Optimización de Cronograma de Rodaje para un Programa de Televisión o Película

Ana Melissa Alonso Reina C-412 Alejandro Ramírez Trueba C-411

February 17, 2025

1 Introducción

Este proyecto se enfoca en la planificación y gestión integral de todos los recursos necesarios para rodar un programa de televisión o una película de manera eficiente y ajustada al presupuesto. Para lograrlo, se consideran múltiples factores que suelen complicar la producción:

- **Disponibilidad de actores:** Cada actor o actriz tiene un horario limitado que puede solaparse con otras actividades o producciones, por lo que la asignación de escenas debe coincidir con su presencia.
- **Restricciones de localizaciones:** Muchas locaciones están disponibles solo ciertos días o requieren permisos especiales, lo que obliga a programar secuencias específicas durante ese margen. Además, las localizaciones deben ser usadas en el momento del día que corresponde con las escenas que se filman en ellas, porque si una escena es en el patio de la escuela al atardecer no se puede filmar por la noche.
- **Continuidad narrativa:** Algunas escenas deben rodarse antes que otras para mantener la coherencia de la trama; esto implica respetar un orden parcial en la grabación.
- **Presupuesto fijo:** A menudo, los gastos totales no pueden superar un monto predeterminado, por lo que cualquier cambio en la planificación debe considerar su efecto sobre el costo final. También se puede considerar el gasto de combustible al dirigirse de una localización a otra para filmar en más de una en el mismo día.

La meta principal es minimizar el tiempo total requerido para completar el rodaje y evitar sobrecostos, todo ello manteniendo la calidad de la producción y cumpliendo con los plazos de entrega pactados. Para abordar estos retos, se pueden proponer distintos enfoques, que tendrán sus fortalezas y limitaciones, pero todos deben lidiar con la complejidad combinatoria que nace de la gran cantidad de restricciones interconectadas.

Este problema es de gran importancia práctica en el ámbito audiovisual, ya que la correcta organización de recursos humanos y técnicos puede ahorrar miles de dólares al evitar retrasos o cambios de última hora. Además, garantizar que todos los elementos necesarios (actores, locaciones, equipo) estén disponibles en el momento oportuno reduce la probabilidad de rehacer escenas o duplicar esfuerzos. Por otra parte, la industria audiovisual trabaja a menudo bajo calendarios muy

ajustados, por lo que toda mejora en la eficiencia del programa de rodaje se traduce en grabaciones más ágiles y un mayor cumplimiento de las fechas de entrega.

2 Estado del arte

Debido a la complejidad inherente a este problema, se han realizado múltiples investigaciones que proponen soluciones considerando diversos aspectos. Estas investigaciones buscan encontrar soluciones óptimas mediante enfoques que priorizan diferentes elementos del problema. En el trabajo "Optimal Scheduling in Film Production to Minimize Talent Hold Cost" [1], se presenta un modelo combinatorio donde la minimización del costo debido a los días de espera es el único criterio para la programación óptima de un filme. Aunque generalmente se involucran una variedad de otros criterios en el proceso de toma de decisiones, en este caso específico se aísla el costo de los días de espera debido a su prominencia en situaciones prácticas. Esto permite una modelización más precisa y centrada en este factor clave. Otras consideraciones tales como la disponibilidad restringida de algunos talentos, costos de preparación de escenas, restricciones artísticas, y otros factores no son considerados en este trabajo. Por lo tanto, se intenta secuenciar los días de filmación de manera que los actores sean empleados de la manera más continua posible y el costo de pagar a los actores por días de espera se minimice. En la literatura también se ha abordado el problema de la programación de rodajes no solo desde la perspectiva del costo de los actores, sino también considerando el costo de alquilar locaciones de filmación, como se expone en [4]. El costo de alquiler de una locación puede variar significativamente, no solo día a día, sino también hora a hora, debido a la fluctuación de la demanda. Por ejemplo, filmar en días festivos suele ser más costoso en comparación con días regulares, y a veces filmar de noche no es factible bajo un presupuesto estricto. La filmación diurna de escenas nocturnas, un proceso conocido como day-for-night, es generalmente más rentable, ya que permite utilizar locaciones costosas o indisponibles por la noche. Estas variaciones en los costos motivan la necesidad de estudiar una extensión del problema de programación de talentos que considere el costo de las locaciones, denominado TSP-LC (Talent Scheduling Problem with Location Costs). El objetivo del modelo propuesto en [4] es encontrar la secuencia de rodaje y las fechas de inicio de cada escena que minimicen el costo total, incluyendo los costos de actores y locaciones. Aunque no es necesario que las escenas se filmen consecutivamente, todas deben completarse dentro del horizonte de planificación.

