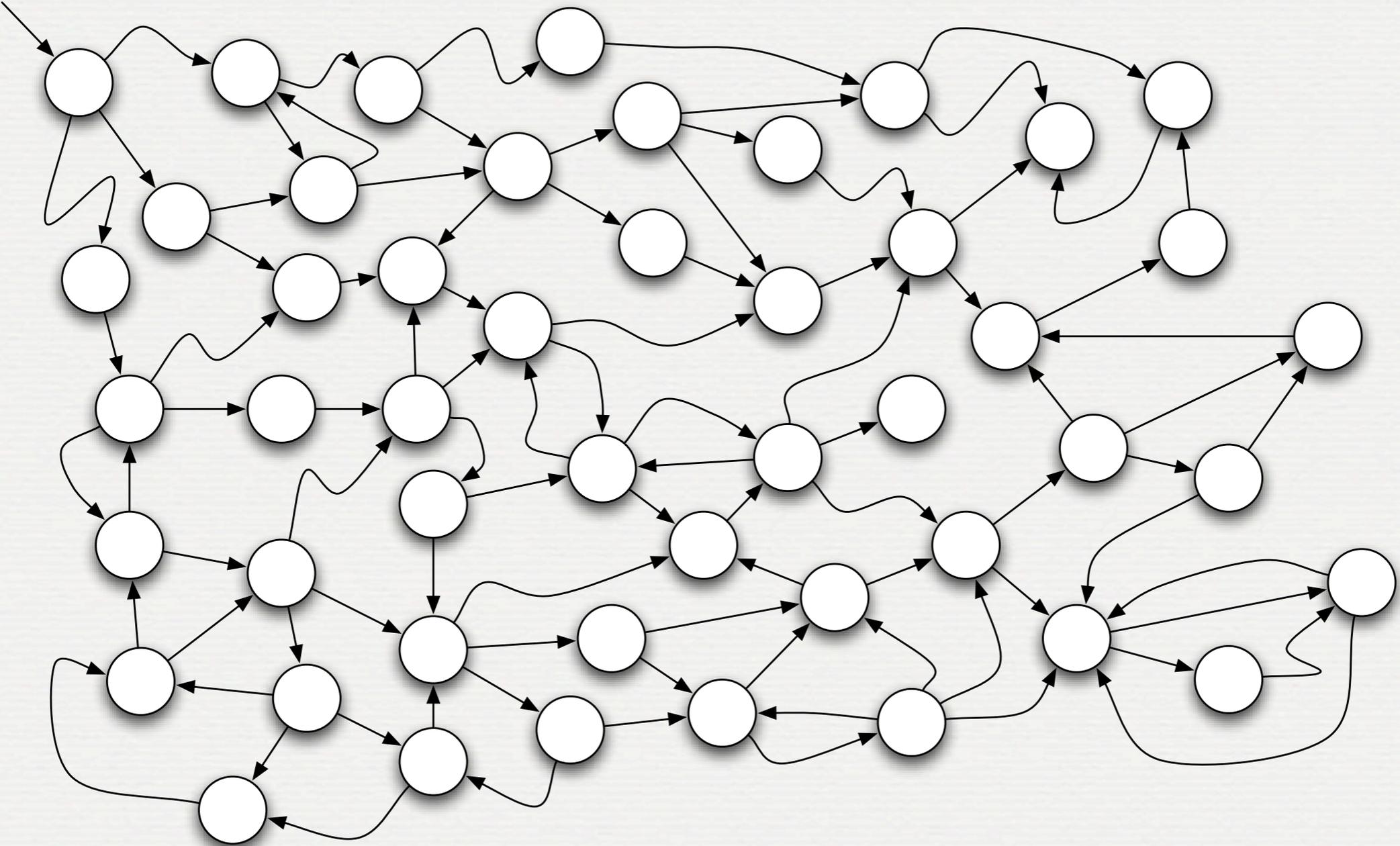
PERFILES DE USUARIO



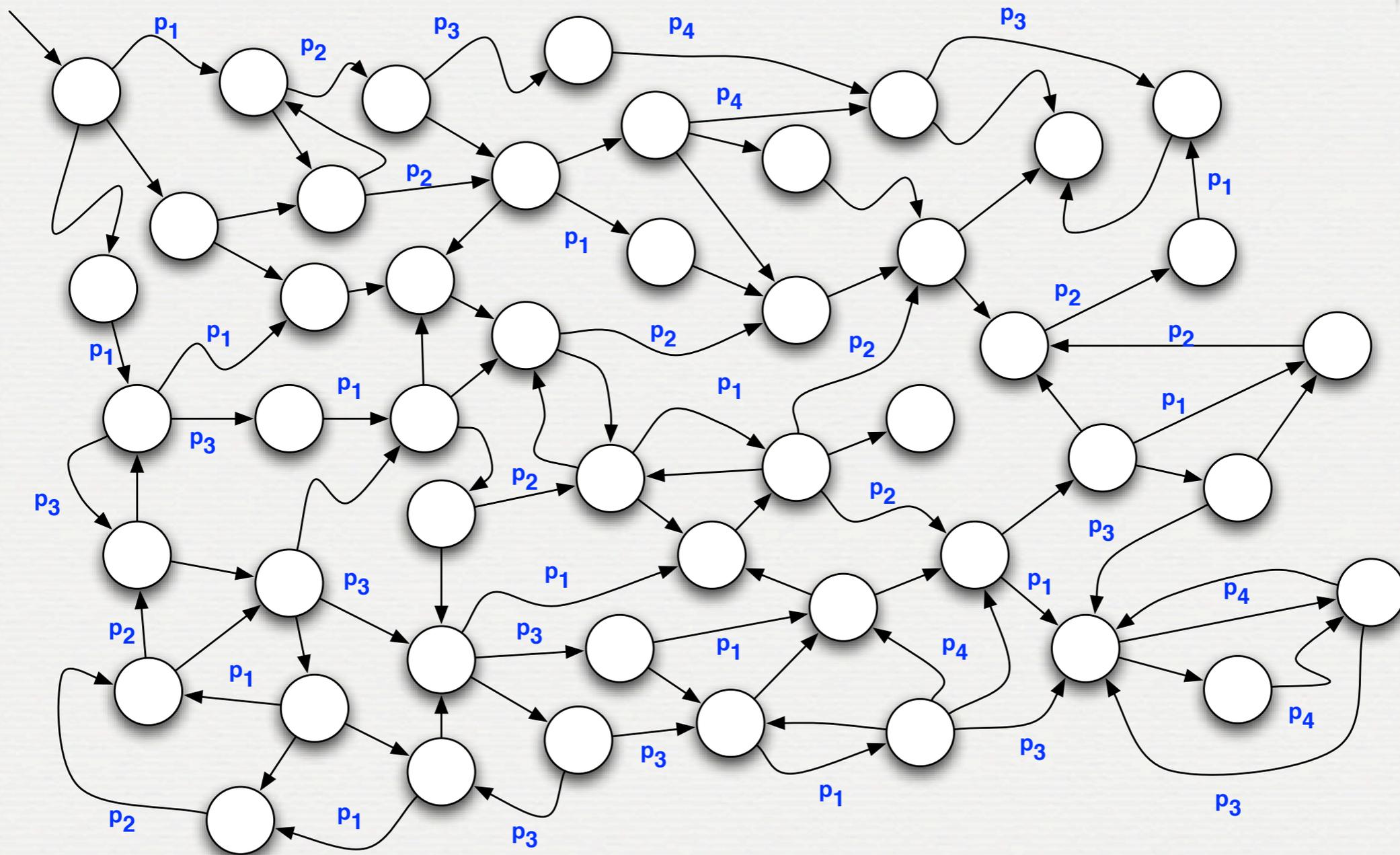
Aún así, hay que tener cuidado ...



