Speedrun de Super Mario 64

María Fernanda Suárez González, Adrián Navarro Foya y Daniel Polanco Pérez

Universidad de La Habana

Resumen Super Mario 64 es un videojuego icónico que revolucionó los juegos de plataformas en 3D. En este trabajo, exploramos dos mecánicas fundamentales para realizar un speedrun del juego: el **Enrutamiento de Etapas** y el **Impulso de Daño**. El Enrutamiento de Etapas consiste en determinar el orden óptimo para completar los niveles del juego, maximizando el uso de habilidades desbloqueables. Por otro lado, el Impulso de Daño se enfoca en seleccionar la mejor combinación de eventos dentro de un nivel para maximizar el tiempo recuperado, a pesar del daño recibido. Demostramos que ambos problemas son computacionalmente complejos: el problema de Routing es NP-hard, y el problema de Impulso de Daño admite un esquema de aproximación en tiempo pseudo-polinomial (FPTAS). Además, presentamos una adaptación de estas mecánicas para Super Mario 64, teniendo en cuenta la progresión del juego basada en la recolección de estrellas. Finalmente, proporcionamos estrategias y algoritmos para optimizar el tiempo de finalización del juego, lo que puede ser útil tanto para speedrunners como para entusiastas de la teoría de la complejidad computacional aplicada a videojuegos.

Keywords: Super Mario 64 · Speedrun · Enrutamiento de Etapas · Impulso de Daño · Complejidad computacional · NP-hard · FPTAS · Algoritmos de aproximación · Optimización · Videojuegos.

Introducción

Super Mario 64 es un videojuego de plataformas en 3D desarrollado por Nintendo bajo la dirección de Shigeru Miyamoto y lanzado en 1996 para la consola Nintendo 64. En el juego, el jugador controla a Mario, un fontanero italiano, quien debe explorar diversos mundos dentro del castillo de la Princesa Peach para recolectar Estrellas de Poder y rescatar a la princesa, secuestrada por Bowser. Este título revolucionó los juegos de plataformas al introducir movimientos en tres dimensiones y se convirtió en un referente para futuras entregas de la saga, como Super Mario Sunshine, Super Mario Galaxy y Super Mario Odyssey. El personaje de Mario, ya consolidado como mascota de Nintendo, reforzó su popularidad con este éxito, apareciendo en cientos de juegos posteriores.

Desarrollo

La complejidad de un nivel de Mario 64

En Super Mario 64, Mario puede moverse libremente en entornos 3D. Inicialmente, Mario tiene habilidades básicas como correr, saltar, escalar y nadar.

Si obtiene Power-Ups especiales, como el Sombrero de Metal o el Sombrero con Alas, adquiere habilidades únicas:

- Metal Mario: Se vuelve invencible, inmune a daños y capaz de caminar bajo el agua.
- Wing Mario: Puede planear por el aire durante un tiempo limitado.
- Mario Invisible: Atraviesa ciertas barreras, aunque esta transformación no se usa en nuestras pruebas.

Las acciones básicas de Mario incluyen saltos normales (de 4.5 bloques de altura, para una referencia Mario mide 1.75 bloques), saltos triples en secuencia y carreras. Movimientos avanzados, como el salto de pared o el salto hacia atrás, permiten acceder a áreas ocultas. Mario también puede interactuar con objetos como bloques, interruptores y cañones, que lo lanzan a zonas distantes.

El mundo de Super Mario 64 está estructurado en çursos"dentro del castillo de Peach, cada uno con desafíos únicos para obtener Estrellas de Poder. El entorno incluye:

- Bloques estándar: Sólidos y sin propiedades especiales.
- Bloques de monedas: Liberan monedas al ser golpeados.
- Bloques de Power-Ups: Otorgan habilidades temporales al romperse.
- Plataformas móviles: Se desplazan en patrones predefinidos, exigiendo sincronización.

Para más informacion del juego ver [4].

En esta sección vamos a demostrar lo siguiente:

Theorem 1. *El problema de determinar si se puede alcanzar una meta desde un inicio en un nivel generalizado de Mario 64 es NP-Hard.*

Demostración. Dado un conjunto $S = \{C_1, \ldots, C_l\}$ de l cláusulas construidas a partir de n variables booleanas tenemos que contruir una instancia de un nivel generalizado de Mario 64 tal que el nivel es completable ssi S es satisfacible.