Además, algunos artículos han abordado el problema de la programación de escenas de películas considerando la ubicación de la filmación, el costo del traslado de escenas, la remuneración de los actores y la duración de la filmación [5]. Este estudio propone un modelo matemático utilizando métodos de investigación operativa para encontrar la solución óptima al problema. Los métodos de investigación operativa se utilizan ampliamente en la gestión de personal, la selección y evaluación de proyectos, entre otros aspectos.

El problema de la programación de escenas de películas se abstrae en un modelo de programación lineal entera (ILP). Para resolver este problema, se utilizan algoritmos metaheurísticos como el método basado en búsqueda tabu (TSBM), el método basado en optimización por enjambre de partículas (PSOBM), el método basado en optimización por colonia de hormigas (ACOBM), el método basado en genética (GBM), el método basado en recocido simulado (SABM), el método basado en redes neuronales artificiales (ANNBM), y el método basado en programación evolutiva (EPBM).

El TSBM es una aplicación exitosa del algoritmo de optimización combinatoria, que utiliza una técnica de memoria flexible en el proceso de búsqueda, conocida como la lista tabu. El PSOBM

tiene una estructura simple y menos parámetros a controlar. El ACOBM demuestra una fuerte robustez en la resolución de problemas. En el estudio, se adoptan TSBM, PSOBM y ACOBM para analizar el problema de la programación de escenas de películas (MSSP) y se comparan y analizan sus desempeños mediante experimentos numéricos.

Paper 1

3 Problema de Programación de Filmación (FS).

Uno de los costos más críticos y a menudo subestimados en la programación eficiente de los días de rodaje es el asociado con los "días de espera" del talento, que incluye actores, extras y a veces personal técnico especializado.

3.1 Definición del Problema

Los días de espera son aquellos en los cuales el talento está presente en el set de rodaje pero no participa activamente en la filmación de escenas. Durante estos días, aunque no están trabajando, los actores y otros talentos deben ser compensados, generando un gasto considerable para la producción.

3.2 Desafíos Específicos

- **Coordinación de múltiples actores:** Cada actor puede tener diferentes días en los que es necesario que participe, dependiendo del guion y la logística de las locaciones.
- **Restricciones de disponibilidad:** A menudo los actores tienen compromisos previos o limitaciones contractuales que restringen su disponibilidad.
- **Complejidad combinatoria:** El número de combinaciones posibles de días de rodaje aumenta exponencialmente con el número de actores y días involucrados, lo que convierte la programación en un problema combinatorio complejo.

3.3 Importancia de la Optimización

Dada la naturaleza competitiva y costosa de la industria cinematográfica, la eficiencia en la programación puede traducirse en ahorros significativos y en una mayor capacidad para invertir en otros aspectos de la producción como efectos especiales, mejores locaciones o talentos de mayor renombre. Además, una planificación óptima mejora el ambiente de trabajo al reducir los tiempos de inactividad forzada, lo que potencialmente puede afectar el desempeño y la satisfacción del talento.

4 Modelación del Problema FS

4.1 Entradas del Problema

- **Matriz DODM (Day Out of Days Matrix):** Matriz binaria $T^0 \in \{0,1\}^{m \times n}$, donde:

$$t_{ij}^0 = \begin{cases} 1, & \text{si el actor } i \text{ está requerido el día } j, \\ 0, & \text{en otro caso.} \end{cases}$$

- **Vector de Costos:** $C \in \mathbb{R}^m$, donde c_i es el costo por día de espera del actor i.

4.2 Función Objetivo

Dada una permutación σ de los días de rodaje, el costo total de días de espera se calcula como:

$$K(\sigma) = \sum_{i=1}^{m} c_i \left[l_i(\sigma) - e_i(\sigma) + 1 - \sum_{j=1}^{n} t_{ij}^0 \right],$$

donde:

- $e_i(\sigma)$: Primer día en que el actor i está requerido en la secuencia σ.
- $l_i(\sigma)$: Último día en que el actor i está requerido en σ.
- $\sum_{j=1}^{n} t_{ij}^0$: Total de días que el actor i está requerido.