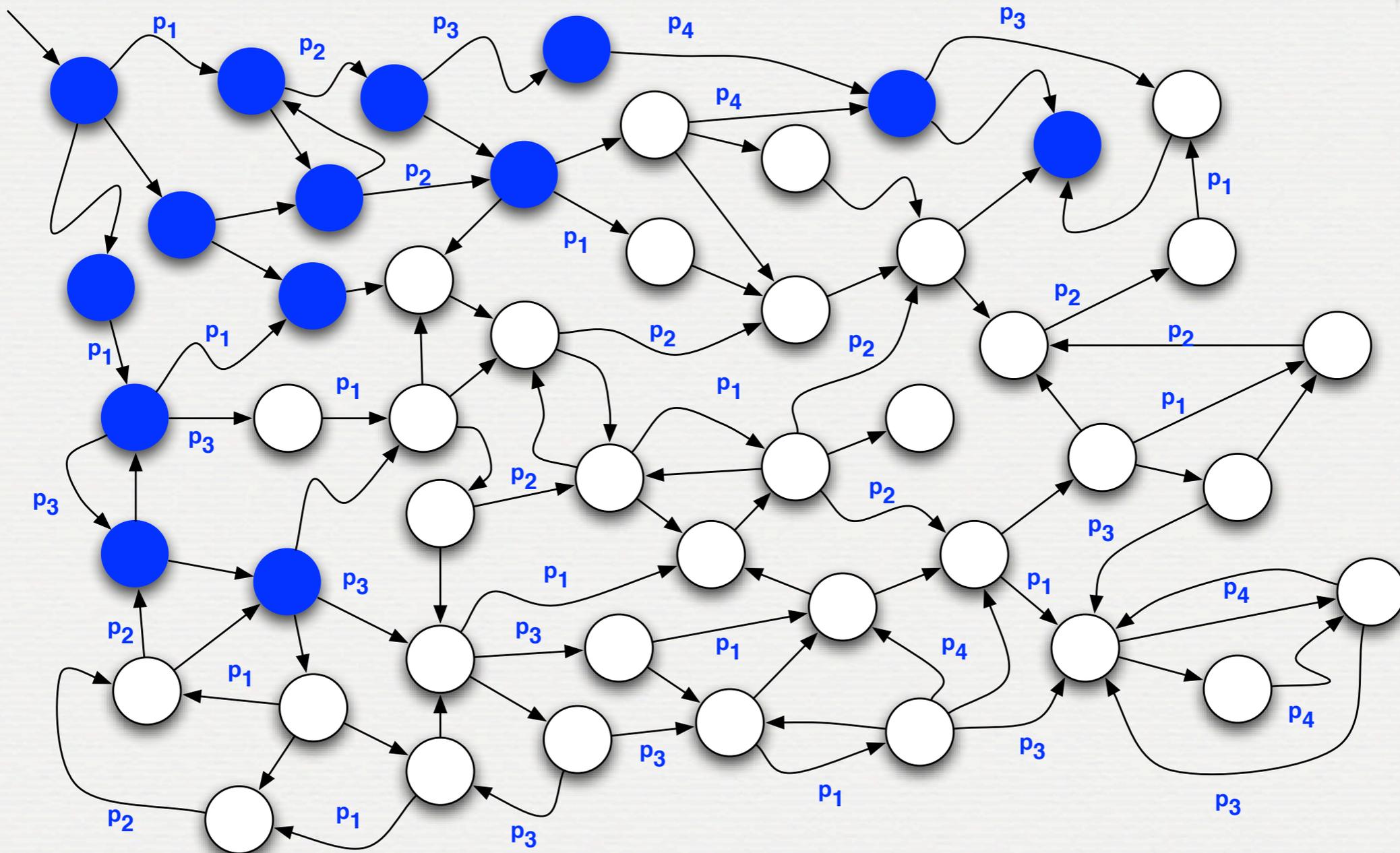
BUSCANDO EL 80/20



BUSCANDO EL 80/20