Para esto vamos a utilizar las siguientes estructuras especializadas que vamos a denominar "gadgets"los cuales replicarán los elementos lógicos del problema 3-SAT.

- Variable: Este gadget representa las variables booleanas del problema 3-SAT. Va a obligar al jugador a elegir exclusivamente entre dos caminos: el camino True (x) o el camino False ($\neg x$). Una vez elegido un camino se bloquea permanentemente la alternativa para que de esta forma no se pueda retroceder o cambiar de elección.
- Cláusula: Este gadget representa las cláusulas del problema 3-SAT. Este gadget solo es accesible desde los caminos de literales que la componen. Al visitar este gadget el jugador lo "desbloquea"(esto es un cambio de estado permanentemente).
- Check: Solo se puede ″desbloquear″ este gadget si se desbloquearon los de todas las cláusulas. La meta final del jugador es desbloquear este último gadget.

La instancia y el flujo del nivel se construyen de la siguiente manera. Inicialmente Mario ″spawnea″en el gadget que representa a la variable x_1 como se muestra en la figura 1.

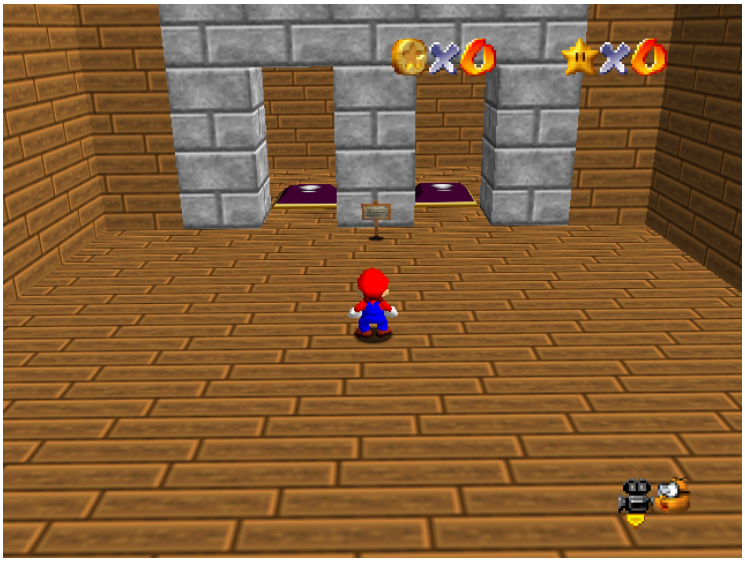

Figura 1. Gadget de Variable

Como se puede observar el gadget de cada variable es una habitación con dos salidas. La puerta de la izquierda representa el camino True y la de la derecha el camino False. Una vez que se cruza una de estas puertas Mario presionará el botón que se encuentra en el umbral de la puerta lo que activará un mecanismo que le impedirá regreasar y revertir su decisión.

Según el camino que escoja el jugador este le conducirá a los gadgets de las cláusulas en las que ese literal esté presente. Un gadget de cláusula consiste en una habitación con una moneda roja como se puede observar en la figura 2 . Hay l monedas rojas, una por cada cláusula. El objetivo del jugador es coleccionar todas estas para desbloquear la estrella en el Check gadget. Cada vez que el jugador sale de un gadget de variable el camino que tome le conducirá por los gadgetes de cláusulas en las que ese literal está presente. Una vez que termine de recorrer todas esas cláusulas el camino le conducirá al gadget de la siguente variable y se repite el proceso nuevamente.

Una vez que ya se recorrieron todas los gadgets de variables o todos los de cláusulas el camino conducirá al jugador al Check gadget 3. En caso de que Mario haya pasado por todas las cláusulas y coleccionado las monedas rojas entonces la estrella aparecerá y Mario podrá obtenerla para completar el nivel, en caso de que Mario no haya pasado por todas las cláusulas la estrella no aparecerá.