5 Demostracion NP-Hard del Problema de Programación de Filmación

En esta sección, demostraremos que el problema de programación de filmación (FS) es NP-Hard mediante una reducción desde el problema de **Optimal Linear Arrangement (OLA)**, el cual es NP-Hard[2]. El problema FS es **NP-Hard**, ya que incluso su versión restringida, donde cada actor es requerido exactamente en dos días y el vector de pago es unitario, es reducible polinomialmente a OLA. Esto implica que la existencia de un algoritmo pseudopolinomial es improbable [3].

Problema de Optimal Linear Arrangement (OLA)

Entrada: Un grafo no dirigido $G = (V, E)$ y un entero positivo B. **Pregunta:** ¿Existe una función biyectiva $\sigma : \{1, 2, \ldots, |V|\} \to \{1, 2, \ldots, |V|\}$ tal que

$$\sum_{\{v_i, v_j\} \in E} |\sigma(i) - \sigma(j)| \leq B?$$

Teorema

El problema FS es NP-Hard.

Demostración

Dada una instancia de OLA con $G = (V, E)$ y B, construimos una instancia del problema FS de la siguiente manera:

1. **Actores como aristas:** Cada arista $E_i \in E$ corresponde a una fila i en la matriz DODM T^0.
2. **Días como vértices:** Cada vértice $V_j \in V$ corresponde a una columna j de T^0, donde:
$$t_{ij}^0 = \begin{cases} 1, & \text{si } V_j \text{ es un extremo de } E_i, \\ 0, & \text{en otro caso.} \end{cases}$$
3. **Costos unitarios:** El vector de pago se define como $c_i = 1$ para todo i.

Afirmamos que la respuesta a OLA es "sí" si y solo si la instancia de FS tiene un costo de días de espera $K(\sigma) \leq B - |E|$.

$$\text{Observamos que } \sum_{j=1}^{|V|} t_{ij}^0 = 2 \quad \forall i.$$

Para una permutación σ, el costo de días de espera es:

$$\begin{aligned} K(\sigma) &= \sum_{i=1}^{|E|} 1 \times [l_i(\sigma) - e_i(\sigma) + 1 - 2] \\ &= \sum_{i=1}^{|E|} [l_i(\sigma) - e_i(\sigma)] - |E| \\ &= \sum_{\{v_i, v_j\} \in E} |\sigma(i) - \sigma(j)| - |E|. \end{aligned}$$

Si existe una permutación σ para OLA con $\sum |\sigma(i) - \sigma(j)| \leq B$, entonces $K(\sigma) \leq B - |E|$. La conversa también es cierta. Esto completa la reducción polinomial y demuestra que FS es NP-Hard. □

6 Algoritmos y Análisis

6.1 Algoritmo de Ramificación y Poda (Branch-and-Bound)

Descripción: El algoritmo de ramificación y poda comienza con la construcción de un árbol de búsqueda, donde cada nodo representa una secuencia parcial de los días de rodaje. La construcción de la secuencia se realiza desde los extremos hacia el centro, determinando primero los días extremos y luego los intermedios. Se calcula una cota inferior para cada nodo, representando el costo mínimo de días de espera si la secuencia de ese nodo se completara hasta formar una secuencia total.

Construcción de cotas: Las cotas inferiores se calculan usando la información de los días ya programados en la secuencia parcial. Se considera el número mínimo de días de espera que un actor debe incurrir, dado que algunos días de rodaje ya están fijados en la secuencia y su presencia es requerida en días aún no programados.

- **Cota inferior:** Para un nodo P, se calcula como:

$$K_{\text{lower}}(P) = \sum_{i=1}^{m} c_i \left[D_i^e(P) + D_i^l(P) \right],$$

donde $D_i^e(P)$ y $D_i^l(P)$ son días de espera obligatorios para el actor i en la secuencia parcial P.

Complejidad y Correctitud:

- **Tiempo:** La complejidad es $O(n!)$, reflejando que en el peor caso, se deben explorar todas las permutaciones posibles de los días de rodaje.
- **Correctitud:** La poda elimina del árbol de búsqueda aquellas secuencias que definitivamente no pueden superar al mejor costo encontrado hasta el momento, asegurando que no se descarten soluciones óptimas.

6.2 Método Heurístico

Descripción: Este método heurístico incluye dos pasos principales:

1. **Paso 1 - Ruta Más Atractiva:** Construcción de una secuencia inicial seleccionando los días que minimicen el incremento inmediato del costo total de días de espera, empezando desde los extremos hacia el centro.
2. **Paso 2 - Intercambios de Pares:** Mejora de la secuencia inicial mediante intercambios de pares de días para reducir el costo total. El proceso se repite hasta alcanzar un óptimo local.