BUSCANDO EL 80/20



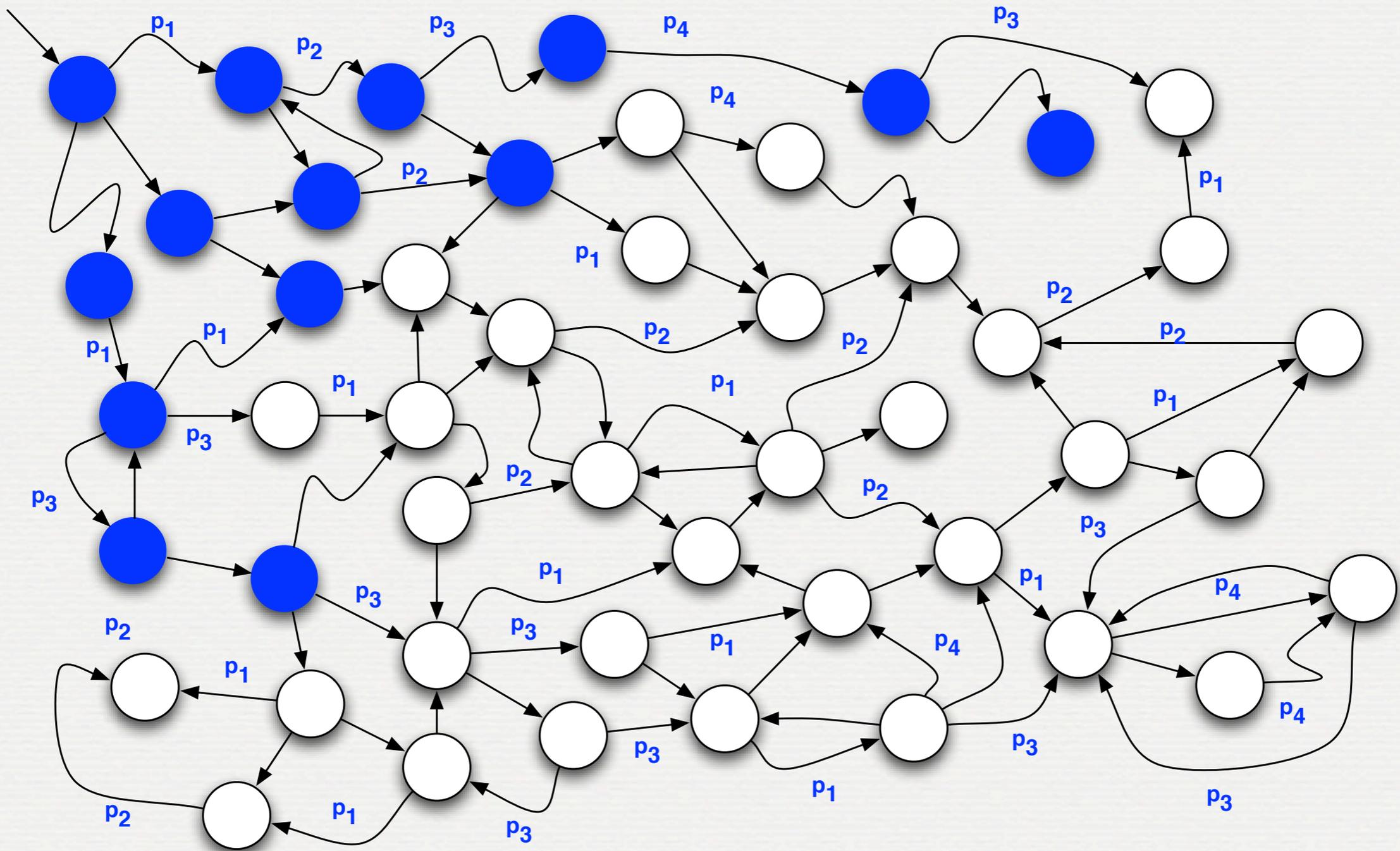


PROBLEM?