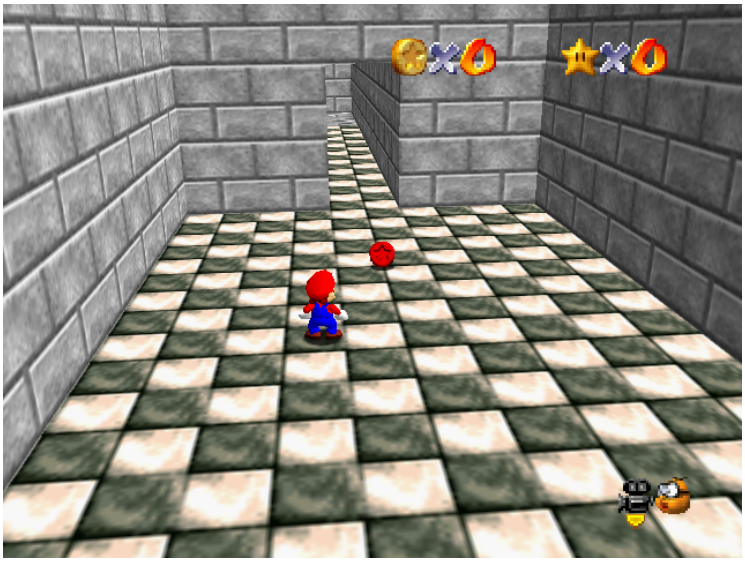

Figura 2. Gadget de Cláusula

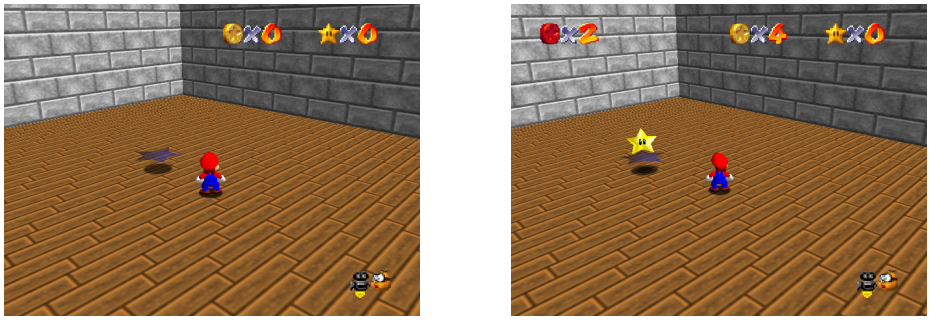

Figura 3. Check gadget

Nos queda probar que nuestra insctancia de 3-SAT es satisfacible ssi el nivel que creamos es completable.

Supongamos que S es satisfacible y sea v una asignación satisfacible. Si $v(x_i)$ es True Mario debe tomar la puerta que conduce por el camino True en el gadget de la variable x_i, si $v(x_i)$ es False Mario debe tomar el camino False. Como v es una asignación satisfacible al menos un literal de cada cláusula toma valor True y de esta forma aseguramos que Mario obtenga todas las menos rojas disponibles en las cláusulas. Finalmente cuando Mario llegue A al gadget Check la estrella estará disponible y podrá terminar el nivel.

Por otro lado supongamos que el nivel es completable. Si el nivel es completable existe un camino desde el inicio el cual pasa por todas las cláusulas y desbloquea la estrella en el Check gadget. Dependiendo de la puerta que escoja Mario en cada gadget de varible podemos construir una asignación satisfacible para S, si escogió el camino True para la variable x_i entonces le asignamos valor True a x_i y si escogió el camino False le asignamos el valor False.

Por tanto podemos afirmar que completar un nivel de Mario 64 generalizado es NP-Hard.

Estrategias de Solución

Para hacer un speedrun de Mario 64 tenemos dos mecánicas principales: **Enrutamiento de Etapas**, para escoger en qué orden acceder a los niveles del juego para utilizar las diversas habilidades desbloqueables. La otra mecánica es **Impulso de Daño**, la cual consiste en escoger la mejor combinación de eventos tal que maximice el tiempo recuperado gracias a tomar ese evento a pesar del daño recibido, esto por cada nivel.

Formalicemos estas ideas

Impulso de Daño

Un nivel N va a ser representado como una secuencia $S = (e_1, \ldots, e_n)$ de eventos.

Un evento e es una oportunidad de ganar un tiempo t recibiendo una cantidad de daño d, es decir, $e = (d, t)$.