Complejidad y Correctitud:

- **Tiempo:** $O(n^3)$ por iteración, debido a los intercambios de pares ($O(n^2)$) y los cálculos de costo por intercambio ($O(n)$).
- **Correctitud:** No garantiza la solución óptima global pero es eficaz para obtener soluciones cercanas al óptimo local con un error promedio menor al 10%.

Paper 2

7 Talent Scheduling Problem que toma en cuenta el costo de locaciones(TSP-LC)

8 Descripción del Problema

Se considera un conjunto de escenas $S = \{s_1, s_2, ..., s_n\}$ y un conjunto de actores $A = \{a_1, a_2, ..., a_m\}$. Cada escena s_j requiere un subconjunto de actores $A_j \subseteq A$ y tiene una duración d_j. Los actores reciben un pago diario desde su primera aparición en el rodaje hasta la última, independientemente de si participan en todas las escenas de ese período. Además, cada locación tiene un costo de alquiler variable a lo largo del tiempo.

El objetivo es determinar una secuencia de filmación y las fechas de inicio de cada escena para minimizar el costo total, considerando:

- Costos de los actores basados en su período de contratación.
- Costos de alquiler de locaciones, que varían a lo largo del horizonte de planificación.
- Restricciones de precedencia entre escenas.
- Cumplimiento del horizonte de planificación.

9 Formulación Matemática

El problema se modela como un **problema de programación entera mixta (MILP)** con las siguientes variables de decisión:

- x_{ij}: variable binaria que indica si la escena s_j se filma después de s_i.
- y_{ij}: variable binaria que indica si la escena s_i comienza en el día j.
- t_j: día de inicio de la escena s_j.
- e_i, l_i: primer y último día en que el actor a_i es requerido.

El modelo optimiza la función objetivo:

$$\min \sum_{i=1}^{m} c_i(l_i - e_i + 1) + \sum_{i=1}^{n} \sum_{j=1}^{p} y_{ij} L_{ij}, \tag{1}$$

donde c_i representa el pago diario del actor a_i y L_{ij} el costo de alquiler de locación si la escena s_i comienza el día j.

Las restricciones aseguran la secuenciación correcta de las escenas, la asignación de fechas dentro del horizonte de planificación y el cálculo correcto del tiempo de contratación de los actores.

10 Demostración de NP-Hard de TSP-LC

Para demostrar que es NP-Hard nos basaremos en el problema OLA al igual que hicimos para demostrar que FS era NP-Hard. Dada una instancia de OLA, cuya entrada es un grafo no dirigido G = (V, E) y un entero positivo B,construimos la matriz que recibe TSP-LC (las filas son los actores y las columnas las escenas en las que trabajan) tal que por cada arista que pertenezca a E voy a tener una fila en la matriz y por cada vértice de G voy a tener una columna. En una celda de esta matriz ponemos una X solamente si el vértice que corresponde a la columna es uno de los extremos de la arista de la fila. Declaro el costo de renta de cada localización como 0 para cualquier fecha y le asigno a cada escena, representada por las columnas de la matriz, una duración de exactamente un día. Para todos los actores los salarios correspondientes(ci) se fijarán en 1. De esta manera la función objetivo solo necesita minimizar:

$$\sum_{i=1}^{m} c_i \left[l_i(\sigma) - e_i(\sigma) + 1 \right],$$

Por tanto, la función objetivo quedaría:

$$\sum_{i=1}^{|E|} 1 \times [l_i(\sigma) - e_i(\sigma) + 1] = \sum_{i=1}^{|E|} [l_i(\sigma) - e_i(\sigma)] + |E| \\ = \sum_{\{v_i, v_j\} \in E} |\sigma(i) - \sigma(j)| + |E|.$$

Si existe una permutación σ para OLA con $\sum |\sigma(i) - \sigma(j)| \leq B$, entonces se debe cumplir que el resultado de la función objetivo sea $\leq B + |E|$, por ser el mínimo. La conversa también es cierta. Esto completa la reducción polinomial y demuestra que el problema es NP-Hard. □

11 Método de Solución

12 ILS

Dado que el problema es NP-Hard se utiliza el método Iterated Local Search (ILS) con los siguientes componentes clave:

12.1 Construcción de la Solución Inicial

- Se utiliza una estrategia de inserción secuencial simple para construir el horario inicial.
- Intenta colocar cada escena en un orden que minimice los costos de los actores.