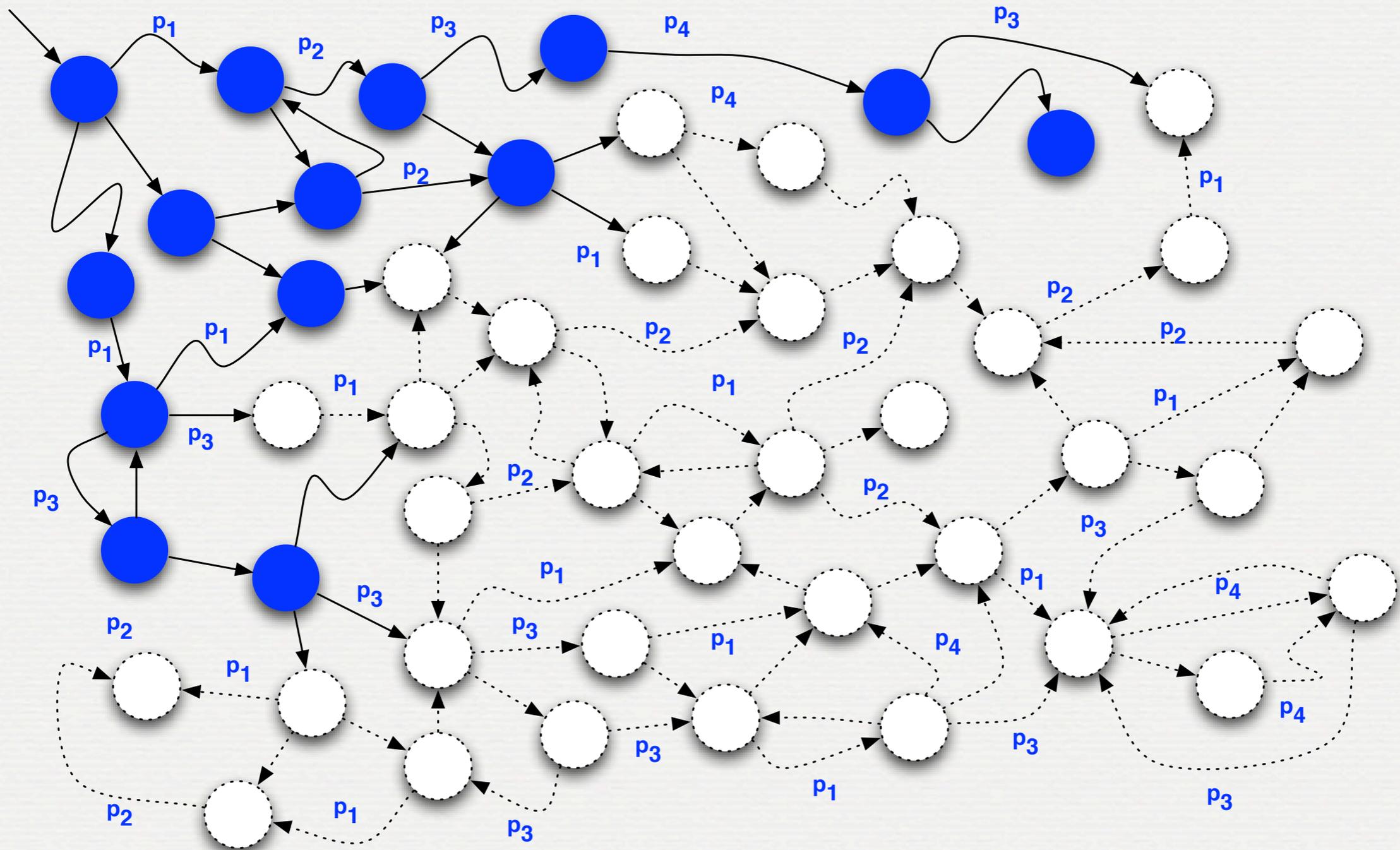
Este enfoque en la práctica tiene un par de problemas...

- ¿20-80? ¿10-85? ¿Cómo saberlo? ¿Con qué criterio?
- El enfoque presupone que podemos anotar el espacio completo de estados
 - ¿Realmente podemos?
 - y además, ¿no era lo que queríamos evitar?

PENSANDO LO AL REVÉS... (OTRA VEZ)



PENSANDOLO AL REVÉS... (OTRA VEZ)



¿QUÉ DIRECTOR QUEREMOS?



DFS (Depth first search)