- Asumimos que d y t son enteros posiblemente negativos.
- Si d y t son ambos negativos, le denominaremos **evento de salud**.
- $d(e)$ y $t(e)$ serán las notaciones para el daño recibido y el tiempo recuperado del evento e.

Una etapa $S = (e_1, \ldots, e_n)$ es una lista ordenada de eventos.
Una solución $\hat{S} = (\hat{e}_1, \ldots, \hat{e}_n)$ de una etapa S es una subsecuencia de S.
Un evento e_i es seleccionado si $e_i \in \hat{S}$.
Un entero hp representa los puntos de salud de Mario al inicio de la etapa.

- Los puntos de salud de Mario no pueden ser excedidos por el daño recibido.
- Cada evento $\hat{e}_i \in \hat{S}$ deja al jugador con un número de puntos de salud $h_{\hat{S}}(\hat{e}_i)$ después de ser tomado.

Definimos:

$$h_{\hat{S}}(\hat{e}_1) = \text{mín}(hp, hp - d(\hat{e}_1))$$

y

$$h_{\hat{S}}(\hat{e}_i) = \text{mín}(hp, h_{\hat{S}}(\hat{e}_{i-1}) - d(\hat{e}_i)) \quad \forall i \in \{2, \ldots, k\}$$

- Una solución es válida si $h_{\hat{S}}(\hat{e}_i) > 0 \ \forall \hat{e}_i \in \hat{S}$.

Dadas una etapa S y un máximo de puntos de salud hp, el objetivo de **Impulso de Daño** es encontrar una solución válida $\hat{S}$ para S que maximice:

$$t(\hat{S}) = \sum_{e \in \hat{S}} t(e)$$

Reducir problema de la Mochila a problema de la Mochila con pesos y valores negativos

Problema de la Mochila

Maximizar:

$$\sum_{i=1}^{n} v_i x_i$$

sujeto a:

- $\sum_{i=1}^{n} w_i x_i \leq W$
- $x_i \in \{0, 1\}$
- $v_i \geq 0 \ \forall i \in \{1, \ldots, n\}$
- $w_i \geq 0 \ \forall i \in \{1, \ldots, n\}$
- $W \geq 0$
- $v_i, w_i, W \in \mathbb{Z} \ \forall i \in \{1, \ldots, n\}$

Problema de la Mochila negativa

Maximizar:

$$\sum_{i=1}^{n} v_i x_i$$

sujeto a:

- $\sum_{i=1}^{n} w_i x_i \leq W$
- $x_i \in \{0, 1\}$
- $W \geq 0$
- $v_i, w_i, W \in \mathbb{Z} \ \forall i \in \{1, \ldots, n\}$

Reducción

Dividimos los elementos en 4 grupos:

- $v'_i = v_i + M_v$, $w'_i = w_i + M_w$ $\forall i \in \{1, \ldots, k_1\}$
- $v'_i = v_i + M_v$, $w'_i = w_i - M_w$ $\forall i \in \{k_1 + 1, \ldots, k_2\}$
- $v'_i = v_i - M_v$, $w'_i = w_i + M_w$ $\forall i \in \{k_2 + 1, \ldots, k_3\}$
- $v'_i = v_i - M_v$, $w'_i = w_i - M_w$ $\forall i \in \{k_3 + 1, \ldots, k_4\}$
- $M_v > 0$, $M_w > 0$, $M_v, M_w \in \mathbb{Z}$
- $n_1 = k_1$
- $n_2 = k_2 - k_1$
- $n_3 = k_3 - k_2$
- $n_4 = k_4 - k_3$
- $W' = W + (n_1 + n_3)M_w - (n_2 + n_4)M_w$