12.2 Heurística de Búsqueda Local

- Utiliza dos estructuras de vecindario:
  - Relocate: Mueve una escena de su posición actual a otra posición.
  - Swap: Intercambia las posiciones de dos escenas.
- Evalúa los movimientos utilizando una función objetivo aumentada que aproxima el costo total.

12.3 Estrategia de Perturbación

- Aplica modificaciones aleatorias al horario para escapar de mínimos locales.
- Se seleccionan aleatoriamente dos estrategias:
  - Intercambio aleatorio: Intercambia las posiciones de dos escenas.
  - 2-opt aleatorio: Invierte una subcadena aleatoria de la secuencia de escenas.

12.4 Criterios de Aceptación

- Utiliza una regla de aceptación por umbral:
  - Se acepta una nueva solución si su costo es como máximo $(1 + \Delta)$ veces el mejor costo actual.
  - El umbral Δ disminuye gradualmente con el tiempo.

12.5 Condición de Terminación

- El algoritmo se detiene cuando se alcanza un número fijo de iteraciones.
- La heurística de inserción inicial mejora la calidad de la solución inicial, haciendo que la convergencia sea más rápida.
- Los movimientos de intercambio y reubicación permiten una mejor exploración del espacio de búsqueda.
- El paso de perturbación ayuda a escapar de mínimos locales realizando modificaciones controladas.
- La estrategia de aceptación por umbral permite soluciones temporales peores, mejorando la optimización global.
- Fijar el número de iteraciones proporciona un tiempo de ejecución predecible, incluso si no siempre encuentra la mejor solución.

13 Análisis de Complejidad Temporal y Correctitud

13.1 Complejidad Temporal

La complejidad de los métodos propuestos se desglosa así:

Componente	**Complejidad Teórica**
Modelo MILP	$O\big(2^{n \cdot p} \cdot m\big)$ (exponencial en el peor caso)
Iterated Local Search (ILS)	$O\big(T \cdot (I + P) \cdot n^3\big)$,

Table 1: Complejidad teórica de los enfoques presentados en el paper

Detalles:

- **MILP:** Se utiliza *(*OR-Tools/SCIP) para resolver la formulación MILP. Debido a la naturaleza NP-hard del problema, el tiempo de ejecución puede crecer exponencialmente con respecto a n, p y m. Para instancias pequeñas, el solver encuentra soluciones óptimas en un plazo razonable; sin embargo, a medida que n y p aumentan, la resolución puede tardar mucho más tiempo.

- **ILS:** La complejidad viene dada por el número de iteraciones T y las operaciones en cada búsqueda local y etapa de perturbación, donde T = número de iteraciones, I = tamaño del vecindario y P = número de perturbaciones. En [4] se demuestra empíricamente que esta aproximación heurística obtiene buenas soluciones en menor tiempo que el método exacto al crecer el tamaño del problema.

13.2 Correctitud

La correctitud de los métodos se fundamenta en:

1. **Cumplimiento de restricciones**: El modelo MILP del paper incluye todas las ecuaciones necesarias para representar:
   - El orden de las escenas y su encadenamiento (restricciones tipo *precedencia*).
   - La planificación dentro del horizonte de tiempo, respetando los límites de cada escena.
   - Los costos de actores y locaciones, garantizando que se sumen correctamente en la función objetivo.

   Al resolver dichas restricciones con un solver que sigue métodos de *branch-and-bound* y técnicas de corte, se asegura la factibilidad de la solución hallada.

2. **Optimalidad de la solución**:
   - **MILP:** En [4] se muestra que, para instancias pequeñas o medianas, el método exacto con CPLEX converge a soluciones óptimas, confirmadas por la verificación de *gaps* muy reducidos (o nulos) respecto al *bound* inferior calculado.
   - **ILS:** Aunque el *Iterated Local Search* no ofrece garantía de optimalidad, el paper señala que, en la práctica, se acerca a soluciones de alta calidad en menos tiempo para instancias de mayor tamaño. Se justifica que el ILS mantiene la corrección en el sentido de generar siempre soluciones factibles (cumplen las mismas restricciones que el MILP), pero no garantiza el óptimo global.

14 Resultados Experimentales

En esta sección se presentan los resultados obtenidos al ejecutar las implementaciones inspiradas en los algoritmos descritos anteriormente. Con el fin de reproducir, hasta donde es posible, la metodología de [4], se han desarrollado dos algoritmos principales:

1. Un método *exacto* basado en un modelo MILP simplificado que asigna días a cada escena suponiendo un orden predeterminado.
2. Una *heurística ILS (Iterated Local Search)* que, de forma análoga a lo expuesto en el paper, introduce una solución inicial, luego aplica movimientos de vecindad y perturbaciones, y finalmente se repite un cierto número de iteraciones.