Explora en mucho detalle, pero de muy pocos escenarios

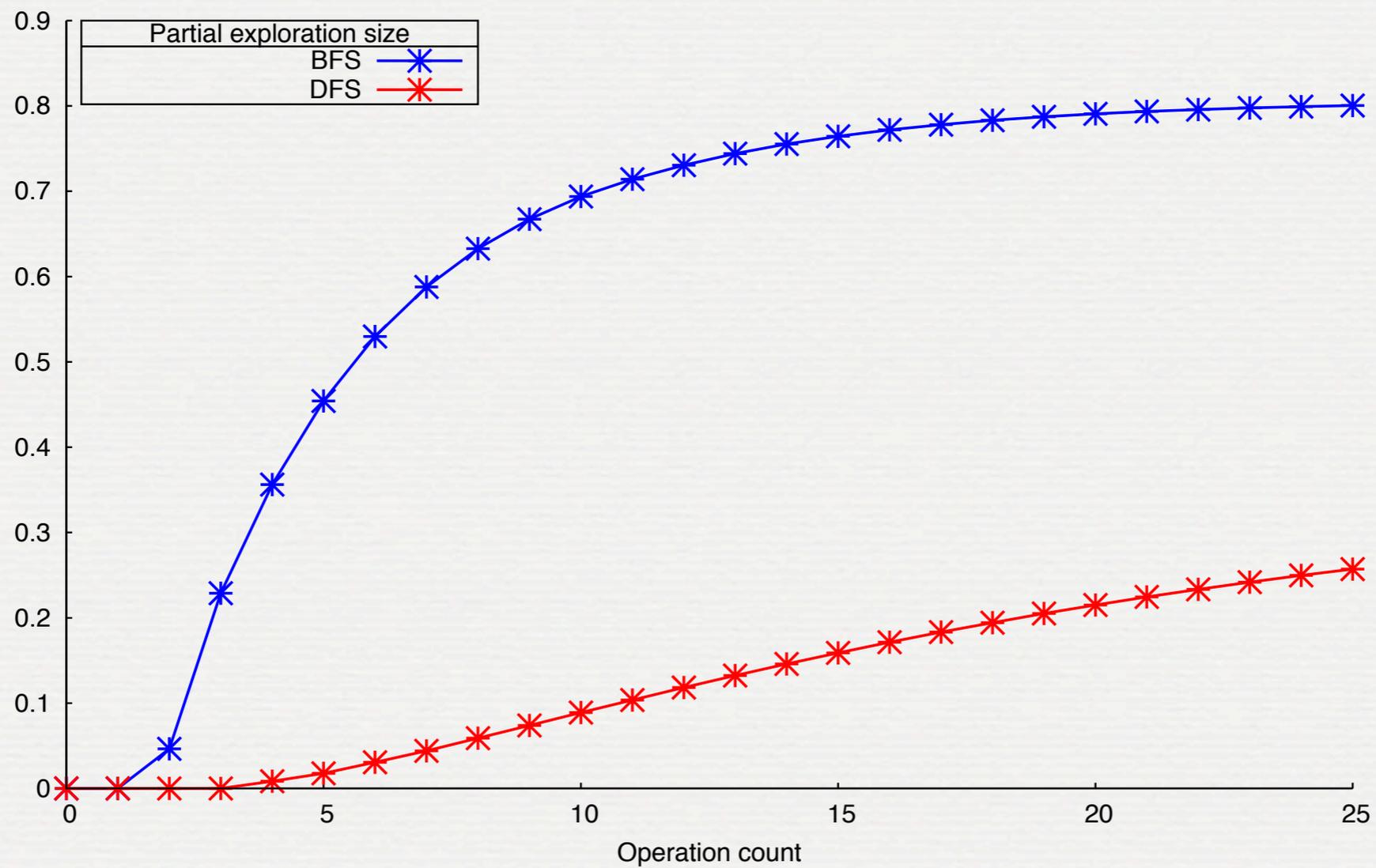
En la práctica, la calidad de la exploración resulta similar a la de realizar (random) testing

BFS (Breadth first search)

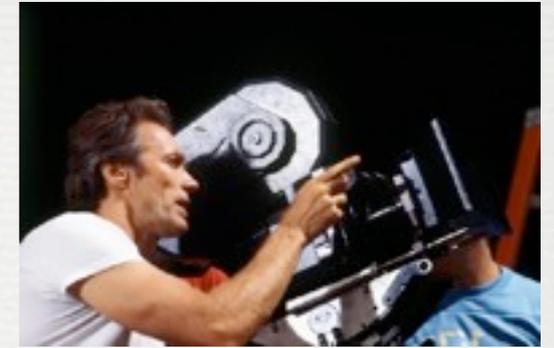
Abarca poco de todos los escenarios

... pero en general la calidad de la exploración es **mucho mejor**

¿QUÉ DIRECTOR QUEREMOS?



¿QUÉ DIRECTOR QUEREMOS?



¿Puede hacerse mejor que con BFS?