Equivalencia

$$\sum_{i=1}^{n} w_i x_i \leq W \Leftrightarrow \sum_{i=1}^{n} w'_i x_i \leq W'$$

$$\sum_{i=1}^{k_1}(w'_i x_i + M_w) + \sum_{i=k_1+1}^{k_2}(w'_i x_i - M_w) + \sum_{i=k_2+1}^{k_3}(w'_i x_i + M_w) + \sum_{i=k_3+1}^{k_4}(w'_i x_i - M_w) \leq W + (n_1+n_3)M_w - (n_2+n_4)M_w$$

$$\sum_{i=1}^{k_1}(w'_i x_i) + n_1 M_w + \sum_{i=k_1+1}^{k_2}(w'_i x_i) - n_2 M_w + \sum_{i=k_2+1}^{k_3}(w'_i x_i) + n_3 M_w + \sum_{i=k_3+1}^{k_4}(w'_i x_i) - n_4 M_w \leq W + (k_1 + k_3 - k_2)M_w - (k_2 - k_1 + k_4 - k_3)M_w$$

$$\sum_{i=1}^{n}(w_i x_i) + (k_1 - k_2 + k_1 + k_3 - k_2 - k_4 + k_3)M_w \leq W + (k_1 - k_2 + k_1 + k_3 - k_2 - k_4 + k_3)M_w$$

$$\sum_{i=1}^{n}(w_i x_i) \leq W$$

Análogo para la función objetivo.

Solución usando DP

El problema de **Impulso de Daño** está definido por la tupla (S, hp), las cuales son el conjunto de eventos del nivel y los puntos de salud de Mario.

Definimos $H(i,t)$, el cual representa la cantidad de puntos de salud restantes al tomar los primeros i eventos de la solución $\hat{S}$ recuperando una cantidad t de tiempo.

Definimos T como el tiempo máximo ganado en algún evento.

Definimos $H(0,0) = hp$.

Definimos $H(0,t) = -\infty \ \forall t > 0$. Por tanto:

$$H(i,t) = \min\{hp, \max\{H(i-1,t), H(i-1,t-t(e_i)) - d(e_i)\}\}$$

Luego, nuestra solución estará en $H(n,t_i) \mid t_i > 0$.

Como $H(i,t)$ necesita ser computado para cada i de 1 a n y para cada t_i, los cuales van desde 1 hasta nT (porque cada evento puede ser de tiempo T y el DP recorre hasta la suma de todos los tiempos), se puede solucionar en un tiempo pseudo-polinomial de $O(n^2T)$.

Pero esto es mejorable aún más. **Impulso de Daño** admite un FPTAS y puede ser aproximado con un factor de $1-\epsilon$ en tiempo $O(n^3/\epsilon)$ para todo $\epsilon > 0$.

Sea $\epsilon > 0$ y $c = \epsilon T/n$. Definimos $S' = (e'_1, \ldots, e'_n)$, donde $e'_i = (d(e_i), \lfloor t(e_i)/c \rfloor)$.

Además, recordemos que $\hat{S}$ es una subsecuencia de S que maximiza $t(\hat{S})$.

$\forall e_i \in S$ se cumple que:

$$t(e_i)/c - 1 \leq t(e'_i) \leq t(e_i)/c$$

$$t(e_i)/c - 1 \leq \lfloor t(e_i)/c \rfloor \leq t(e_i)/c$$

Por eso:

$$t(\hat{S}') \geq \sum_{e_i \in \hat{S}} t(e'_i) \geq t(\hat{S})/c - n$$

$$\sum_{e'_i \in \hat{S}'} t(e'_i) \geq \sum_{e_i \in \hat{S}} t(e'_i) \geq \sum_{e_i \in \hat{S}} t(e_i)/c - n$$

$$\sum_{e'_i \in \hat{S}'} \lfloor t(e_i)/c \rfloor \geq \sum_{e_i \in \hat{S}} \lfloor t(e_i)/c \rfloor \geq \sum_{e_i \in \hat{S}} t(e_i)/c - n$$

$$\sum_{e'_i \in \hat{S}'} \lfloor t(e_i)/c \rfloor \geq \sum_{e_i \in \hat{S}} \lfloor t(e_i)/c \rfloor \geq \sum_{e_i \in \hat{S}} (t(e_i)/c - 1) \geq \sum_{e_i \in \hat{S}} t(e_i)/c - n$$

$$\sum_{e'_i \in \hat{S}'} \lfloor t(e_i)/c \rfloor \geq \sum_{e_i \in \hat{S}} \lfloor t(e_i)/c \rfloor \geq \sum_{e_i \in \hat{S}} (t(e_i)/c) - n \geq \sum_{e_i \in \hat{S}} t(e_i)/c - n$$

$$\sum_{e'_i \in \hat{S}'} \lfloor t(e_i)/c \rfloor \geq \sum_{e_i \in \hat{S}} \lfloor t(e_i)/c \rfloor \geq \sum_{e_i \in \hat{S}} t(e_i)/c - n \geq \sum_{e_i \in \hat{S}} t(e_i)/c - n$$

Y la primera desigualdad se cumple por la condición de optimalidad de $\hat{S}'$ en S'.