Si bien no se cuenta con la misma formulación MILP completa ni con CPLEX, se han respetado las ideas centrales del documento, incluyendo la manera de generar instancias aleatorias con el número de escenas, actores y horizonte de planificación en rangos similares. De esta manera, los

resultados permiten observar el comportamiento de un enfoque exacto simulado en comparación con la búsqueda local iterativa, siguiendo la esencia de la evaluación propuesta en [4].

Los algoritmos en el artículo fueron implementados en C++ compilado con GCC 9.3.0, utilizando CPLEX 12.10 para resolver el MILP, ejecutando las pruebas en un ordenador de escritorio con sistema operativo Kubuntu 20.04, equipado con un procesador AMD Ryzen 3700X a 4.0GHz y 16GB de RAM. Debido a la diferencia de recursos computacionales se decidió proceder con tres instancias para demostrar la aplicabilidad de los algoritmos.

A continuación, se exhiben las tablas que recogen los valores obtenidos para distintas instancias generadas. La primera tabla 2 recopila los datos correspondientes al método "MILP" simulado, mientras que la segunda tabla 3 expone los valores que caracterizan el desempeño de la heurística ILS en cada instancia.

Table 2: Resultados del método MILP

id	**n**	**m**	**p**	**Z(MILP)**	**gap(MILP)(%)**	**time(MILP)(s)**
0	25	14	189	3.16387e+06	0	0.05
1	9	18	46	1.02375e+06	0	0
2	10	18	47	4.72324e+05	0	0

Table 3: Resultados del método ILS (DLS)

id	**Zbest**	**gap'(%)**	**gap(%)**	**time(ILS)(s)**
0	3.03644e+06	-4.03	0.3	138.22
1	9.88486e+05	-3.44	0	18.91
2	4.23896e+05	-10.25	0	14.42

- **id:** Identificador de la instancia. Cada instancia se numera de manera secuencial.
- **n:** Número de escenas que componen la instancia.
- **m:** Número de actores requeridos en total.
- **p:** Horizonte de planificación (en días) disponible para filmar todas las escenas.
- **Z(MILP):** Costo total hallado por el "método MILP" simulado (en este código, se considera el orden natural de escenas y se resuelven únicamente los días de inicio).
- **gap(MILP)(%):** Brecha del método MILP frente a un posible óptimo. En este ejemplo, aparece en 0% por simplificación.
- **time(MILP)(s):** Tiempo, en segundos, que tardó la parte "MILP" en ejecutarse (muy reducido al ser un subproblema simple).
- **Zbest:** Mejor costo encontrado por el algoritmo de ILS en sus 10 corridas.

- **gap'(%):** Diferencia porcentual entre *Zbest* del ILS y *Z(MILP)*. Un valor negativo indica que ILS obtuvo una solución de menor costo que el MILP simulado.
- **gap(%):** Brecha promedio de las 10 corridas de ILS con respecto a la mejor. Si es 0, todas las corridas encontraron el mismo valor óptimo local.
- **time(ILS)(s):** Tiempo promedio de ejecución (en segundos) del ILS en cada una de sus corridas.

Las tres filas representan distintas instancias, cada una con diferentes dimensiones (*n, m, p*), salarios y costos de locación aleatorios. Se observa, por ejemplo, que en la primera instancia (id = 0), el método MILP simple reporta un costo de 3.16×10^6, mientras que el ILS logra 3.03×10^6, mostrando un gap'(%) de -4.03%. El *time(ILS)(s)* de 138.22 indica el promedio de segundos que tomó resolver dicha instancia a lo largo de las 10 corridas. En la segunda y tercera instancias (id = 1 y 2), se aprecian situaciones análogas, con $gap(\%) = 0$ en las corridas de ILS, lo que significa que todas las ejecuciones encontraron la misma solución óptima local.

Paper 3

15 Movie Scene Scheduling Problem (MSSP)

16 Descripción del Problema

El problema de la Programación de Escenas de Películas (MSSP) busca optimizar la secuencia de filmación de escenas para reducir el costo total de producción. Se considera un conjunto de escenas que deben ser filmadas, donde cada escena puede requerir diferentes locaciones y la participación de actores. Los costos asociados incluyen los costos de traslado entre locaciones y los salarios de los actores, que se acumulan por cada día que un actor está contratado, desde su primera hasta su última escena filmada. El objetivo es determinar una secuencia de filmación de escenas que minimice la suma de estos costos.