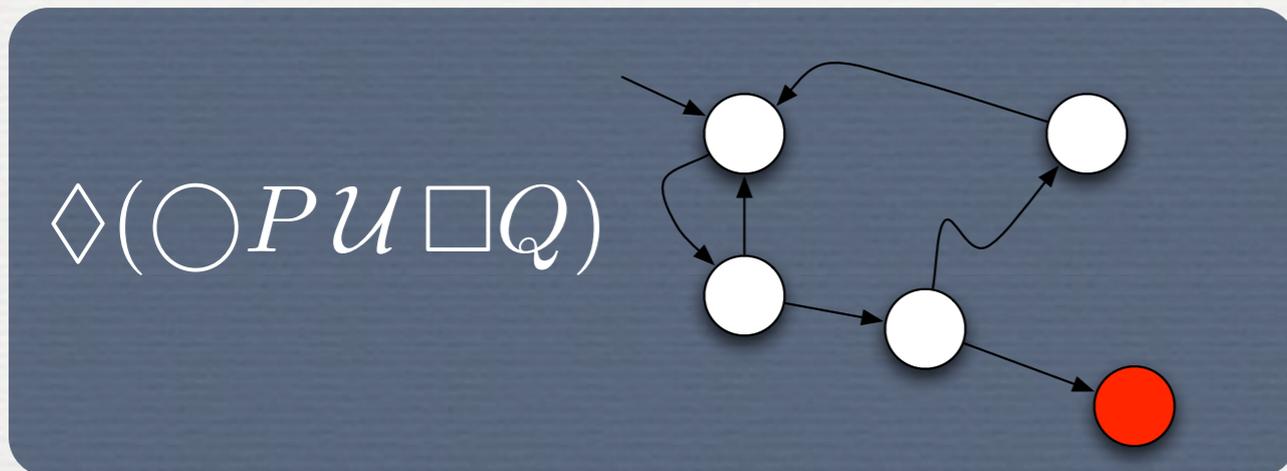
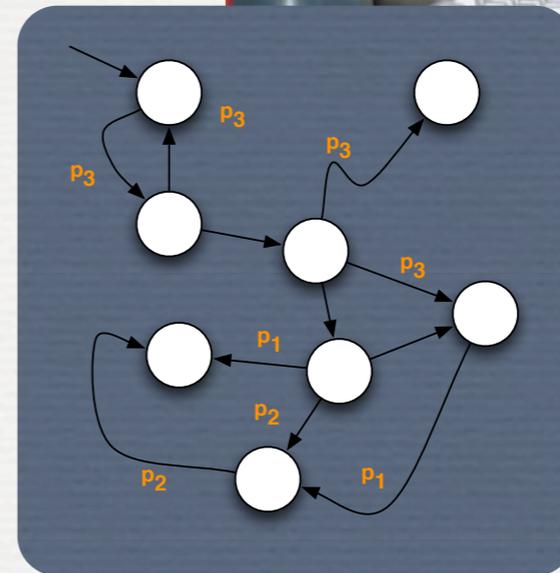
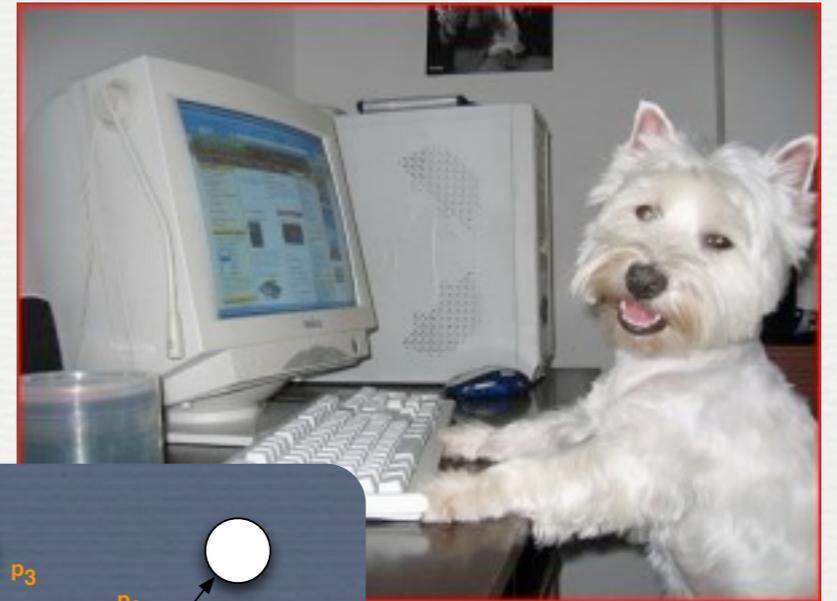
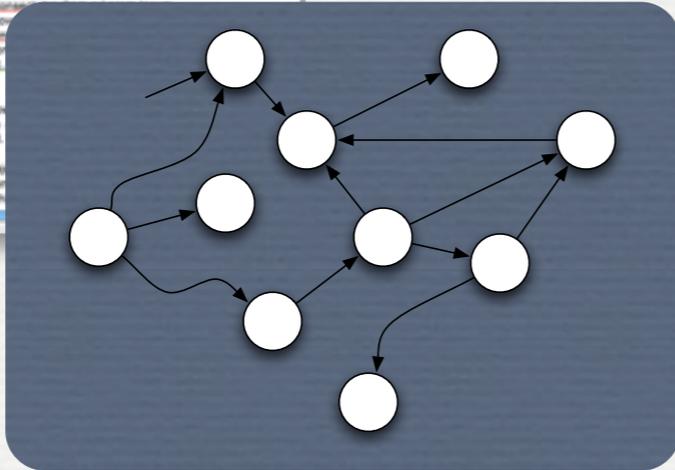
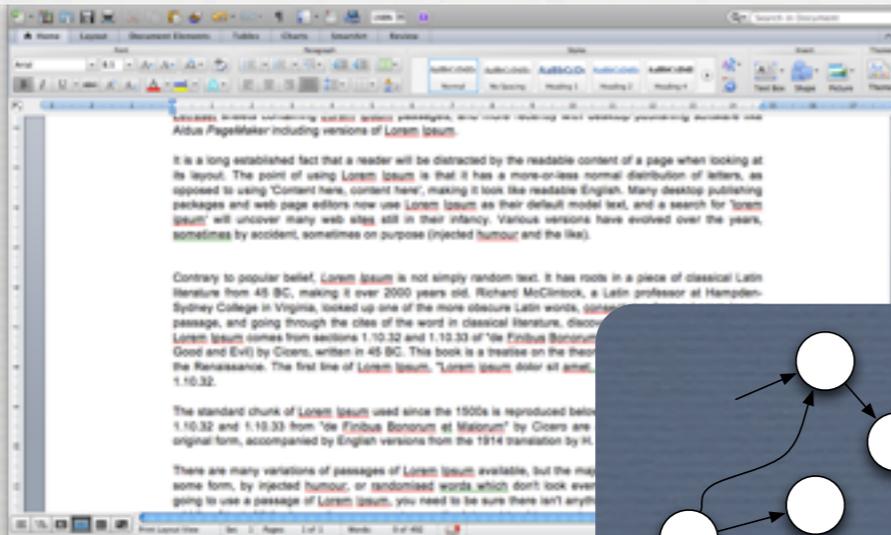
Incorporamos más información (no probabilística) del perfil

Invariante de uso :

Especificación de los estados en que *se conjetura* que los usuarios pasarán más tiempo