Por tanto, tenemos asegurado que se cumple la desigualdad:

$$t(\hat{S}') \geq \sum_{e_i \in \hat{S}} t(e'_i) \geq t(\hat{S})/c - n$$

$\hat{S}'$ es una solución válida de S, ya que los valores de daño no cambiaron de S a S'. Luego, el tiempo recuperado tomando los eventos de $\hat{S}'$ como nuestra solución para S es:

$$\sum_{e'_i \in \hat{S}'} t(e_i) \geq \sum_{e'_i \in \hat{S}'} ct(e'_i) \geq c(t(\hat{S})/c - n) = t(\hat{S}) - cn = t(\hat{S}) - \epsilon T \geq (1-\epsilon)t(\hat{S})$$

Donde usamos que $t(\hat{S}) \geq T$ en la última inecuación. Por tanto, demostramos que $t(\hat{S}') \geq (1-\epsilon)t(\hat{S})$. El algoritmo es en tiempo $O(n^2 T/(\epsilon T/n)) = O(n^3/\epsilon)$.

Routing

El jugador puede visitar un conjunto de etapas en cualquier orden. Completar una etapa le otorga al jugador una nueva habilidad. Cada etapa tiene un conjunto de eventos de recuperación de tiempo, y cada habilidad puede ser usada para ganar tiempo en un evento. El tiempo recuperado en un evento depende de la mejor habilidad disponible. Un juego es un conjunto $S = \{S_1, \ldots, S_n\}$. Una etapa $S_i = \{e_1, \ldots, e_k\}$ es un conjunto de eventos. Un evento $e_i : S \to \mathbb{N}$ es una función que convierte cada etapa en un entero.

- Si la etapa S_i es superada, entonces un tiempo de $e_j(S_i)$ puede ser recuperado mientras atraviesas e_j usando la habilidad obtenida de S_i.
- Sea $C \subseteq S$. Escribiremos $e(C) = \max_{S \in C} e(S)$. Si C es el conjunto de etapas completadas, asumiremos que el evento e será superado usando la mejor opción disponible.

Dado C, el tiempo recuperado en una etapa S viene a ser:

$$t(S, C) = \sum_{e \in S} e(C)$$

El objetivo del problema de Routing es encontrar una ordenación lineal π de S tal que:

$$\sum_{i=1}^{n} t(S_i, \{S_j \mid S_j <_\pi S_i\})$$

sea máxima.

Definimos el grafo dirigido de dependencia $D(S)$ para un conjunto de etapas S:

- $D(S) = (S, A, w)$ tiene un vértice por cada etapa y, por cada par ordenado (i, j), un arco de S_i a S_j con peso $w(S_i, S_j) = \sum_{e \in S_j} e(S_i)$.

Demostración

Para una etapa S_i, denotamos por e_i al único evento de S_i. Dado un orden π de las etapas $S = \{S_1, S_2, \ldots, S_n\}$, definimos $p_\pi(S_i)$ como la etapa anterior a S_i en π que maximiza la ganancia de tiempo en e_i. Formalmente:

$$p_\pi(S_i) = \arg \max_{S_j <_\pi S_i} e_i(S_j),$$

donde $S_j <_\pi S_i$ significa que S_j aparece antes que S_i en el orden π. Esta elección es única porque cada etapa tiene un solo evento.

La ganancia total de tiempo para un orden dado π es:

$$t = \sum_{S_i \in S} w(p_\pi(S_i), S_i),$$

donde $w(p_\pi(S_i), S_i)$ representa el peso del arco dirigido desde $p_\pi(S_i)$ a S_i.