17 Formulación Matemática

El problema se modela como un **problema de programación lineal entera no continua**. La secuencia de intercambio de escenas representa la dirección de búsqueda de la solución, y el valor de ajuste de la partícula es el valor del costo total de la película bajo la secuencia de filmación. El modelo busca optimizar el orden de las escenas para minimizar el costo total, que depende de los costos de traslado entre escenas y los salarios de los actores durante el tiempo de rodaje.

18 Métodos de Solución

19 Tabu Search Based Method (TSBM)

Dado que el problema es NP-Hard, se implementa el método de Búsqueda Tabú (TSBM). TSBM es un algoritmo de optimización iterativo que mejora una solución candidata moviéndose repetida-

mente a una solución vecina en el espacio de búsqueda. Para evitar ciclos y escapar de óptimos locales, mantiene una lista tabú de movimientos recientemente realizados. Los componentes clave de TSBM para el MSSP son:

19.1 Estructura del Dominio y Lista Tabú

- **Estructura del Dominio:** Generada mediante movimientos de dominio de modos 2-opt y 2-swap, que determinan el número de soluciones vecinas, la estructura de las soluciones y la relación entre ellas.
- **Lista Tabú:** Registra las posiciones de movimiento, que son las posiciones de selección e intercambio de 2-opt y 2-swap. La lista tabú es de longitud fija y se actualiza de forma FIFO, eliminando los movimientos más antiguos a medida que se añaden nuevos, para evitar caer en óptimos locales.

19.2 Estrategia Tabú y Nivel de Ruptura Prohibido

- **Estrategia Tabú:** Registra las soluciones buscadas con la lista tabú a través de la estructura del dominio para evitar la repetición de soluciones iterativas y caer en un bucle.
- **Nivel de Ruptura Prohibido:** Coordina la estrategia tabú, permitiendo aceptar un movimiento tabú si lleva a una solución mejor que la mejor solución histórica, evitando perder regiones de solución potencialmente óptimas.

19.3 Estrategia de Selección

- Selecciona la mejor solución del dominio como solución inicial para la siguiente iteración. La selección se basa en la función de aptitud, que en este caso es el costo total de filmación. Se elige la solución que minimiza el valor de la función objetivo.

19.4 Procedimiento Básico

1. Se genera una solución inicial aleatoria, que representa la secuencia inicial de filmación.
2. Se evalúa la función de aptitud de la solución inicial. Se inicializa una lista tabú vacía y se define la longitud de la lista tabú y el número máximo de iteraciones.
3. Se genera un conjunto de soluciones candidatas mediante movimientos 2-opt y 2-swap.
4. Para cada solución candidata, se verifica si está en la lista tabú y si cumple con el nivel de ruptura prohibido.
5. Se selecciona la mejor solución no tabú o la mejor solución que cumple el nivel de ruptura prohibido como la nueva solución actual.
6. Se actualiza la lista tabú con el movimiento realizado para llegar a la nueva solución.
7. Se repiten los pasos 3-6 hasta alcanzar el criterio de parada (número máximo de iteraciones).

20 Particle Swarm Optimization Based Method (PSOBM)

También se implementó el método de Optimización por Enjambre de Partículas (PSOBM). PSOBM es un algoritmo heurístico de optimización estocástico basado en inteligencia de enjambre. En PSOBM, una población de partículas (soluciones candidatas) se mueve en el espacio de búsqueda. El movimiento de cada partícula está influenciado por su mejor posición histórica y la mejor posición global encontrada por el enjambre. Los componentes clave de PSOBM para el MSSP son:

20.1 Concepto Básico

- Inspirado en el comportamiento de búsqueda de alimento de las aves. Cada partícula representa una posible secuencia de filmación y se mueve a través del espacio de soluciones.
- Cada partícula mantiene una posición y una velocidad, y su movimiento está guiado por la mejor posición que ha encontrado (pbest) y la mejor posición encontrada por todo el enjambre (gbest).

20.2 Flujo Básico de PSOBM

1. Se inicializa un enjambre de partículas con posiciones y velocidades aleatorias. El número de partículas y el número máximo de iteraciones se establecen como parámetros.
2. Para cada partícula, se calcula el valor de la función de aptitud, que es el costo total de filmación para la secuencia de escenas representada por la partícula.
3. Se compara el valor de aptitud de cada partícula con su pbest. Si la aptitud actual es mejor, se actualiza pbest con la posición actual de la partícula.
4. Se compara el valor de aptitud de cada partícula con el gbest. Si la aptitud actual es mejor que gbest, se actualiza gbest con la posición actual de la partícula.
5. Se actualiza la velocidad y la posición de cada partícula. La velocidad se actualiza considerando la inercia, la atracción hacia pbest y la atracción hacia gbest. La posición se actualiza sumando la velocidad a la posición actual.
6. Se repiten los pasos 2-5 hasta alcanzar el criterio de parada (número máximo de iteraciones o si no hay mejora en gbest durante un número de iteraciones).