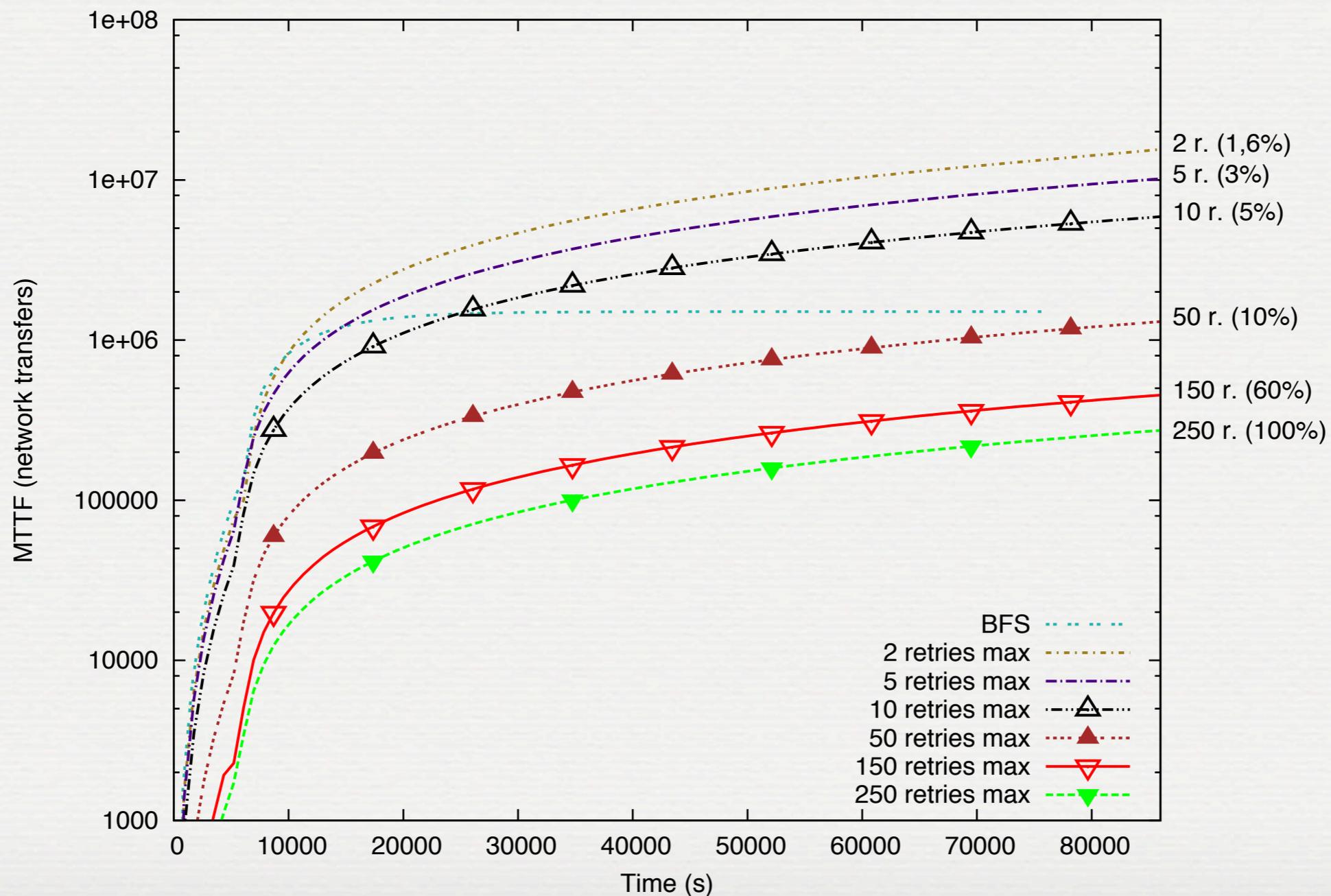
CÓMO FUNCIONA ESTO



THE GOOD...



Puede funcionar *mucho* (pero *mucho*) mejor que BFS



... THE BAD ...