Ahora consideremos el conjunto de arcos:

$$A' = \{(p_\pi(S_i), S_i) : 1 < i \leq n \text{ y } e_i(p_\pi(S_i)) > 0\}.$$

Este conjunto tiene dos propiedades importantes:

- **Es acíclico**: El subgrafo formado por A' no contiene ciclos dirigidos. Esto se debe a que cada arco en A' apunta hacia una etapa posterior en el orden π, lo que garantiza que no haya ciclos.
- **Tiene grado de entrada limitado**: Cada vértice en el subgrafo tiene a lo sumo un arco entrante, excepto el primer vértice de π, que no tiene ningún arco entrante.

Estas propiedades implican que A' forma una *ramificación* (un subgrafo acíclico dirigido donde cada vértice tiene a lo sumo un arco entrante). Además, el peso total de A' es exactamente la ganancia de tiempo t.

Por el contrario, dada cualquier ramificación B de $D(S)$ con un conjunto de arcos A', podemos obtener un orden lineal de S realizando un ordenamiento topológico de B. En este orden, cada evento e_{S_i} puede completarse utilizando la entrada del vecino de S_i en A' (si existe). Por lo tanto, el problema de encontrar el orden óptimo π se reduce a encontrar una ramificación de peso máximo en $D(S)$.

Luego, el problema de encontrar una ramificación de peso máximo puede resolverse en tiempo $O(|A| + |S| \log |S|)$ mediante una reducción al problema del árbol generador dirigido de peso máximo, utilizando el algoritmo de Gabow et al. [1].

Por tanto, hemos demostrado que si cada etapa contiene un solo evento, el problema de Routing puede resolverse en tiempo $O(|A| + |S| \log |S|)$.

Equivalencia al problema del sub-DAG de peso máximo

Ahora demostremos que si, para cada evento e, existe solo una etapa $S_i \in S$ tal que $e(S_i) > 0$, entonces ROUTING es equivalente al problema de encontrar el subgrafo acíclico dirigido (DAG) de peso máximo en $D(S)$.

Demostración Supongamos que tenemos una instancia de **ROUTING** definida por un conjunto de etapas $S = \{S_1, S_2, \ldots, S_n\}$, donde cada etapa S_i tiene un conjunto de eventos e. Además, asumimos que para cada evento e, existe exactamente una etapa $S_i \in S$ tal que $e(S_i) > 0$. Esto significa que cada evento depende únicamente de una etapa específica.

Construimos el grafo $D(S)$ de la siguiente forma:

- El grafo $D(S) = (S, A, w)$ tiene un vértice para cada etapa S_i.
- Para cada par ordenado (i, j), hay un arco desde S_i a S_j con peso $w(S_i, S_j) = \sum_{e \in S_j} e(S_i)$. Este peso representa la ganancia de tiempo que se obtiene al completar S_i antes de S_j.
- Si S_i se completa antes que S_j, entonces el evento $e \in S_j$ puede aprovechar el arma obtenida de S_i para ahorrar tiempo.
- Como cada evento e depende únicamente de una etapa S_i, el peso $w(S_i, S_j)$ captura exactamente la contribución de S_i a S_j si S_i se completa antes que S_j.

Dado un orden π de las etapas S, la ganancia total de tiempo es:

$$t = \sum_{S_i <_\pi S_j} w(S_i, S_j),$$

donde $S_i <_\pi S_j$ significa que S_i aparece antes que S_j en el orden π.

Los arcos $\{(S_i, S_j) \in A : S_i <_\pi S_j\}$ no pueden formar ciclos en $D(S)$, ya que π es un orden lineal. Un orden π que maximiza la ganancia de tiempo t corresponde directamente a un subgrafo acíclico dirigido (DAG) de $D(S)$ con peso total t.

Por lo tanto, resolver **ROUTING** es equivalente a encontrar un sub-DAG de peso máximo en $D(S)$.

Ahora mostramos que cualquier solución al problema del sub-DAG de peso máximo en $D(S)$ puede convertirse en una solución equivalente para **ROUTING**.

Construimos una instancia de ROUTING:

- Dado un grafo $H = (V, A, w)$ que representa una instancia del problema del sub-DAG de peso máximo, creamos una instancia de **ROUTING** de la siguiente manera:
- Para cada vértice $u \in V$, creamos una etapa S_u.
- Para cada arco $(v, u) \in A$, agregamos un evento e_v^u en la etapa S_u con $e_v^u(S_v) = w(v, u)$.
- Todos los demás tiempos de finalización de eventos se establecen en 0.