21 Ant Colony Optimization Based Method (ACOBM)

Finalmente, se aplicó el método de Optimización por Colonia de Hormigas (ACOBM). ACOBM es un algoritmo de optimización heurístico probabilístico inspirado en el comportamiento de búsqueda de alimento de las hormigas. En ACOBM, un conjunto de hormigas virtuales construye soluciones moviéndose a través de los componentes del problema. Las hormigas depositan feromonas en los caminos que toman, y la concentración de feromonas influye en las decisiones de las hormigas subsiguientes. Los componentes clave de ACOBM para el MSSP son:

21.1 Concepto Básico

- Basado en el comportamiento de búsqueda de alimento de las hormigas reales. Las hormigas exploran diferentes caminos para encontrar alimento y depositan feromonas en los caminos más cortos y eficientes.
- En ACOBM, las hormigas construyen secuencias de filmación moviéndose de escena en escena. La probabilidad de elegir una escena como siguiente en la secuencia depende de la cantidad de feromonas depositadas en el camino a esa escena y de una función heurística.

21.2 Flujo Básico de ACOBM

1. Se inicializan los niveles de feromonas en todos los posibles traslados entre escenas. Se establecen parámetros como el número de hormigas, el número máximo de iteraciones, el coeficiente de evaporación de feromonas, el factor heurístico y el factor de feromonas.
2. Cada hormiga se coloca en una escena de inicio aleatoria y construye una solución (secuencia de filmación) moviéndose secuencialmente de escena en escena. La elección de la siguiente escena se basa en una probabilidad que depende de los niveles de feromonas y de la información heurística.
3. Una vez que todas las hormigas han construido sus soluciones, se calcula el valor de la función objetivo (costo total de filmación) para cada solución.
4. Se actualizan los niveles de feromonas. Las feromonas se evaporan en todos los caminos, y se depositan feromonas adicionales en los caminos utilizados por las hormigas que encontraron mejores soluciones. La cantidad de feromonas depositadas es proporcional a la calidad de la solución.
5. Se repiten los pasos 2-4 hasta alcanzar el criterio de parada (número máximo de iteraciones).

22 Resultados Experimentales

Para evaluar el rendimiento de TSBM, PSOBM y ACOBM, se realizaron experimentos comparando el costo total de filmación obtenido por cada método y sus tiempos de ejecución. En general, los resultados sugieren que, para el MSSP, ACOBM proporciona las mejores soluciones en términos de costo, aunque a un costo computacional ligeramente mayor que TSBM, mientras que TSBM ofrece un buen equilibrio entre calidad de solución y tiempo de computación. PSOBM, aunque también optimiza el problema, tiende a ser menos eficiente en términos de calidad de solución y tiempo de cómputo en comparación con TSBM y ACOBM para este problema específico.

23 Conclusiones

Este informe ha presentado varias alternativas al problema de programación del plan de rodaje para un filme, demostrando la complejidad de encontrar soluciones exactas para ellas, así como explicando algunos de los algoritmos utilizados para hallar soluciones aproximadas.

Referencias

References

[1] T. Cheng, J. Diamond, y B. M. Lin, "Optimal scheduling in film production to minimize talent hold cost," *Journal of Optimization Theory and Applications*, vol. 79, no. 3, pp. 479–492, 1993.

[2] Garey, M. R., Johnson, D. S., & Stockmeyer, L. (1976). Some Simplified NP-Complete Graph Problems. *Theoretical Computer Science*, 1(3), 237–267.

[3] Garey, M. R., & Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

[4] 1st Thu Trang Hoa y 2nd Minh Anh Nguyen, "An Iterated Local Search for the Talent Scheduling Problem with Location Costs," *Phenikaa University*, Hanoi, Vietnam, 2024.

[5] Xiaoqing Long y Jinxing Zhao, "Scheduling Problem of Movie Scenes Based on Three Meta-Heuristic Algorithms," *IEEE Access*, vol. 8, pp. 59091-59099, 2020. DOI: 10.1109/ACCESS.2020.2982664.