**Parámetros DAGsim:**

**no\_of\_transactions**: define el número de transacciones que se van a procesar durante la simulación. Influirá en el tamaño y complejidad del TANGLE que se genera.

**no\_of\_agents**: define el número de nodos en la red que participan en la emisión y validación de transacciones.

**Lambda (λ):** define la frecuencia con la que las transacciones entran al sistema. A un λ alto genera transacciones más rápido.

Ejemplo con lambda = 1 vs lambda = 10:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Podemos ver que con lambda = 10 Las transacciones se generan rápidamente y tienen múltiples oportunidades para conectarse a varias transacciones anteriores. En el caso de lambda = 1 las transacciones ocurren menos frecuentemente, lo que permite que el Tangle se desarrolle más lentamente y con menos opciones para cada nueva transacción de conectarse a múltiples transacciones anteriores.

Según el artículo sigue una distribución de Poisson cito "Los tiempos de llegada de las transacciones entrantes se muestrean a partir de la exponencial según la tasa de transacciones entrantes, que sigue un proceso de Poisson."  La lambda define cuántas transacciones se espera que lleguen por unidad de tiempo, por ejemplo si es 10, se esperan 10 transacciones por unidad de tiempo, al ser Poisson algunas veces pueden llegar + o - transacciones de lo esperado.

**Alpha (α)**: sirve para la aleatoriedad en el algoritmo de selección de puntas. Con un Alpha alto parece que es más determinista y con un Alpha bajo parece que es más aleatorio.   
  
Ejemplo con Alpha = 0.1 vs Alpha = 0.5

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Imagen en blanco y negro

Descripción generada automáticamente con confianza baja

La relación entre el método de selección de puntas ("tip selection") y el parámetro alpha (*α*) en la simulación del Tangle de IOTA es crucial, especialmente cuando el algoritmo de selección de puntas es ponderado ("weighted"):

* Random: No utiliza el parámetro α, ya que simplemente elige puntas de manera aleatoria entre las transacciones visibles y válidas.
* Unweighted MCMC (Monte Carlo Markov Chain): Este método también ignora el parámetro α porque realiza caminatas aleatorias sin ponderar las transacciones por su peso acumulativo.
* Weighted MCMC: Aquí es donde α entra en juego. En este método, α afecta directamente cómo se ponderan las transacciones durante la caminata aleatoria. Específicamente, se utiliza para ajustar la probabilidad de transición entre las transacciones en la caminata aleatoria, haciendo que la probabilidad de seleccionar ciertas puntas sea proporcional al peso acumulativo de estas, ajustado por α.

Por lo tanto, el Alpha es un factor de ajuste que modula la influencia del peso acumulativo de una transacción durante la selección de puntas en el algoritmo ponderado. Un α más alto incrementa la probabilidad de que las transacciones con mayor peso sean seleccionadas como puntas, lo cual puede favorecer una convergencia más rápida del consenso en la red, pero a la vez puede aumentar la centralización del grafico debido a que las transacciones más 'pesadas' dominan la selección de puntas.

**Latencia** : Se refiere al tiempo de retraso en la comunicación entre nodos. En el artículo establecen siempre latencia = 1

Ejemplo latencia 10 vs latencia 1

Gráfico

Descripción generada automáticamente

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Con latencia 10 se puede ver cómo , durante la mayor parte del tiempo simulado, el Agente 1 tiene un número muy bajo de tips, mientras que el Agente 0 no tiene tips directamente. Esto indica que la alta latencia puede estar impidiendo que las transacciones del Agente 1 se vean y seleccionen como tips por otros agentes de manera oportuna. Si nos fijamos en el Tangle las transacciones del agente 1 parecen ser dominantes, es decir, casi todas las transacciones confirman al menos 1 transacción del agente 0.

Con una latencia baja, el número de tips de ambos agentes varía más uniformemente a lo largo del tiempo. El Agente 1 muestra un incremento en el número de tips hacia la mitad y al final del período simulado, lo que sugiere que sus transacciones están siendo más frecuentemente vistas y seleccionadas por otros agentes y el Agente 0 también muestra un incremento más uniforme. Si ves ahora el Tangle parece que están contribuyendo ambos a la red y que transacciones del Agente 0 pueden confirmar del Agente 1 y viceversa.

La latencia representa el tiempo de retardo en la red para que una transacción sea visible para otros nodos o agentes. La latencia afecta cuando una transacción se considera "visible" para otros nodos a modo de que le ha llegado y por tanto, cuándo puede ser referenciada o confirmada por otras transacciones.

Con la lambda se generan tiempos de llegada para cada transacción.  Durante el procesado de cada transacción, parece que el simulador verifica si la transacción es visible para otros nodos en un momento específico del tiempo, compara el tiempo de llegada de una nueva transacción (ajustado por esa latencia) con los tiempos de llegada de transacciones anteriores. Si el tiempo de llegada de la transacción anterior más la latencia es <= al tiempo de llegada de la nueva transacción entonces, la transacción anterior es visible y puede ser "referenciada" o "validada" por llamarlo así.

Se mide en las mismas unidades de tiempo que los tiempos de llegada, segundos vamos, esa latencia se suma a los tiempos de llegada como te he dicho antes para ver cuando se considera visible.

Por ejemplo, si una transacción llega en el segundo 5 y la latencia es de 1 segundo, esta transacción no será visible para otras transacciones que lleguen antes del segundo 6.

**Distancia (d)**: distancia a la que están los agentes entre sí.

Ejemplo entre distancia = 1 y distancia = 10

Diagrama

Descripción generada automáticamente

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Podemos ver los tips que los agentes logran tener activos durante un periodo de tiempo. Con distancias más bajas entre agentes vemos un crecimiento estable en los tips y un TANGLE más estable. Con distancias más largas puede volver a darse la superioridad de un agente sobre otro y afectar al Tangle.

Usa una matriz de distancia para representar la latencia entre cada par de agentes en función de la distancia. En el código parece que la distancia simplemente determina qué transacciones son visibles para un agente dado, jugando con esa latencia. La matriz establece cómo los diferentes agentes están conectados entre sí y cuánto tiempo toma para que la información de un agente llegue a otro. Cada entrada en la matriz indica el retardo (o latencia) entre un par de agentes. También parece que usa la matriz para escoger el camino óptimo entre dos agentes, creo que lo he entendido así al leer el artículo.

Captura de pantalla de computadora

Descripción generada automáticamente con confianza media

**tip\_selection\_algo**: Define el método utilizado para seleccionar qué transacciones anteriores aprobará una nueva transacción, podemos elegir entre "random", "weighted", "unweighted" (aleatorio, teniendo en cuenta sus pesos o sin tener en cuenta los pesos)

Ejemplo weighted vs random:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Podemos ver que ambos agentes tienen tips disponibles en el caso random (no se observan casos de crecimiento "abrupto") y no se da el caso en el que en un momento un agente pega un crecimiento bestial (pasando de 1 tip a 4) como en el caso de weighted.