Demostremos ahora que las soluciones son equivalentes:

- Supongamos que H tiene un sub-DAG de peso t. Entonces, podemos obtener un orden π de las etapas S en **ROUTING** que corresponda a un ordenamiento topológico del sub-DAG.
- En este orden π, la ganancia total de tiempo será exactamente t, ya que cada arco (v, u) en el sub-DAG contribuye con $w(v, u)$ al tiempo ganado.

Recíprocamente, dado un orden π de las etapas en **ROUTING** con ganancia de tiempo t, podemos construir un sub-DAG de H con peso total t seleccionando los arcos correspondientes a los pares (S_i, S_j) tales que $S_i <_\pi S_j$.

La equivalencia entre **ROUTING** y el problema del sub-DAG de peso máximo implica que todos los resultados de dureza conocidos para el problema del sub-DAG de peso máximo también se aplican a **ROUTING**.

En particular:

- **ROUTING** es NP-duro incluso si el grado máximo de $D(S)$ es 4 (debido a la dureza del problema de cubrimiento de vértices en grafos cúbicos [2]).
- El problema del sub-DAG de peso máximo no puede aproximarse dentro de una razón mejor que 1/2, asumiendo la Conjetura de Juegos Únicos [3].

Aproximación

A pesar de la dureza del problema, es fácil alcanzar una aproximación de factor 1/2 para **ROUTING**, siguiendo el mismo principio que para el problema del sub-DAG de peso máximo:

- Tomar cualquier orden π de las etapas.
- O bien π o su inverso alcanzará al menos 1/2 de la máxima ganancia de tiempo posible.

Esto se formaliza en la siguiente proposición:

ROUTING admite un algoritmo de aproximación de factor 1/2.

Tenemos que:

$$\sum_{i=1}^{n} \sum_{e \in S_i} e(S)$$

es una cota superior obvia en la ganancia de tiempo máxima alcanzable, ya que sería escoger la mejor arma para cada evento. Entonces, si elegimos un orden aleatorio $(S_1, \ldots, S_n)$ de S, y sea $(S_n, \ldots, S_1)$ el orden inverso, uno de estos dos debe lograr una ganancia de tiempo $e(S)$ para al menos la mitad de los eventos e que están en S.

Adaptación para Mario 64

Ahora solo tenemos que resolver un pequeño problema con respecto a nuestro juego, y es que en Mario 64 no todos los niveles están disponibles desde el

principio, como define el problema de Routing. Por tanto, tenemos que hacer una ligera adaptación.

En Mario 64, según la cantidad de estrellas obtenidas, se desbloquean nuevas etapas (niveles o zonas del castillo). Por tanto, podemos hacer que por cada cantidad de estrellas que desbloquean nuevos niveles, hagamos un problema nuevo de Routing.

Inicialmente, el jugador empieza con 0 estrellas, lo cual solo desbloquea la zona inicial del castillo y el primer nivel. Eso nos crea un problema de Routing con dos etapas. Posteriormente, al desbloquear la primera estrella, tenemos un nuevo problema de Routing, añadiendo el segundo nivel, y al obtener la tercera estrella, dos niveles más. Y así sucesivamente...

Las cantidades de estrellas que desbloquean nuevas etapas son:

$$[0, 1, 3, 8, 10, 12, 15, 20, 25, 30, 35, 50, 60, 70]$$

Luego, como el problema de Routing es NP-hard, 14 problemas de Routing son también NP-hard.

Referencias

1. Paola Alimonti and Viggo Kann. Hardness of approximating problems on cubic graphs. In Italian Conference on Algorithms and Complexity, pages 288-298. Springer, 1997.
2. Harold N Gabow, Zvi Galil, Thomas Spencer, and Robert E Tarjan. Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. Combinatorica, 6(2):109-122, 1986.
3. Venkatesan Guruswami, Rajsekar Manokaran, and Prasad Raghavendra. Beating the random ordering is hard: Inapproximability of maximum acyclic subgraph. In Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on, pages 573-582. IEEE, 2008.
4. Mario Wiki, `https://www.mariowiki.com/Super_Mario_64`