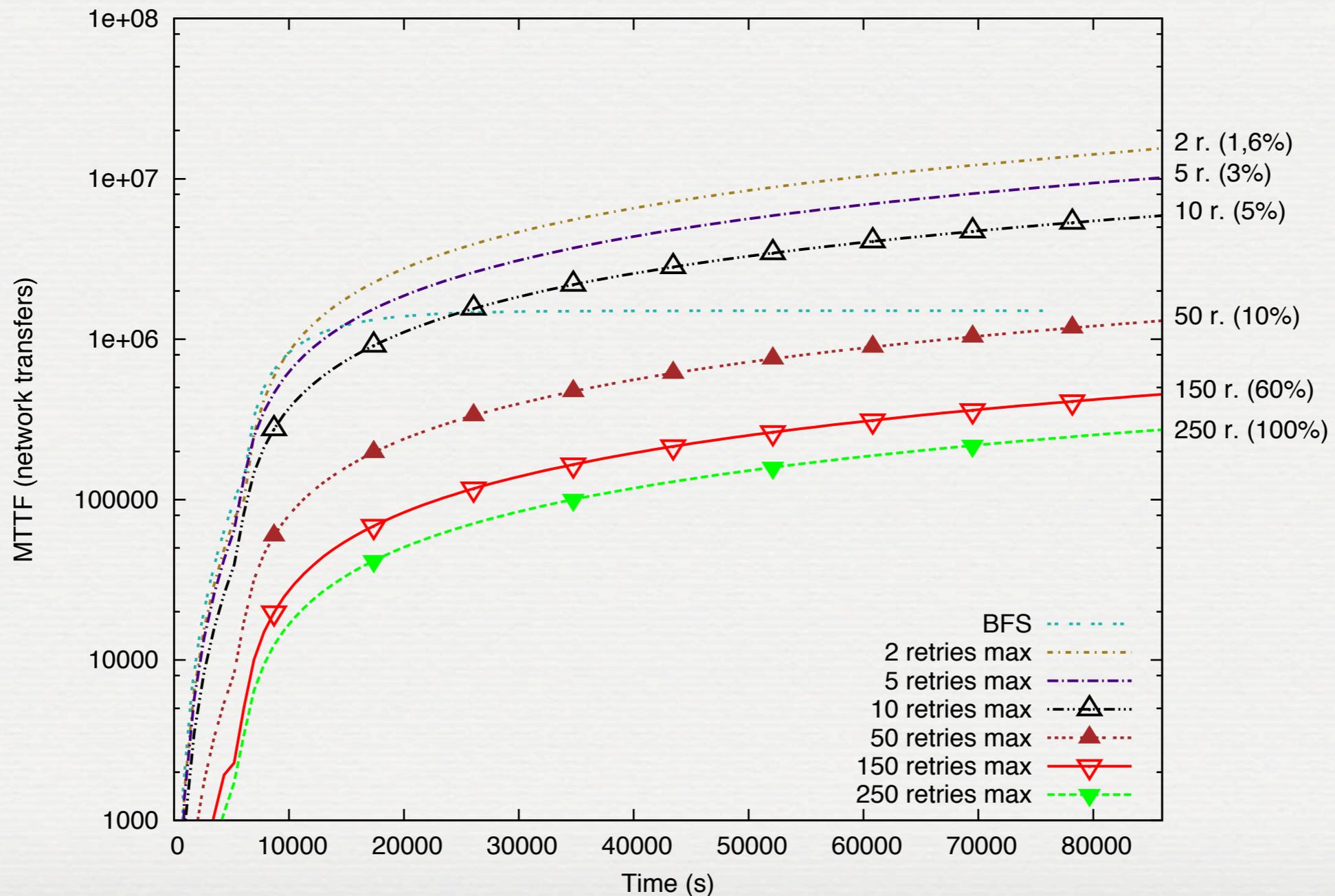


... pero también puede andar peor ...

... THE BAD ...



... pero también puede andar peor ...

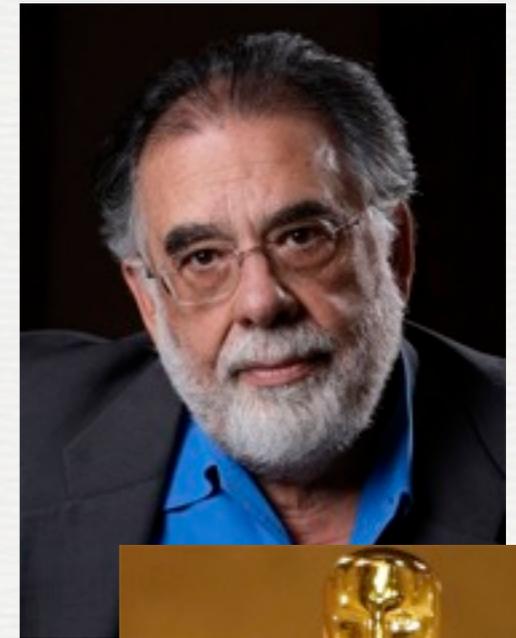


...AND THE UGLY

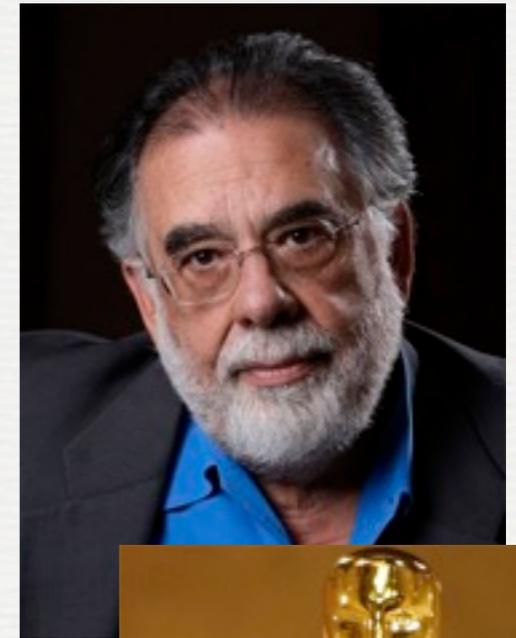


... y no tenemos una forma segura de predecirlo!

DÓNDE ESTAMOS



DÓNDE ESTAMOS



¿CÓMO OBTENER LOS PERFILES?

Analizar logs:

- ▶ buscando comportamiento repetitivo (invariantes!)
- ▶ entendiendo el comportamiento global del usuario (perfiles de uso!)
 - ▶ comportamiento no markoviano?



¿CÓMO DIRIGIR MEJOR LA BÚSQUEDA?



▶ encontrar la mejor exploración para ciertas restricciones (tiempo / espacio)

▶ ¿se podrá hacer al tiempo que vamos avanzando?

¿PODEMOS HACERLO SOBRE CÓDIGO?



Los modelos sobre los que trabajamos son bastante abstractos ...

Nos gustaría trabajar con código real ... Java, .NET, etc.



ACERCA DE LAFHIS



¡Amplia experiencia!



¡Llame YA!



LA FUERZA DEL TRABAJO



INGENIERÍA DE SOFTWARE PARA VAGOS

(verificando software según la ley del menor esfuerzo)

Lic. Esteban Pavese